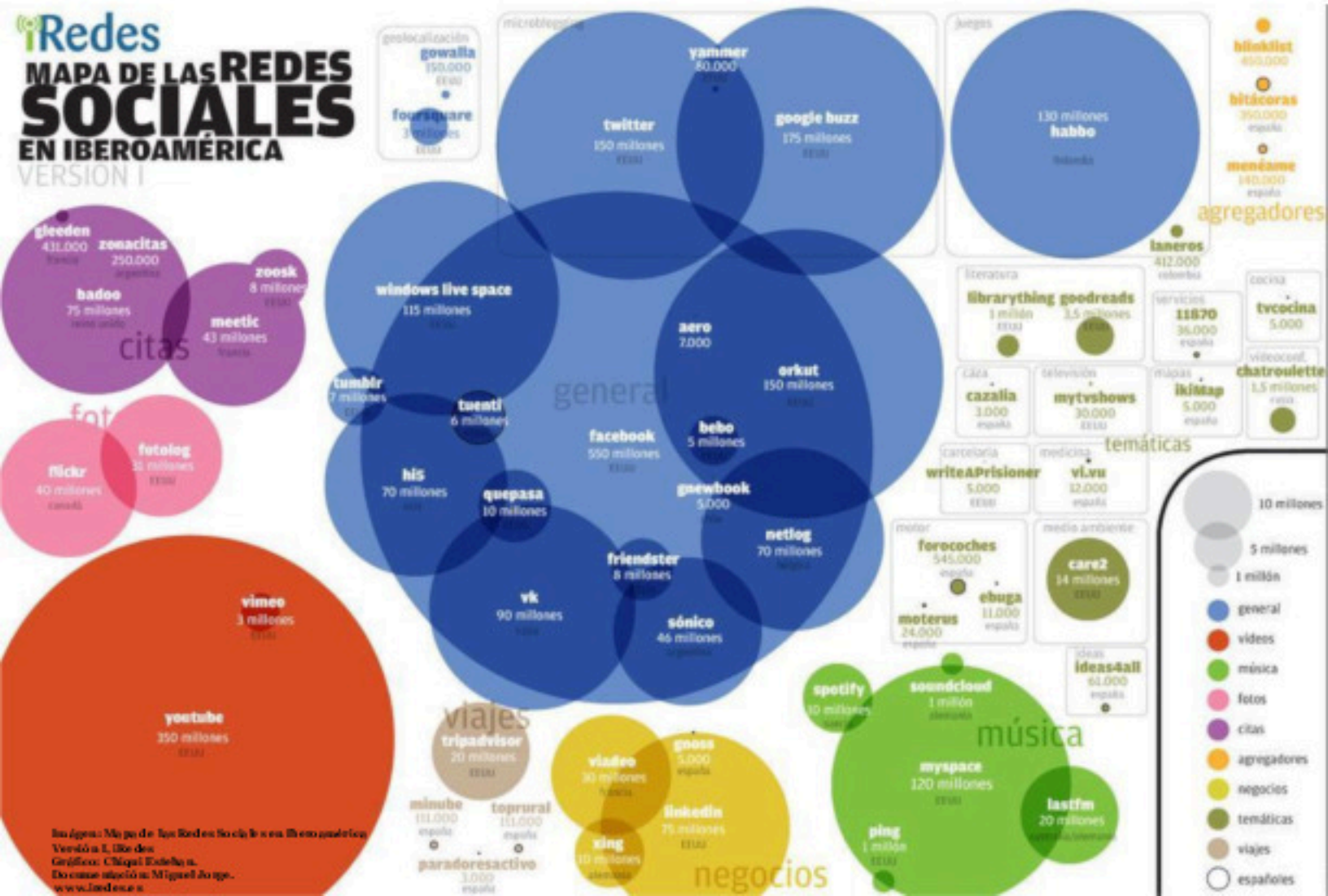
Manejo de Datos

Antonio Carrillo Ledesma
Karla Ivonne González Rosas



Manejo de Datos

Antonio Carrillo Ledesma y Karla Ivonne González Rosas
Facultad de Ciencias, UNAM
http://academicos.fciencias.unam.mx/antoniocarrillo

La última versión de este trabajo se puede descargar de la página:
https://sites.google.com/ciencias.unam.mx/acl/en-desarrollo
http://132.248.181.216/acl/EnDesarrollo.html

2025, Versión 1.0α[1]

[1]El presente trabajo está licenciado bajo un esquema Creative Commons Atribución CompartirIgual (CC-BY-SA) 4.0 Internacional. Los textos que componen el presente trabajo se publican bajo formas de licenciamiento que permiten la copia, la redistribución y la realización de obras derivadas siempre y cuando éstas se distribuyan bajo las mismas licencias libres y se cite la fuente. ¡Copia este libro! ... Compartir no es delito.

Índice

1 Introducción **6**
1.1 Computadoras Actuales 7
1.2 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias 45
1.3 Agradecimientos 49

2 Sistemas Operativos **50**
2.1 Windows 67
2.2 UNIX y BSD 74
2.3 Apple y sus macOS e iOS 76
2.4 GNU/Linux 81
2.5 Android 95
2.6 Chromebook y Chrome OS 98
2.7 Otros Sistemas Operativos 101

3 Seguridad, Privacidad y Vigilancia **107**
3.1 ¿Qué es la Privacidad y por qué es Importante? 108
3.2 Las Vulnerabilidades y Exposiciones Comunes 115
3.3 Alfabetismo Digital 117
3.4 Amenazas a la Ciberseguridad 119
3.5 Dicen que si no Pagas por un Producto, Entonces el Producto Eres Tú. 126
3.6 Datos que Recopila Google 129
3.7 ¿Cómo me Protejo? 132

4 Búsquedas Básicas en la Red **137**
4.1 Algunas Búsquedas Básicas Usando Google 137
4.2 Algunas Funciones y Cosas que Puedes hacer con IA 149

5 Búsquedas Avanzadas en la Red **155**
5.1 Qué es un motor de búsqueda 155
5.2 Tipos de Buscadores 155
5.3 Buscadores Verticales 156
5.4 Cuántos Motores de Búsqueda Existen 157
5.5 Principales Motores de Búsqueda 158
5.6 Para qué Sirven los Motores de Búsqueda 159

5.7 Buscadores para Obtener Respuestas Rápidas 160
5.8 ¿Para qué Saber Sobre los Motores de Búsqueda? 166
5.9 Averiguar Todo Sobre Cualquier Persona en Internet 170

6 Programación y Lenguajes de Programación 176
6.1 Paradigmas de Programación 179
6.2 Lenguaje de Programación 182
6.3 Conceptos Transversales 187
6.4 Algo de Programación 194
6.5 Introducción a los Paradigmas de Programación 207
6.6 Errores de Redondeo y de Aritmética 216
6.7 Documentación del Código Fuente 227
6.7.1 Documentar en C, C++ y Java 230
6.7.2 Documentar en Python 237
6.8 Los Diagramas UML para Documentar 241

7 Programación Orientada a Objetos 246
7.1 Herencia y Composición 252
7.2 Clase Abstracta 265
7.3 Polimorfismo y Programación Genérica 273
7.4 Excepciones 278
7.5 Altas, Bajas y Cambios 286
7.5.1 Ejemplo en Java 286
7.5.2 Ejemplo en Python 298

8 Entornos de Desarrollo y Herramientas de Programación 305
8.1 Java 308
8.2 Python 317
8.3 Julia 334
8.4 C y C++ 337
8.5 Fortran 344
8.6 R 350
8.7 Herramientas de Programación 351
8.7.1 ¿Qué es eso de ASCII, ISO-8859-1 y UTF-8? 357
8.7.2 Uso de Espacios o Tabuladores en Fuentes 361
8.7.3 Comparar Contenido de Fuentes 363
8.7.4 Astyle 364
8.7.5 Compilación y la Optimización del Ejecutable 365

8.7.6 Análisis de Rendimiento y Depuración . . . 370
8.7.7 Mejora del Rendimiento en Python . . . 375
8.7.8 Git . . . 380
8.7.9 GitLab vs GitHub . . . 399
8.7.10 Otras opciones . . . 404
8.8 Programando Desde la Nube . . . 406

9 Bases de Datos 410
9.1 Clasificación . . . 410
9.2 ¿Qué son las bases de datos SQL? . . . 412
9.2.1 Lenguaje de definición de datos (DDL) . . . 415
9.2.2 Lenguaje de manipulación de datos DML . . . 417
9.3 ¿Qué son las bases de datos NoSQL? . . . 431
9.3.1 SQL en comparación con NoSQL . . . 434
9.3.2 SQL en comparación con Terminología NoSQL . . . 436
9.4 ¿Qué son las bases de datos de Series de Tiempo? . . . 442
9.5 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias . . . 447
9.6 Programando en Diferentes Manejadores de Bases de Datos . . . 451
9.6.1 SQLite . . . 452
9.6.2 MySQL/MariaDB . . . 455
9.6.3 PostgreSQL . . . 462
9.6.4 MongoDB . . . 465
9.6.5 Redis . . . 471
9.7 LAMP y LEMP . . . 481
9.8 Lenguajes de Programación y Bases de Datos SQL . . . 483
9.8.1 Java y el Acceso a Base de Datos Relacionales . . . 484
9.8.2 Python y el Acceso a Base de Datos Relacionales . . . 491
9.9 Seguridad en Bases de Datos . . . 493
9.9.1 Cifrado . . . 493
9.9.2 Restringir el Acceso y Personalizar la Configuración Predeterminada . . . 494
9.9.3 Asegúrese de que los Servidores sean Físicamente Seguros 494
9.9.4 Usar Herramientas de Evaluación de la Seguridad de la Base de Datos . . . 495
9.9.5 Pruebas de Penetración . . . 496
9.10 Programando desde la Web . . . 498
9.10.1 SQLite . . . 498

9.10.2 MySQL y MariaDB . . . 499
9.10.3 PostgreSQL . . . 500
9.10.4 MongoDB . . . 501

10 Intercambio de Información 504
10.1 Archivos de Formatos Abiertos . . . 510
10.2 Estándares Según el Tipo de Datos . . . 521
10.3 ¿Qué son los Metadatos? . . . 525
10.4 Qué son los NFTs ¿Burbuja, Moda o Nuevo Paradigma? . . . 526
10.5 La Web 3 . . . 534
10.6 Protegiendo Nuestros Metadatos . . . 537

11 POO Java vs Python 541
11.1 Ejemplo de Clases Java vs Python . . . 541
11.2 Atributos de Objetos . . . 543
11.3 Métodos y Funciones . . . 554
11.4 Herencia y Polimorfismo . . . 555
11.5 Reflexión . . . 564

12 Apéndice A: Software Libre y Propietario 569
12.1 Software Propietario . . . 572
12.2 Software Libre . . . 574
12.3 Seguridad del Software . . . 581
12.4 Tipos de Licencias . . . 584
12.4.1 Licencias Creative Commons . . . 590
12.4.2 Nuevas Licencias para Responder a Nuevas Necesidades 592
12.5 Implicaciones Económico-Políticas del Software Libre . . . 595
12.5.1 Software Libre y la Piratería . . . 595
12.5.2 ¿Cuánto Cuesta el Software Libre? . . . 596
12.5.3 La Nube y el Código Abierto . . . 599
12.5.4 El Código Abierto como Base de la Competitividad . . . 602
12.5.5 Software Libre en Empresas y Corporaciones . . . 603
12.6 Código Abierto y las Organizaciones Internacionales . . . 611
12.6.1 Las Naciones Unidas y el Código Abierto . . . 612
12.6.2 La Comisión Europea se Compromete a Liberar Todo el Software que Pueda Beneficiar a la Sociedad . . . 613

13 Apéndice B: Algunos Consejos para Programadores **616**
13.1 ¿Cómo Hago mi Programa? 616
13.2 Diez Principios de la Usabilidad 617
13.3 Principios SOLID de la Programación Orientada a Objetos . . 618
13.4 10 Principios de Diseño para POO 621
13.5 Errores en Programación 624
13.5.1 Errores en Java 629
13.5.2 Errores en Pyhton 635

14 Bibliografía **640**

1 Introducción

Con el aumento del acceso a internet por parte de personas y empresas se ha generado un auge de las redes sociales, comercio electrónico y demás actividades que implican el uso del cómputo, generando un aumento exponencial de la información y datos que se generan a los cuales se puede tener acceso. El gran reto del manejo de datos es el poder recolectar, almacenar, estructurar y analizar grandes cantidades de información en el menor tiempo posible que pueden ser usados para la toma de decisiones, adquirir conocimiento y otras tantas nuevas posibilidades que se abren con el paso de los años.

Comencemos por algunos conceptos fundamentales:

- Dato: cualquier valor aislado
- Contexto: para darle sentido al dato es necesario ponerlo en un contexto, el cual será fundamental para realizar una interpretación
- Información: la interpretación de un dato con un contexto se convierte en información
- Patrones: cuando se analiza la información se pueden descubrir patrones; es decir, que hay ciertas cosas que se repiten constantemente a lo largo del tiempo bajo determinadas condiciones
- Conocimiento: al descubrir patrones se puede generar conocimiento, por qué basándose en casos anteriores es posible predecir qué va a pasar en determinadas situaciones
- Toma de decisiones: con predicciones y conocimiento de un tema es posible tomar una decisión informado sobre escenarios futuros

El manejo de datos, es una creciente parte de la ciencia, que algunas veces es llamada ciencia de datos. Esta es usada en por ejemplo:

- Medicina: análisis de enfermedades o síntomas que se han detectado en cierta población y, con base en eso, saber si son susceptibles de adquirir alguna otra enfermedad
- Clima: a partir del análisis de ciertas condiciones climatológicas, se hacen predicciones sobre el estado del tiempo

- Mercadotecnia: a partir del análisis del patrón de comportamiento humano, se puede predecir si las personas comprarán o rechazarán un producto, en donde es preferible poner una tienda y un largo etc.

Así, mediante el análisis descriptivos de la consulta de la información almacenada localmente en bases de datos o la que se puede descargar de internet y gracias al uso de estadística y probabilidad, minería de datos y la incorporación de la inteligencia artificial y el aprendizaje de máquina se han podido descubrir patrones y modelos, y lograr análisis predictivos y prescriptivos entre muchos otros.

Para poder hacer realidad todo lo descrito anteriormente, ha sido necesario un desarrollo vertiginoso de nuevo y potente Hardware (que soporte el almacenamiento de la ingente cantidad de información y su interconexión mediante el uso de internet) como de toda una gama de Software especializado para soportar los nuevos sistemas de almacenamiento para los diferentes tipos de datos y para la recuperación de grandes volúmenes de información.

1.1 Computadoras Actuales

La computadora (también conocida como ordenador) actual es una máquina digital programable que ejecuta una serie de comandos para procesar los datos de entrada, obteniendo convenientemente información que posteriormente se envía a las unidades de salida. Una computadora está formada físicamente por numerosos circuitos integrados y varios componentes de apoyo, extensión y accesorios, que en conjunto pueden ejecutar tareas diversas con suma rapidez y bajo el control de un programa (Software).

La constituyen dos partes esenciales, el Hardware, que es su estructura física (circuitos electrónicos, cables, gabinete, teclado, etc.), y el Software, que es su parte intangible (programas, datos, información, documentación, etc.).

Con respecto al Hardware[1], se encuentra compuesto por una serie de

[1] En Debian GNU/Linux podemos instalar la aplicación *lshw* para conocer los distintos componentes de la computadora, mediante:

```
# apt install lshw
```

Así, para ver de forma resumida los dispositivos que componen la computadora usamos:

```
# lshw -short
```

dispositivos, clasificados según la función que estos desempeñen. Dicha clasificación se compone de:

- Los dispositivos de entrada son todos aquellos que permiten la entrada de datos a una computadora. Estos dispositivos (periféricos), son los que permiten al usuario interactuar con la computadora. Ejemplos: teclado, Mouse (ratón), micrófono, Webcam, Scanner, etc.
- Los dispositivos de salida, son todos aquellos que permiten mostrar la información procesada por la computadora. Ejemplos: monitor, impresora, auriculares, altavoces, etc.
- Los dispositivos de comunicación son aquellos que permiten la comunicación entre dos o más computadoras. Ejemplos: Modem, Router, placa de red, Bluetooth, etc.
- Los dispositivos de almacenamiento, son todos aquellos que permiten almacenar datos en el ordenador. Ejemplos: disco duro, Pendrive, Diskette, CD y DVD, etc.
- Los dispositivos de cómputo, son aquellos encargados de realizar las operaciones de control necesarias, sobre el resto de los dispositivos la computadora.

La unidad central de procesamiento (Central Processing Unit CPU) se comunica a través de un conjunto de circuitos o conexiones llamada Bus de datos o canal de datos. El bus conecta la CPU a los dispositivos de almacenamiento, los dispositivos de entrada y los de salida.

Desde el punto de vista funcional es una máquina que posee, al menos, una unidad central de procesamiento, unidad de memoria (Random Access Memory RAM , Read Only Memory ROM y Caché) y dispositivos de entrada/salida (periféricos). Los periféricos de entrada permiten el ingreso de datos, la CPU se encarga de su procesamiento (operaciones aritmético-lógicas) y los dispositivos de salida los comunican a los medios externos. Es así, que la computadora recibe datos, los procesa y emite la información resultante, la que luego puede ser interpretada, almacenada, transmitida a

Si necesitamos más detalle usamos:

```
# lshw
```

otra máquina o dispositivo o sencillamente impresa; todo ello a criterio de un operador o usuario y bajo el control de un programa de computación.

CPU la Unidad Central de Proceso (Central Processing Unit CPU) es aquella parte del procesador que se encarga de ejecutar las diversas acciones que ordenemos al dispositivo que debe llevar a cabo. La CPU es el componente básico dentro de todo dispositivo inteligente, ya que prácticamente cualquier proceso que se ordene al sistema pasa por él. Con el paso del tiempo, además, su eficiencia y calidad ha alcanzado grandes cotas, aunque tecnologías al alza como las NPU han supuesto un mayor salto cualitativo que el de aumentar la potencia bruta de la CPU.

GPU, para el procesamiento gráfico la Unidad de Procesamiento Gráfico (Graphics Processing Unit GPU) es el apartado que se dedica a las acciones de mayor peso: las de componente gráfico. De este modo, acciones como la ejecución de videojuegos, o la edición y renderizado de vídeos, se llevan a cabo a través de la GPU del sistema. La calidad del teléfono, ordenador u otro dispositivo inteligente, suele ir supeditada a menudo a la calidad de su GPU, que dependerá de la banda de precio del aparato en cuestión. Eso sí, si los otros componentes no tienen la misma calidad, se puede producir el temido cuello de botella.

En los smartphones, la GPU ya va integrada en los procesadores, pero en los PC's, como AMD y NVIDIA se muestran como marcas especializadas en las GPUs. Cuentan con todo tipo de gamas y también poseen un amplio abanico de precios, un valor siempre directamente proporcional a la calidad y capacidad de sus gráficas, y a su vigencia en el mercado.

NPU, redes neuronales en tu dispositivo la Unidad de Procesamiento Neuronal (Neural Processing Unit NPU), a diferencia de la GPU, que cuenta con un funcionamiento paralelo al de la CPU, puede encargarse de funciones similares a las de la CPU, pero lo hace de un modo mucho más eficiente. Impulsada por Inteligencia Artificial, una NPU es capaz de priorizar procesos para ejecutarlos de un modo exponencialmente más veloz y con un consumo mucho menor. En móviles, se usa especialmente para mejorar el procesado de fotografías, aunque participa en muchos otros procesos.

Además, esta arquitectura todavía tiene años de progreso por delante, a diferencia de la CPU y la GPU, cuyas mejoras ya son de carácter más leve

y basadas en aumentar la potencia bruta. La tecnología NPU, en cambio, lleva menos tiempo entre nosotros y todavía tiene mucho margen de mejora para ofrecer un rendimiento cada vez más poderoso.

Desde la llegada de los procesadores de más de un núcleo a PC, como consecuencia de la imposibilidad de hacerlos escalar en potencia solo por velocidad de reloj, la forma de entender los diferentes chips cambió. Han pasado ya dos décadas de dicho cambio y ante la inminente salida de los Chips disgregados o por Chiplets al mercado de masas, no está de más recordar la organización más común en el mundo del Hardware en todo este tiempo.

¿Qué es un SoC? las siglas SoC significan System on a Chip y hace referencia a todo chip que tiene la mayoría de componentes integrados en una misma pieza de silicio sin llegar a ser un microcontrolador. Se trata de la pieza de Hardware más usada por el hecho de que a día de hoy todo procesador para PC, teléfono móvil, consola de videojuegos, televisor o incluso servidores, es un SoC y pese a las diferencias entre ellos, todos tienen una organización común.

En realidad, todas las CPU actuales son SoC, ya que se trata de "varias CPU" diseñadas para funcionar alrededor de un elemento de intercomunicación central. Este se encarga de interconectar los diferentes elementos entre sí y de darles acceso a interfaces externas.

Por ejemplo, con la memoria RAM (u otros), a la que está asociado el controlador de memoria que comparten todos sus elementos o a los periféricos, y a los cuales se puede acceder directamente con una serie de interfaces específicas, o a través de un Chipset externo encargado de gestionar los diferentes periféricos y componentes.

En PC, debido a que la comunicación con los periféricos se hace a través del uso de direcciones de memoria de la RAM principal, los componentes relacionados con esta se encuentran subordinados al controlador de memoria. Por lo que son una pieza más conectada a la parte central.

¿Qué es una APU y en qué se diferencia de un SoC? las siglas APU significan Accelerated Processor Unit y fue usada por AMD cuando sus CPU e iGPU (o GPU integrada, la veremos a continuación) no traían consigo ningún sistema de gestión de E/S (entrada y salida). Sin embargo, la cosa empezó a cambiar ya con la arquitectura "Carrizo" que fue el nombre clave de los últimos SoC antes del lanzamiento de los AMD Ryzen, donde se

incorporaron varias interfaces de periféricos directamente en la CPU.

A día de hoy todo es un SoC, lo que ocurre es que, en sobremesa, las torres tienen tanta conectividad y capacidad de expansión que se suele emplear un Chipset, y lo mismo ocurre en estaciones de trabajo y servidores, pero no en el resto. Y es que si hablamos de un Chip para un PC portátil, una consola o un móvil, entonces al no existir tantas interfaces para periféricos y otros componentes, entonces estás se pueden integrar en un mismo chip.

En realidad, el término APU es más bien comercial de AMD y al día de hoy se utiliza como sinónimo de SoC, pero se puede resumir en que una APU carece de cualquier gestión de periféricos y requiere de un Chip externo para ello. Mientras que un SoC tiene las especificaciones mínimas en ese aspecto, pese a ser también ampliables. Para simplificar la idea, un SoC es una APU más completa.

¿¡GPU qué es y cuáles son sus características concretas frente a una dGPU? las siglas iGPU corresponden a integrated GPU, y hace referencia a todo componente de este tipo que se encuentre integrado en una APU o un SoC. Por lo que se trata de procesadores gráficos de potencia limitada que se pueden ver lastrados en velocidad de reloj por el problema del ahogamiento termal (Thermal Throttling) que se produce cuando muchas partes comparten el mismo espacio físico.

Si bien, es posible llegar a ciertos niveles de rendimiento que son aceptables de cara a reproducir las escenas en 3D a tiempo real en los videojuegos, y otras tareas de carácter profesional donde se usa una tarjeta gráfica, estas se ven cuanto menos limitadas:

El hecho de tener que compartir espacio en el mismo Chip con los diferentes núcleos de la CPU produce que esta no pueda alcanzar la misma velocidad de reloj que se alcanzaría siendo un Chip aparte e independiente.

En PC, debido a que como la memoria RAM usa DDR o LPDDR, el ancho de banda es pequeño y hemos de partir del hecho de que el rendimiento de todo Chip gráfico, incluido una iGPU, depende del ancho de banda que se le puede otorgar con dicha memoria RAM.

Los SoC con iGPU actuales tienen un tamaño fijo, definido este por la cantidad de pines soportados por la interfaz con la placa base. Esto limita el tamaño, no solo del Chip, sino también de la gráfica integrada en el mismo.

iGPU en consolas de videojuegos al contrario de lo que ocurre con los PC, las consolas de videojuegos no tienen que seguir una serie de normas respecto a sus componentes. Para empezar, no ven el tamaño de sus chips limitados por un estándar de placa base, dado que son productos únicos y exclusivos. Esto les permite tener el tamaño que quieran, incluso más que una CPU convencional para PC, lo que les permite tener una iGPU en su SoC mucho más avanzada.

El otro punto es la memoria utilizada, ya que la de las tarjetas gráficas también se usan como memoria principal. Esta da el ancho de banda necesario para que la iGPU alcance cierto nivel de rendimiento, pero su latencia es mucho más alta que la RAM convencional de PC que está más optimizada en ese aspecto, por lo que el rendimiento de su CPU suele ser más bajo que su equivalente para ordenador.

¿Qué es una dGPU en un PC o portátil? las dGPU, o GPU dedicadas, no son otra cosa que las GPU de toda la vida, pero la particularidad es que también son SoC, ya que tiene varios núcleos, especializados en tareas gráficas, alrededor de una interfaz central y compartiendo todos ellos un mismo acceso a memoria.

Sin embargo, carecen de núcleos de CPU en su interior, de ahí a que no se les llame APU o SoC, por el hecho de que de existir estos elementos pasarían a ser una iGPU. Por lo tanto, su principal particularidad de las dGPU es que tienen su propia memoria RAM (SDRAM para ser concretos) la cual suele rodear estos Chips, ya sea en forma de tarjeta gráfica o soldados en la placa de los ordenadores portátiles. A esta la llamamos VRAM y es de uso exclusivo de la dGPU o GPU.

Los equipos de cómputo los podemos clasificar[2] por:

- Equipos móviles: estos equipos buscan un equilibrio entre su capacidad de cómputo versus el rendimiento energético de sus baterías -para operar el mayor tiempo posible sin recargarse- y su peso, entre estos equipos destacan las Laptops, Notebook, Netbook, Ultrabook, tabletas, teléfonos inteligentes, etc.

[2]El ordenador del Apollo 11, el Block II, funcionaba a una velocidad de 2 MHz y tenía 2 KB de memoria RAM y 32 KB de memoria ROM. Para ponerlo en contexto, el chip de un cargador USB-C moderno es 563 veces más potente que la computadora que se usó en el Apollo 11, al menos en términos de potencia bruta.

- Equipos de escritorio: estos equipos al estar permanentemente conectados a la corriente eléctrica pueden tener un mayor número de componentes y disponen de una mejor capacidad disipación de calor por lo que pueden contener una mayor cantidad componentes, como discos, RAM o tarjetas de video y el tamaño, peso o consumo energético no es un inconveniente.

- Servidores: son equipos que suelen atender a múltiples usuarios simultáneamente y disponen de gran cantidad de Cores, RAM, disco y son interconectados por red de alta velocidad con otros servidores para atender las crecientes necesidades de los centros de datos los cuales deben estar permanentemente en operación. Generalmente los equipos son montados en Racks con otras decenas de equipos, por lo que su arquitectura se ve limitada a una moderada generación de calor por parte de sus componentes ya que su sistema de ventilación es por aire para todo el Rack.

- Estaciones de trabajo: son equipos individuales diseñados para atender cargas computacionales intensas, por lo que requieren Hardware más complejo y potente como puede ser múltiples tarjetas de video (con decenas de miles de Cores gráficos) , discos (con cientos de Terabytes), gran cantidad de RAM (pueden llegar a superar el Terabyte) y sistema de refrigeración por aire o líquido, etc.

- Cómputo intensivo: son equipos interconectados por red de alta velocidad con procesadores y tarjetas gráficas dedicadas para el cálculo numérico que soportan cargas intensas por largos periodos de tiempo, los más comunes son los que forman parte de los Cluster que llegan a tener millones de cores.

FLOPS Una medida relativamente objetiva para analizar el rendimiento de un dispositivo suele ser medir sus operaciones de punto flotante por segundo o más conocidas como FLOPS. Hay que tener en cuenta que la medición de FLOPS es muy compleja porque las diferentes operaciones en punto flotante llevan diferentes cantidades de tiempo para ejecutarse. Y no todo el mundo utiliza las mismas operaciones para establecer los cálculos.

Por ejemplo, una división simple como 1/5, toma significativamente menos tiempo que el cálculo del logaritmo de 5. Por eso, se estableció el algoritmo

de Linpack como un estándar representativo con el que poder medir todos los sistemas bajo el mismo baremo de FLOPS.

Es importante señalar que el algoritmo de Linpack utiliza el formato en punto flotante de doble precisión (64-bit). Sin embargo, como veremos la mayoría de los valores que dan los fabricantes son con precisión simple (32-bit). Además, los valores que dan los fabricantes suelen ser teóricos y en la práctica suelen ser inferiores debido a otros factores limitantes como la frecuencia de reloj o la velocidad de las memorias ROM y RAM.

Por tanto, aunque todos hemos acabado midiendo el rendimiento en FLOPS[3], no es una medida absoluta de la potencia del CPU ni de una GPU. Por ejemplo para algunos dispositivos tenemos:

Móviles El SoC Snapdragon 821 que monta una GPU Adreno 530 tiene una potencia de 519.2 gigaFLOPS (0.52 TFLOPS), y los Chips Apple A9X del iPad Pro alcanzan los 345.6 gigaFLOPS (0.35 TFLOPS), todos ellos medidos con precisión simple de 32-bits.

CPU

- Intel Xeon W-3245: 1.4 TFLOPS
- Intel Core i9-9900X: 1.2 TFLOPS
- AMD Ryzen 9 3950X: 1.1 TFLOPS

Los procesadores de gama media-alta rondan el medio TFLOPS:

- AMD Ryzen 7 3700X: 546.0 GFLOPS - 0.55 TFLOPS
- Intel Core i9-9900: 499.0 GFLOPS - 0.50 TFLOPS
- AMD Ryzen 5 3600X: 461.0 GFLOPS - 0.46 TFLOPS

[3] El Cray-1 fue puesto marcha en 1975 y utilizaba una CPU a 80 MHz y llevaba integrada una unidad SIMD de 64 bits de precisión de punto flotante, lo cual fue un salto de gigante que permitió un salto de los 3 MFLOPS de potencia del CDC 6600 a los 160 MFLOPS en el Cray-1.

Tarjetas Gráficas Ojo: La tabla está ordenada por los valores en precisión simple (32-bit) primer columna

GPU	FP32 TFLOPS	FP64 TFLOPS
TITAN V	13.8	6.9
Radeon RX Vega 64	12.7	0.8
GeForce GTX 1080 Ti	11.3	0.4
GeForce GTX 1080	8.9	0.3
Radeon R9 Fury X	8.6	0.5
Radeon HD 7990	7.8	1.9
GeForce GTX 1070	6.5	0.2
Radeon RX 480	5.8	0.4
GeForce GTX 690	5.6	0.2
Radeon R9 290X	5.6	0.7
GeForce GTX 780 Ti	5.3	0.2
Radeon HD 6990	5.1	1.3
GeForce GTX 980	4.9	0.15
Radeon RX 470	4.9	0.3
Radeon R9 290	4.8	0.6
GeForce GTX Titan	4.7	1.5
GeForce GTX 1060	4.4	0.14
Radeon HD 7970 GHz	4.3	1.1
GeForce GTX 780	4.1	0.17
Radeon R9 280X	4.0	1.0
Radeon R9 280	3.3	0.83
GeForce GTX 680	3.1	0.13
Radeon HD 7950	2.9	0.71

Como podemos ver, las tarjetas gráficas de Nvidia, normalmente, tienen una potencia muy alta en precisión simple, pero muy mala en precisión doble. La precisión simple es la que se usa en los juegos, pero la precisión doble es la que se utiliza en los cálculos complejos científicos y en el minado de muchas criptomonedas.

Consolas Todas ellos son en valores de precisión simple (32-bit)

- PlayStation 4: 1.3 TFLOPS
- Xbox One: 1.8 TFLOPS

- PlayStation 4 Pro: 4.2 TFLOPS
- Nintendo Switch: entre 0.4 y 0.5 TFLOPS
- PlayStation 5 promete una GPU con 10.28 TFLOPS
- La Xbox Series X promete una GPU de 12 TFLOPS

SuperCómputo Se dio a conocer en noviembre del 2023 la publicación de la 62^a edición del ranking de las 500 computadoras de mayor rendimiento del mundo:

- El Frontier, ubicado en el Laboratorio Nacional Oak Ridge del Departamento de Energía de Estados Unidos, sigue posicionado en el primer lugar (manteniendo desde mediados del año pasado). El clúster tiene 8.7 millones de núcleos de procesador y proporciona un rendimiento de 1.194 exaflops, el doble que el clúster, que ocupa el segundo lugar (con un menor consumo de energía).
- El nuevo clúster Aurora, desplegado en el Laboratorio Nacional Argonne del Departamento de Energía de Estados Unidos. El clúster tiene casi 4.8 millones de núcleos de procesador y proporciona un rendimiento de 585 petaflops, que es 143 petaflops más que el clúster que anteriormente ocupaba el segundo lugar.
- El clúster Eagle, lanzado este año por Microsoft para la nube Azure. El clúster contiene 1.12 millones de núcleos de procesador (CPU Xeon Platinum 8480C 48C 2GHz) y demuestra un rendimiento máximo de 561 petaflops. El Software del clúster está basado en Ubuntu 22.04.

El umbral mínimo de rendimiento para ingresar al Top 500 durante 6 meses fue de 2.02 petaflops (hace seis meses, 1.87 petaflops). Hace cinco años, sólo 272 clusters mostraban un rendimiento de más de un petaflop, hace seis años 138 y hace siete años - 94). Para el Top 100, el umbral de entrada aumentó de 6.3 a 7.89 petaflops, y para el Top 10, de 61.44 a 94.64 petaflops.

El rendimiento total de todos los sistemas en la clasificación durante 6 meses aumentó de 5.2 a 7 exaflops (hace cuatro años era 1.650 exaflops y hace seis años, 749 petaflops). El sistema que cierra el ranking actual ocupaba en la última edición el puesto 454.

Para poner en contexto los avances en este campo, en el año 2004 IBM era dueña y señora del mundo de la supercomputación, su espectacular BlueGene/L dominaba la lista TOP.500. Aquel monstruo contaba con 32,768 procesadores PowerPC 440 a 700 MHz y 16 TB de memoria. 20 años después una sola NVIDIA GeForce RTX 4090 con 24 GB de memoria GDDR6X es más potente que esa supercomputadora -lo es al menos en rendimiento bruto-, BlueGene/L contaba en ese momento con un rendimiento de 70.72 TFLOPS, pero la propia NVIDIA dejaba claro en el lanzamiento de sus RTX 4090 que estas tarjetas gráficas contaban con una potencia de 83 TFLOPS.

Es más, cuatro RTX 4090 con soporte FP8 logran también rivalizar con la supercomputadora más potente de 2009. Y eso sin apretarle las tuercas a las RTX 4090: en noviembre de 2022 es precisamente lo que hicieron en Wccftech y lograron que la RTX 4090 se convirtiera en la primera tarjeta gráfica del mundo en alcanzar los 100 TFLOPS.

Esa comparación es como decimos real en esa potencia de cálculo en bruto, pero también es cierto que en esa y otras supercomputadoras se tenian mecanismos especiales de comunicación entre procesadores o de transferencia de datos, algo para lo que las GPUs actuales, aún siendo sobresalientes, no están tan optimizadas.

¿Cómo Trabaja una Computadora? Todas las computadoras sean de uno o más procesadores ejecutan los programas realizando los siguientes pasos:

1. Se lee una instrucción
2. Se decodifica la instrucción
3. Se encuentra cualquier dato asociado que sea necesario para procesar la instrucción
4. Se procesa la instrucción
5. Se escriben los resultados

Esta serie de pasos, simple en apariencia, se complican debido a la jerarquía de memoria RAM, en la que se incluye la memoria Caché, la memoria principal y el almacenamiento no volátil como pueden ser los discos duros o de estado sólido (donde se almacenan las instrucciones y los datos del

programa), que son más lentos que el procesador en sí mismo. Con mucha frecuencia, el paso (3) origina un retardo muy largo (en términos de ciclos del procesador) mientras los datos llegan en el bus de la computadora.

Durante muchos años, una de las metas principales del diseño microinformático ha sido la de ejecutar el mayor número posible de instrucciones en paralelo, aumentando así la velocidad efectiva de ejecución de un programa. No obstante, estas técnicas han podido implementarse en Chips semiconductores cada vez más pequeños a medida que la fabricación de estos fue progresando y avanzando, lo que ha abaratado notablemente su costo.

El procesador es el cerebro de un ordenador. No hay que olvidar otros componentes como la memoria, el almacenamiento o la tarjeta gráfica dedicada, desde luego, pero el procesador está un escalafón por encima en la jerarquía.

Piensa que, si cambiamos el procesador en dos equipos con la misma memoria, almacenamiento o tarjeta gráfica, el comportamiento puede variar notablemente. Sin embargo, para un mismo procesador, los cambios en el resto de componentes no impactan de forma tan directa en la experiencia de uso de un equipo.

¿Qué es una CPU? Antes de nada, vamos a definir exactamente lo que es una CPU o un procesador. Como bien indican sus siglas en inglés (Central Processing Unit) es la unidad de procesamiento -puede ser Intel, AMD, ARM, etc- encargada de interpretar las instrucciones de un Hardware haciendo uso de distintas operaciones aritméticas y matemáticas. Características principales de un procesador:

- Frecuencia de reloj. Este primer término hace referencia a la velocidad de reloj que hay dentro del propio procesador. Es un valor que se mide en Mhz o Ghz y es básicamente la cantidad de potencia que alberga la CPU. La mayoría de ellas cuentan con una frecuencia base -para tareas básicas- y otra turbo que se utiliza para procesos más exigentes -con un aumento en el consumo de energía y por ende un aumento en la temperatura del procesador, requiriendo sistemas de disipación de calor eficientes-.

- Consumo energético. Es normal que nos encontremos con CPU 's donde su consumo energético varía notablemente. Es un valor que se muestra en vatios (W) y como es obvio, aquellos procesadores de gama superior,

serán más propensos a consumir más energía. Ante esto, es importante contar con un eficiente sistema de enfriamiento además de contar con una fuente de alimentación acorde a la potencia requerida por el procesador, la tarjeta gráfica y sus respectivos sistemas de enfriamiento.

- Número de núcleos. Con el avance de la tecnología, ya es posible encontrar tanto procesadores de Intel como de AMD que cuentan ya con decenas de núcleos. Estos cores son los encargados de llevar a cabo multitud de tareas de manera simultánea.

- Número de hilos. Si un procesador tiene Hyperthreading en el caso de Intel o SMT (Simultaneous Multi-Threading) en el caso de AMD, significa que cada uno de los núcleos es capaz de realizar dos tareas de manera simultánea, lo que se conoce como hilos de proceso. Por lo tanto, un procesador de cuatro núcleos físicos con Hyperthreading tendría ocho hilos de proceso, y sería capaz de ejecutar ocho órdenes al mismo tiempo -los hilos no tienen las mismas capacidades de un core real y en muchos casos su uso merma el rendimiento del CPU, pero los sistemas operativos los reconocen como si fueran cores reales-, por ejemplo el procesador AMD EPYC Genoa con 96 núcleos y 192 hilos.

- Memoria Caché. A la hora de "recordar" cualquier tarea, el propio ordenador hace uso de la memoria RAM. Sin embargo no es eficiente este proceso y por tanto es necesario que utilice la memoria Caché de la CPU para paliar esta deficiencia. El Caché se caracteriza porque se llega a ella de forma más rápida y puede ser tipo L1, L2 y L3.

- Zócalo. Es el tipo de conector con pines o Socket al que se conecta la placa base. Por ejemplo, las últimas de Intel suelen tener el Socket LGA 1200, mientras que las de AMD con Ryzen son AM4.

- Red. Si bien la red es un recurso indispensable en un equipo de cómputo, en el caso de equipos paralelos la velocidad de la red es el mayor cuello de botella en cuanto a rendimiento, por ello es necesario usar redes de alto desempeño como las de InfiniBand con un alto costo económico pero de alto desempeño que pueden llegar al orden de cientos de Gigabytes por segundo.

Nuevos Procesadores la creciente demanda de dispositivos de cómputo ha generado una gran variedad de procesadores, los podemos clasificar como:

- Procesador compuesto por múltiples núcleos de alta eficiencia -con un consumo energético reducido- que sacrifican potencia de procesamiento en aras de extender la carga útil de las baterías de los dispositivos móviles.
- Procesador compuesto por múltiples núcleos de alto rendimiento que pueden estar al tope de su capacidad sin generar excesivo calor y son especialmente usados en servidores y en cómputo intensivo.
- Procesadores compuestos por múltiples núcleos de alto rendimiento que pueden ajustar su velocidad de reloj de manera dinámica para tratar cargas de trabajo pesadas por un cierto tiempo -pues generan gran cantidad de calor-, por lo que requieren un sistema eficiente de enfriamiento, son ideales para estaciones de trabajo.
- Procesadores compuestos por múltiples núcleos híbridos que en lugar de tener un único tipo de núcleo multipropósito, estos Chips cuentan con dos grupos de núcleos. El primero de ellos, compuesto por múltiples núcleos de alta eficiencia, se encarga de procesar las tareas más livianas o en segundo plano que deba realizar un procesador, todo ello, con un consumo energético menor. El otro grupo, compuesto por múltiples núcleos de alto rendimiento, sigue una dinámica opuesta, su consumo ener-gético es superior, pero únicamente entran en funcionamiento cuando la tarea en cuestión requiere un extra de procesamiento.

Para gestionar esta división de núcleos híbridos, se ha integrado un "Thread Director", un elemento que se encarga de determinar qué núcleo procesa cada tarea. Las compañías, además, ha modificado cómo funciona la caché de sus procesadores:

- Cada núcleo de rendimiento tiene su propia caché L2.
- Cada cluster de núcleos de eficiencia tiene una "piscina" de memoria L2 común, de la que beben todos los núcleos que sean partícipes.

- Tanto los núcleos de rendimiento como los de eficiencia tienen acceso a una "piscina" de memoria L3 común para todos ellos.

Otros de los cambios que impactarán en el desempeño de las CPUs es el aumento de velocidad y una mayor cantidad de memoria Caché, compatibilidad con memorias DDR6 y con la interfaz PCIe 6.0.

PCIe PCI Express (Peripheral Component Interconnect Express), abreviado como PCIe, es una tecnología de conexión de Hardware utilizada para la comunicación de alta velocidad entre diferentes componentes de un equipo informático. Este estándar se ha convertido en la interfaz más habitual para la conexión de tarjetas de expansión, como tarjetas gráficas que sirven para correr juegos, tarjetas de sonido, tarjetas de red y dispositivos de almacenamiento de alta velocidad.

Una de las ventajas destacadas de PCI Express es su arquitectura de canales independientes, que permiten la transferencia simultánea de datos en ambos sentidos. Cada carril tiene una tasa de transferencia específica, medida en gigabits por segundo (Gbps), y la capacidad de un slot PCIe se expresa como el número de carriles que tiene. Esto se traduce en un ancho de banda total mayor, lo que facilita la conexión de dispositivos que requieren altas tasas de transferencia, como las tarjetas gráficas modernas o los dispositivos de almacenamiento de última generación.

Los diferentes tipos de ranuras de PCI Express según su tamaño PCIe X1 carriles 1, pines 18, PCIe x4 carriles 4, pines 32, PCIe x8 carriles 8, pines 49, PCIe x16 Carriles 16, pines 82.

Adicionalmente, también es interesante fijarse en las diferentes versiones que se han ido lanzando desde que PCIe se lanzó al mercado:

- PCIe 1.0 ancho banda 8 GB/s, velocidad de transferencia 2.5 GT/s
- PCIe 2.0 ancho banda 16 GB/s, velocidad de transferencia 5 GT/s
- PCIe 3.0 ancho banda 32 GB/s, velocidad de transferencia 8 GT/s
- PCIe 4.0 ancho banda 64 GB/s, velocidad de transferencia 16 GT/s
- PCIe 5.0 ancho banda 128 GB/s, velocidad de transferencia 32 GT/s
- PCIe 6.0 ancho banda 256 GB/s, velocidad de transferencia 64 GT/s

Características Arquitectónicas los procesadores Intel x86 admiten un formato de precisión extendido de 80 bits con un significado de 64 bits, que es compatible con el especificado en el estándar IEEE. Cuando un compilador usa este formato con registros de 80 bits para acumular sumas y productos internos, está trabajando efectivamente con un redondeo unitario de 2^{-64} en vez de 2^{-53} para precisión doble, dando límites de error más pequeños en un factor de hasta $2^{11} = 2048$.

Algunos procesadores Intel y AMD tienen una operación fusionada de multiplicación y suma (FMA), que calcula una multiplicación y una suma combinadas $x + yz$ con un error de redondeo en lugar de dos. Esto da como resultado una reducción en los límites de error por un factor 2.

Las operaciones FMA de bloques de precisión mixta $D = C + AB$, con matrices A, B, C y D de tamaño fijo, están disponibles en las unidades de procesamiento tensorial de Google, las GPU NVIDIA y en la arquitectura ARMv8-A. Para entradas de precisión media, estos dispositivos pueden producir resultados de calidad de precisión simple, lo que puede proporcionar un aumento significativo en la precisión cuando los bloques FMA se encadenan para formar un producto matricial de dimensión arbitraria.

Meltdown y Spectre El tres de enero del 2018 se dio a conocer al público, que 6 meses antes se habían detectado dos distintos fallos en los procesadores de los equipos de cómputo, comunicaciones y redes de internet que usamos. Esto para dar tiempo a los desarrolladores de procesadores y de sistemas operativos de implementar estrategias para mitigar el problema. Estos son problemas de diseño de los procesadores de Intel, AMD, IBM POWER y ARM, esto significa que procesos con privilegios bajos -aquellos que lanzan las aplicaciones de usuarios convencionales- podían acceder a la memoria del Kernel del sistema operativo[4].

Un ataque que explota dicho problema permitiría a un Software malicioso espiar lo que están haciendo otros procesos y también espiar los datos que están en esa memoria en el equipo de cómputo (o dispositivo móvil) atacado. En máquinas y servidores multiusuario, un proceso en una máquina virtual

[4]En GNU/Linux, el Kernel (si usamos una versión actualizada) nos indica las fallas del procesador a las que es vulnerable, usando:

```
$ cat /proc/cpuinfo
$ lscpu
```

podría indagar en los datos de los procesos de otros procesos en ese servidor compartido.

Ese primer problema, es en realidad solo parte del desastre. Los datos actuales provienen especialmente de un grupo de investigadores de seguridad formados por expertos del llamado Project Zero[5] de Google. Ellos han publicado los detalles de dos ataques (no son los únicos[6]) basados en estos fallos de diseño. Los nombres de esos ataques son Meltdown y Spectre. Y en un sitio Web dedicado a describir estas vulnerabilidades destacan que "aunque los programas normalmente no tienen permiso para leer datos de otros programas, un programa malicioso podría explotar Meltdown, Spectre y apropiarse de secretos almacenados en la memoria de otros programas". Como revelan en su estudio, la diferencia fundamental entre ambos es que Meltdown permite acceder a la memoria del sistema, mientras que Spectre permite acceder a la memoria de otras aplicaciones para robar esos datos.

Ya que Meltdown y Spectre son problemas de diseño en los procesadores, no es posible encontrar solución por Hardware para los procesadores existentes y dado que constantemente aparecen nuevas formas de explotar dichos fallos, la única manera de mantener el equipo de cómputo, comunicaciones y redes de internet a salvo es mediante Software que debe implementar las soluciones en los sistemas operativos. En particular en el Kernel de Linux se trabaja en parchar en cada versión del Kernel todos los fallos reportados, por esto y por otra gama de fallos e inseguridades es necesario mantener siempre el sistema operativo y sus aplicaciones actualizadas.

Como se había comentado anteriormente, estos problemas de diseño afectan a todos los procesadores Intel, AMD, IBM POWER y ARM. Eso incluye básicamente a todos los procesadores que están funcionando al día de hoy[7]

[5] https://googleprojectzero.blogspot.com/

[6] Entre las distintas vulnerabilidades detectadas y sus variantes resaltan: Meltdown (AC, DE, P, SM, SS, UD, GP, NM, RW, XD, BR, PK, BND), Spectre (PHT, BTB, RSB, STL, SSB, RSRE), PortSmash, Foreshadow, Spoiler, ZombieLoad (1 y 2), Kaiser, RIDL, Plundervolt, LVI, Take a Way, Collide+Probe, Load+Reload, LVI-LFB, MSD, CSME, RYZENFALL (1, 2, 3, 4), FALLOUT (1, 2, 3) , CHIMERA (FW, HW), MASTERKEY (1, 2, 3), SWAPGS, ITLB_Multihit, SRBDS, L1TF, etc. Más información en:

https://cve.mitre.org
https://meltdownattack.com/

[7] Solo en el año 2021 se detectaron 16 vulnerabilidades en procesadores INTEL y 31 en los procesadores AMD.

en nuestros equipos, ya que estos procesadores llevan produciéndose desde 1995. Afecta a una amplia gama de sistemas.

En el momento de hacerse pública su existencia se incluían todos los dispositivos que no utilizasen una versión convenientemente parcheada de IOS, GNU/Linux, MacOS, Android, Windows y Android. Por lo tanto, muchos servidores y servicios en la nube se han visto impactados, así como potencialmente la mayoría de dispositivos inteligentes y sistemas embebidos que utilizan procesadores con arquitectura ARM (dispositivos móviles, televisores inteligentes y otros), incluyendo una amplia gama de equipo usado en redes. Se ha considerado que una solución basada únicamente en Software para estas fallas alenta los equipos de cómputo entre un 20 y un 40 por ciento dependiendo de la tarea que realizan y el procesador del equipo.

Memoria RAM La memoria RAM (Random Access Memory) o memoria de acceso aleatorio es un componente físico de nuestro ordenador, generalmente instalado sobre la misma placa base. La memoria RAM es extraíble y se puede ampliar mediante módulos de distintas capacidades.

La función de la memoria RAM es la de cargar los datos e instrucciones que se ejecutan en el procesador. Estas instrucciones y datos provienen del sistema operativo, dispositivos de entrada y salida, de discos duros y todo lo que está instalado en el equipo.

En la memoria RAM se almacenan todos los datos e instrucciones de los programas que se están ejecutando, estas son enviadas desde las unidades de almacenamiento antes de su ejecución. De esta forma podremos tener disponibles todos los programas que ejecutamos. Se llama memoria de acceso aleatorio porque se puede leer y escribir en cualquiera de sus posiciones de memoria sin necesidad de respetar un orden secuencial para su acceso.

De forma general existen o han existido dos tipos de memorias RAM. Las de tipo asíncrono, que no cuentan con un reloj para poder sincronizarse con el procesador. Y las de tipo síncrono que son capaces de mantener la sincronización con el procesador para ganar en eficacia y eficiencia en el acceso y almacenamiento de información en ellas. Veamos cuales existen de cada tipo.

Memorias de Tipo Asíncrono o DRAM las primeras memorias DRAM (Dinamic RAM) o RAM dinámica eran de tipo asíncrono. Se denomina DRAM por su característica de almacenamiento de información de

forma aleatoria y dinámica. Su estructura de transistor y condensador hace que para que un dato quede almacenado dentro una celda de memoria, será necesario alimentar el condensador de forma periódica.

Estas memorias dinámicas eran de tipo asíncrono, por lo que no existía un elemento capaz de sincronizar la frecuencia del procesador con la frecuencia de la propia memoria. Esto provocaba que existiera menor eficiencia en la comunicación entre estos dos elementos.

Memorias de Tipo Síncrono o SDRAM a diferencia de las anteriores esta memoria RAM dinámica cuenta con un reloj interno capaz de sincronizar esta con el procesador. De esta forma se mejoran notablemente los tiempos de acceso y la eficiencia de comunicación entre ambos elementos. Actualmente todos nuestras computadoras cuentan con este tipo de memorias operando en ellos. Las principales tipos de memoria son: DDR, DDR2, DDR3, DDR4 y la nueva DDR5. Donde las tasas de transferencia (GB/s) son: DDR (2.1 - 3.2), DDR2 (4.2 - 6.4), DDR3 (8.5 - 14.9), DDR4 (17 - 25.6) y DDR5 (38.4 - 51.2).

Aparte las características propias de cada una de las diferentes memorias DDR, la característica más importante es que, por ejemplo, en la memoria DDR4 cuatro cores pueden acceder simultáneamente a ella y en la DDR5 serán cinco cores.

Caché L1, L2 y L3 La memoria Caché es otra de las especificaciones importantes de los procesadores, y sirve de manera esencial de la misma manera que la memoria RAM: como almacenamiento temporal de datos. No obstante, dado que la memoria Caché está en el procesador en sí, es mucho más rápida y el procesador puede acceder a ella de manera más eficiente, así que el tamaño de esta memoria puede tener un impacto bastante notable en el rendimiento, especialmente cuando se realizan tareas que demandan un uso intensivo del CPU como en el cómputo de alto desempeño o cómputo científico.

La Caché se divide en diferentes jerarquías de acceso:

- La Caché L1 es el primer sitio donde la CPU buscará información, pero también es la más pequeña y la más rápida, a veces para mayor eficiencia, la Caché L1 se subdivide en L1d (datos) y L1i (instrucciones), actualmente los procesadores modernos en cada core tiene su propio cache de datos e instrucciones.

- La Caché L2 suele ser más grande que la L1 pero es algo más lenta. Sin embargo, por norma general es la que mayor impacto tiene en el rendimiento, este también está incluido en cada core.

- La Caché L3 es mucho más grande que las anteriores, y generalmente se comparte entre todos los núcleos del procesador (a diferencia de las anteriores, que normalmente van ligadas a cada core). Este tercer nivel es en el que buscará el procesador la información tras no encontrarla en la L1 y L2, por lo que su tiempo de acceso es todavía mayor.

Para poner en contexto la relevancia de la memoria Caché, supongamos que el acceso a los datos de la memoria Caché L1 por el procesador es de dos ciclos de reloj, el acceso a la memoria Caché L2 es de 6 ciclos de reloj, el acceso a la memoria Caché L3 es de 12 ciclos y el acceso a la RAM es de 32 ciclos de reloj.

Además supongamos que la operación suma y resta necesitan de 2 ciclos de reloj para completar la operación una vez que cuente con los datos involucrados en dicha operación, que la multiplicación requiere 4 ciclos de reloj para completar la operación, la división necesita 6 ciclos de reloj para completar la operación y estamos despreciando el tiempo necesario para poner los datos del Caché L1 a los registros del procesador para poder iniciar el cálculo, así también despreciamos el tiempo requerido para sacar el resultado de los registros del procesador al Caché L1.

Esto nos da una idea del número máximo teórico de operaciones básicas que un procesador puede realizar por segundo dependiendo de la velocidad de reloj de la CPU[8].

Si nosotros necesitamos hacer la multiplicación de una matriz $\underline{\underline{A}}$ es de tamaño $n \times n$ por un vector $\underline{u}$ de tamaño n y guardar el resultado en el vector $\underline{f}$ de tamaño n. Entonces algunos escenarios son posibles:

1. Si el código del programa cabe en el Caché L1 de instrucciones y la matriz $\underline{\underline{A}}$, los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en el Caché L1 de datos, entonces el procesador estará siendo utilizado de forma óptima al hacer los cálculos pues no tendrá tiempos muertos por espera de datos.

[8] Por ejemplo en un procesador AMD Ryzen 9 3900X con 12 Cores (2 Threads por Core) por procesador emulando un total de 24 Cores, corre a una frecuencia base de 3,340 MHz, con una frecuencia mínima de 2,200 MHz y máxima de 4,917 Mhz, con Caché L1d de 384 KiB, L1i de 384 KiB, Caché L2 de 6 MiB y Caché L3 de 64 MiB.

2. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en el Caché L1 de datos pero la matriz $\underline{\underline{A}}$ está dispersa entre los Cachés L1 y L2, entonces el procesador estará teniendo algunos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L2 a L1 para hacer los cálculos y utilizado de forma óptima el procesador mientras no salga del Caché L1.

3. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en el Caché L1 de datos pero la matriz $\underline{\underline{A}}$ está dispersa entre los Cachés L1, L2 y L3, entonces el procesador estará teniendo muchos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L3 y L2 a L1 para hacer los cálculos resultando en mediana eficiencia en el uso del procesador.

4. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en los Cachés L3, L2 y L1 pero los datos de la matriz $\underline{\underline{A}}$ está dispersa entre la RAM y los Cachés L3, L2 y L1, entonces el procesador estará teniendo un exceso de tiempos muertos mientras carga la parte que necesita de la matriz de la RAM a los Cachés L3, L2 y L1 para hacer los cálculos resultando en una gran pérdida de eficiencia en el uso del procesador.

Además, debemos recordar que la computadora moderna nunca dedica el cien por ciento del CPU a un solo programa, ya que los equipos son multitarea[9] y multiusuario[10] por lo que la conmutación de procesos (que se realiza cada cierta cantidad de milisegundos) degrada aún más la eficiencia computacional de los procesos que demandan un uso intensivo de CPU[11].

[9] Cuentan con la capacidad para ejecutar varios procesos simultáneamente en uno o más procesadores, para ello necesitan hacer uso de la conmutación de tareas, es decir, cada cierto tiempo detiene el programa que está corriendo y guardan sus datos, para poder cargar en memoria otro programa y sus respectivos datos y así reiniciar su ejecución por un período determinado de tiempo, una vez concluido su tiempo de ejecución se reinicia la conmutación de tareas con otro proceso.

[10] Se refiere a todos aquellos sistemas operativos que permiten el empleo de sus procesamientos y servicios al mismo tiempo. Así, el sistema operativo cuenta con la capacidad de satisfacer las necesidades de varios usuarios al mismo tiempo, siendo capaz de gestionar y compartir sus recursos en función del número de usuarios que estén conectados a la vez.

[11] Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen

Last Level Cache se le llama Last Level Cache siempre al último nivel de Caché de una CPU, existen dos tipos:

- Last Level Cache Estándar.
- Victim Cache.

Una Victim Cache no actúa como la Caché de último nivel de una CPU, sino que en ese caso lo hace el penúltimo nivel y en la Victim Cache acaban los últimos datos descartados de la Caché y que han sido volcados en la RAM, los cuales son copiados en la Victim Cache para poder acceder a ellos más rápido.

Smart Cache la Smart Cache (o Caché) es esencialmente L3 pero optimizada por Intel para ser más eficiente a la hora de compartir la información en los núcleos de la CPU. A efectos prácticos, se comporta de igual manera que la Caché L3.

Disco Son dispositivos no volátiles (los hay del orden de 20 TB y continuamente incrementan su capacidad), lo que significa que retienen datos incluso cuando no tienen energía. La información almacenada permanece segura e intacta a menos que el disco duro sea destruido o interferido. La información se almacena o se recupera de manera aleatoria en lugar de acceso secuencial. Esto implica que se puede acceder a los bloques de datos en cualquier momento sin necesidad de pasar por otros bloques de datos.

Actualmente, podemos agrupar los discos duros disponibles en cuatro tipos:

---

muy optimizadas intentando conseguir la mejor desenvoltura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado (Liquorix viene optimizado para multimedia y Juegos, por otro lado XanMod tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más para cálculos intensivos) de las últimas versiones estable del Kernel.

- Parallel Advanced Technology Attachment (PATA)
- Serial ATA (SATA)
- Interfaz de sistema de computadora pequeña (SCSI)
- Adjunto de tecnología avanzada paralela
- Unidades de estado sólido (SSD)

En promedio, las velocidades máximas de los discos actuales son:

- Disco SATA3 de 5,400 RPM, Lectura: 102 MB/s, Escritura: 96 MB/s
- Disco SATA3 de 7,200 RPM, Lectura: 272 MB/S, Escritura: 200 MB/s
- Disco SSD SATA, Lectura 550 MB/s, Escritura 520 MB/s
- Disco SSD NVMe, Lectura 6,600 MB/s, Escritura 5,500 MB/s
- Disco SSD PCI 5.0, Lectura 13,000 MB/s, Escritura 12,000 MB/s
- Unidad Flash USB[12] 2.0, 35 MB/s
- Unidad Flash USB 3.0 o 3.1 gen 1, 5 Gbit/s
- Unidad Flash USB 3.0 o 3.1 gen 2, 10 Gbit/s
- Unidad Flash USB 3.2 gen 2x2, 20 Gbit/s

Disco de Estado Sólido SSD Estos son los últimos avances en tecnología de almacenamiento que tenemos en la industria de las computadoras. Son totalmente diferentes de las otras unidades en que no consisten en partes móviles. Tampoco almacenan datos utilizando magnetismo. En su lugar, hacen uso de la tecnología de memoria flash, circuitos integrados o dispositivos semiconductores para almacenar datos de forma permanente, al menos hasta que se borren. Estas son algunas de sus ventajas.

[12]Los colores en los puertos USB son: USB 1.X blanco (12 Mbps), USB 2.X Negro (480 Mbps), USB 3.0 Azul oscuro (5 Gbps), USB 3.1 Azul claro (10 Gbps), USB 3.2 Rojo (20 Gbps). En el caso del USB de color Amarillo, esté es un puerto de carga aún con el dispositivo apagado.

- Acceso a datos más rápido
- Menos susceptible a los golpes
- Menores tiempos de acceso y latencia
- Menos consumo de energía

Los SSD actuales están disponibles tanto en versiones SATA como en versiones M.2, U.2 y en formato de tarjeta PCI Express 4.0. Los tres últimos hacen uso del protocolo NVMe y la interfaz PCI Express 4.0 x4, lo que les permite superar los 6,600 MB/s de velocidades de lectura y escritura, frente a los 550 MB/s que suelen alcanzar como máximo las unidades SATA. La nueva versión PCI 5.0 ofrece un ancho de banda de 32 GT/s el doble de PCI 4.0, permitiendo discos SSD con 13,000 MS/s de velocidad de lectura secuencial y realizar hasta 2,500K operaciones por segundo de lectura aleatoria y tamaño máximo 15.36 TB a un precio exorbitante.

Tarjetas microSD Cuando compramos una tarjeta microSD para ampliar el almacenamiento de nuestro Smartphone, cámara, tableta o cualquier otro dispositivo electrónico la mayoría de la gente normalmente solo se fija en la capacidad de almacenamiento. Ahora bien, ¿qué significan todas esas etiquetas y nombres que llevan adscritas las microSD? ¿Cuál es la diferencia entre una microSDXC y una microSDHC? ¿Es mejor una UHS-I o una UHS-II? ¿Qué quiere decir que una tarjeta es A1 y V30? A continuación, intentamos aclarar toda esta nomenclatura.

Lo primero que tenemos que tener claro es que cuando analizamos una tarjeta micro SD existen multitud de factores que limitan su velocidad y capacidad de almacenamiento. Su rendimiento depende de factores como el tipo de tarjeta que estamos usando, su clase, y otros detalles como el tipo de bus o el número de operaciones que puede realizar por segundo.

Tipos de tarjetas microSD actualmente existen 4 generaciones distintas de tarjetas de memoria microSD. Cuanto más modernas sean, mayores velocidades y almacenamiento podrán ofrecer:

- Tarjetas micro SD (Secure Digital): Estas son las memorias de primera generación. Las desarrolló el fabricante SanDisk y fueron las primeras en utilizar el formato de 15 x 11 x 1 milímetros. Su capacidad máxima es de 32 GB.

- Tarjetas micro SDHC (Secure Digital High Capacity): Tarjetas de segunda generación. Cuentan con un bus de datos mejorado que permite alcanzar velocidades superiores, aunque su capacidad máxima sigue siendo de 32 GB.
- Tarjetas micro SDXC (Secure Digital Extended Capacity): Estas micro SD utilizan un sistema de archivos exFAT y su velocidad de transferencia puede llegar hasta los 312 MB/s. Su capacidad de almacenamiento puede llegar hasta los 2 TB y es el tipo de tarjeta más común utilizado a día de hoy.
- Tarjetas micro SDUC: Estas son las tarjetas de memoria más modernas y punteras. Utilizan el sistema de archivos exFAT y permiten almacenar entre 2 TB y 128 TB de datos.

Evidentemente con esto no es suficiente. Si queremos tener una idea aproximada de la velocidad de la micro SD tendremos que fijarnos en aspectos como la clase y tipo de bus que emplea.

La clase es una característica que nos indica la velocidad de transferencia de datos mínima de la tarjeta de memoria. Actualmente hay 4 tipos de clase diferentes:

- Clase 2: Velocidad mínima de 2 MB/s
- Clase 4: Velocidad mínima de 4 MB/s
- Clase 6: Velocidad mínima de 6 MB/s
- Clase 10: Velocidad mínima de 10MB/s

Hoy en día la mayoría de tarjetas micro SD son de clase 10, ya que son capaces de transferir más de 10 MB/s y superan esa cifra fácilmente.

El bus determina la velocidad de la interfaz de la tarjeta de memoria, y nos puede servir como indicativo para conocer la rapidez con la que se pueden leer y escribir los datos:

- Bus estándar: Su velocidad de transferencia alcanza hasta los 12.5 MB/s. Es el tipo de bus utilizado en tarjetas de clase 2, 4 y 6.
- Bus de alta velocidad (High Speed): Se utiliza en tarjetas de clase 10 y alcanza una velocidad de hasta 25 MB/s.

- Bus Ultra High Speed (UHS): Estos son los buses con la interfaz más rápida, y existen varios tipos:
  - UHS-I: Hay dos tipos de buses UHS-I. Por un lado, tenemos el UHS-I clase 1 (U1) que alcanza velocidades de 50 MB/s. Y luego tenemos el UHS-I clase 3 (U3) que llega hasta los 104 MB/s.
  - UHS-II: Alcanza velocidades de transferencia hasta 312 MB/s.
  - UHS-III: Velocidades de transferencia de datos que alcanzan hasta los 624 MB/s.
- SD-Express: Este es el tipo de bus más potente de todos, llegando hasta los 985 MB/s.

Como referencia, te interesará saber que actualmente la mayoría de tarjetas micro SD de gama media utilizan un bus UHS-I de clase 3 (U3) con velocidades de lectura de hasta 104 MB/s.

Otro factor importante es la velocidad de lectura y escritura aleatoria (IOPS) u operaciones por segundo que puede realizar una tarjeta. Este dato determina el rendimiento mínimo en la lectura y escritura aleatoria de la SD:

- Clase de rendimiento de aplicación A1: Las tarjetas A1 tienen una velocidad mínima de lectura aleatoria de 1,500 IOPS, y una velocidad mínima de escritura aleatoria de 500 IOPS.
- Clase de rendimiento de aplicación A2: Las tarjetas A2 ofrecen velocidades superiores, con 4,000 IOPS de lectura y 2,000 IOPS de escritura.

Normalmente con una tarjeta A1 es más que suficiente para tareas del día a día, aunque si necesitamos un rendimiento superior, por ejemplo, para ejecutar aplicaciones desde la SD o jugar a videojuegos, las tarjetas A2 ofrecen un mejor rendimiento.

La velocidad de escritura aleatoria (A1 y A2) es un dato relevante para los Smartphones y tabletas, pero si tenemos una cámara de grabación, una Action camera o un Dron, la característica en la que nos tenemos que fijar es en el Velocidad de escritura secuencial (sistema V) que utiliza (en inglés, Video Speed Class). O dicho de otra forma, en su velocidad de escritura secuencial.

Esta característica nos indica la cantidad de datos que se pueden grabar en la micro SD de forma constante sin bajar de una velocidad mínima. Esto resulta esencial cuando queremos grabar vídeos en alta y ultra-alta definición:

- V30 (Video Speed Class 30): Velocidad de escritura mínima de 30 MB/s
- V60 (Video Speed Class 60): Velocidad de escritura mínima de 60 MB/s
- V90 (Video Speed Class 90): Velocidad de escritura mínima de 90 MB/s

Por ejemplo, si vamos a grabar vídeo en resolución 4K directamente en la tarjeta micro SD, es necesario que la velocidad V sea lo máximo posible, especialmente si vamos a utilizar un amplio BitRate con bajos niveles de compresión, o calidades superiores como el 8K.

Cintas Magnéticas En el año 2010, se comunicó que todos los datos utilizados para el proyecto del satélite Nimbus se recuperaron de cintas que en ese momento tenían 46 años. A partir de dicho comunicado se extendió el uso de la cinta magnética para almacenamiento de datos en todo el mundo.

En un mundo altamente digital, la cinta magnética es una de las pocas tecnologías que utiliza señales analógicas para mover parte de los datos, en su esencia, la cinta se parece mucho a un HDD, utiliza materiales de base magnética, pero en este caso, la cinta es literalmente una base de material generalmente nailon que tiene un revestimiento magnético. Y en lugar de un disco giratorio, la cinta entra, está enhebrada. Puede parecer una cinta VHS, pero es mucho más robusta.

La cinta se mueve linealmente hacia la unidad de cinta y hacia el cartucho de cinta. Para escribir, el cabezal de la cinta toma señales electrónicas y crea un mini campo magnético que puede cambiar la polaridad del material de la película para formar un patrón de ceros y unos. Una vez que los datos se escriben en la cinta, no se pueden cambiar (pero se pueden borrar y reescribir).

La inmutabilidad y las capacidades de encriptación de la cinta, así como la simplicidad de crear un espacio para almacenarla en una bóveda hacen de la cinta un arma clave para asegurar que los datos sobrevivan frente al Ransomware. Uno de los grandes productores de cintas en la actualidad es IBM, los cuales argumentan que esta función hace que la cinta sea el medio ideal para almacenar datos de archivo a los que no es necesario acceder con frecuencia.

La cinta también puede servir como una copia de seguridad y de versiones fuera de línea de archivos importantes o confidenciales que son resistentes a los ataques cibernéticos. Los tipos de datos que permanecen en la cinta abarcan registros financieros, registros médicos, información de identificación personal y documentos que forman parte de una retención legal de múltiples gobiernos.

Una sola cinta mide aproximadamente 3 pulgadas por 3 pulgadas y 3/4 de pulgada de grosor. Es más pequeño que una unidad de disco duro (HDD), pero pesa alrededor de 0.6 kilogramos. Un cartucho puede almacenar 18 Terabytes de datos sin comprimir y 45 Terabytes comprimidos. IBM está trabajando para duplicar esta capacidad en la próxima generación de la tecnología. En cuanto a la velocidad de recuperación, se obtiene un flujo de datos de una unidad de cinta de 1,000 Megabits por segundo, comprimidos.

Una biblioteca de cintas puede variar en tamaño desde algo que puede poner en su escritorio hasta algo que es del tamaño de un refrigerador pequeño (alrededor de 8 pies cuadrados). La pequeña biblioteca del tamaño de un refrigerador tiene capacidad para 1584 cartuchos. IBM promociona que su biblioteca Diamondback será la biblioteca de cintas más densa del mercado. Podrá contener 69 Petabytes de información mientras ocupa menos de 8 pies cuadrados de espacio.

La cinta magnética supera al disco duro y al flash en cuanto a longevidad, costo financiero y costo de huella de carbono, pero pierde en velocidad de acceso. Las cintas no son recomendables para poner datos de producción en vivo o incluso copias de seguridad, pero son perfectas para cualquier información a la que se acceda con poca frecuencia y que deba conservarse durante mucho tiempo, como registros médicos o datos de archivo.

Es conocido que muchas empresas han usado y seguirán usando cinta magnética en sus operaciones, entre las que destacan las hiperescalas (empresas que han crecido tanto que ofrecen sus propias infraestructuras o tienen datos masivos como resultado de su infraestructura) siempre necesitan muchas formas diferentes de tecnología para manejar la variedad de datos que ingresan a sus sistemas para alimentar una gama de servicios. Entre otras destacan: Bancos, Gobiernos, Milicia y organizaciones como CERN, así como corporaciones como Amazon, Google, Meta, Baidu, Alibaba y Tencent.

Tarjeta Gráfica La tarjeta gráfica o tarjeta de vídeo es un componente que viene integrado en la placa base de la computadora o se instala aparte

para ampliar sus capacidades. Concretamente, esta tarjeta está dedicada al procesamiento de datos relacionados con el vídeo y las imágenes que se están reproduciendo en la computadora.

Procesador Gráfico GPU el corazón de la tarjeta gráfica es la GPU o Unidad de procesamiento gráfico, un circuito muy complejo que integra varios miles de millones de transistores diminutos y puede tener desde uno a miles de núcleos (ya es común encontrar computadoras personales con tarjeta de 10496 cores y 24 GB de GRAM) que tienen capacidad de procesamiento independiente. De la cantidad y capacidad de estos núcleos dependerá la potencia.

Así como los procesadores centrales de las CPU, están diseñados con pocos núcleos pero altas frecuencias de reloj, las GPU tienden al concepto opuesto, contando con grandes cantidades de núcleos con frecuencias de reloj relativamente bajas. Luego tienes la memoria gráfica de acceso aleatorio o GRAM, que son Chips de memoria que almacenan y transportan información entre sí. Esta memoria no es algo que vaya a determinar de forma importante el rendimiento máximo de una tarjeta gráfica, aunque si no es suficiente puede acabar lastrando y limitando la potencia de la CPU.

La idea de usar esa potencia para otros menesteres se denomina GPGPU (General Purpose Computation on Graphics Processing Units) o GPU Computing. En el momento en el que las tarjetas gráficas permiten que se programen funciones sobre su Hardware se empieza a hacer uso de GPGPU. Al principio era necesario utilizar los lenguajes enfocados a la visualización en pantalla (como OpenGL) para realizar otros cálculos no relacionados con los gráficos. Esto implicaba el uso de funciones muy poco flexibles, originalmente diseñadas para otros fines, lo que hacía que la labor de programar para tal fin fuese realmente tediosa y complicada.

Para facilitar el empleo de las tarjetas gráficas para cualquier uso no vinculado con los gráficos, NVidia desarrolló toda una tecnología alrededor de la tarjeta, que permitía usar la misma para cualquier tarea: CUDA. ATI, la principal (y actualmente casi única) competidora, un poco más tarde haría lo propio lanzando su propia tecnología: Stream. En un principio las tarjetas gráficas solo trabajaban con aritmética de 32 bits, pero en la actualidad ya se cuenta con aritmética de 64 bit (para lograr esto, muchas tarjetas usan dos de sus cores de 32 bits para emular uno de 64 bits, reduciendo su número de cores útiles a la mitad).

Procesador Gráfico Integrado muchos procesadores CPU incorporan una o más GPU en su interior, llamada gráfica integrada (iGPU Integrated Graphics Processing Units o APU Accelerated Processing Unit). Generalmente es muy poco potente, pero lo suficiente para realizar tareas básicas como navegar por Internet, ver vídeos, e incluso para algunos juegos básicos, especialmente en las últimas generaciones puesto que cada vez son más potentes. No obstante, en las últimas generaciones de procesadores cada vez se están introduciendo gráficos integrados más potentes, y ya son capaces de manejar varios monitores, resoluciones 4K e incluso son capaces de mover algunos juegos a una tasa digna de FPS.

Tarjeta Gráfica por ejemplo, la tarjeta gráfica de AMD Instinct MI200 cuenta con más de 200,000 cores y 128 GB de HBM2, NVidia GEFORCE RTX 3090 proporciona 10,496 cores y 24 GB de GDDR6x, NVidia A100 cuenta con 80 GB de memoria HBM2 6192 cores y 432 núcleos tensor[13], mientras que la tarjeta NVidia Titan RTX proporciona 130 Tensor TFLOP de rendimiento, 576 núcleos tensores y 24 GB de memoria GDDR6.

Por otra parte, Intel ha desarrollado una aceleradora gráfica Artic Sound-M pensada para centro de datos (especialmente diseñada para juegos en la nube) que utiliza una GPU DG2 Xe-HPG que viene con una configuración de 512 unidades de ejecución lo que equivale a 4,096 Shaders, por ejemplo, esta aceleradora puede manejar hasta 8 Streamings simultáneos de video 4K o más de 30 si el vídeo es en 1080p y cuenta con más de 60 funciones virtualizadas.

En agosto del 2021, se anunció la construcción de la supercomputadora Polaris, acelerado por 2240 GPU NVIDIA A100 Tensor Core, el sistema puede alcanzar casi 1.4 exaflops de rendimiento teórico de IA y aproximadamente 44 petaflops de rendimiento máximo de doble precisión. Polaris, que será construido por Hewlett Packard Enterprise, combinará simulación y aprendizaje automático al abordar cargas de trabajo informáticas de alto rendimiento de inteligencia artificial y con uso intensivo de datos, impulsadas por 560 nodos en total, cada uno con cuatro GPU NVIDIA A100.

Tipos de Redes Según el Medio Físico Si bien, nuestros dispositivos de cómputo pueden funcionar sin conexión de red, estos se ven inmediatamente

[13] Un Tensor core (o núcleos Tensor) calculan la operación de una matriz 4x4 completa, la cual se calcula por reloj. Estos núcleos pueden multiplicar dos matrices FP16 4x4 y sumar la matriz FP32 al acumulador.

limitados. La red nos permite conectarnos a Internet que es el camino por el cual nos conectamos con el mundo. Las formas de conectar nuestros equipos a Internet en un principio fue exclusivamente por red alámbrica, desde ya hace unos años a la fecha se dispone de conexión a red alámbrica e inalámbrica, pero actualmente nuestros dispositivos cuentan casi exclusivamente con conexión inalámbrica.

¿Qué es el ancho de banda? Se trata de la capacidad máxima y la cantidad de datos que se pueden transmitir a través de una conexión (de internet, por ejemplo), en un momento determinado. Algo que debemos tener claro es que el ancho de banda de red es fundamental para la calidad y velocidad de la conexión.

El ancho de banda se mide en *bit/s* o en sus múltiplos *k/bits* o *m/bits* por segundo. Y para la mayoría de los casos, debemos asegurarnos siempre de tener el mayor ancho de banda que nos sea posible, porque de esta manera podremos tener una mejor y más rápida transferencia de datos.

¿Qué es entonces la velocidad de transmisión? Este término se puede definir como la velocidad a la que se transmite la información. Cuando un usuario adquiere un paquete con una empresa prestadora de servicios de internet, recibe, por ejemplo, 10 mbps, 30 mbps, 100 mbps, etc. Y esto se refiere a la cantidad de datos que podemos descargar o subir a la red. Como recomendación, lo indicado es que para que la velocidad pueda existir será necesario tener un ancho de banda igual o superior a la velocidad contratada en el paquete de servicio.

Normalmente vemos en los anuncios de todos los operadores, que estos ofrecen una cantidad cualquiera de megas de navegación; este valor numérico corresponde a la velocidad de descarga únicamente. Para encontrar la velocidad de subida, es necesario acceder a un test que nos revele cuál es el resultado y si lo que nos prometen, es verdad o no.

¿Qué es la latencia? Es el tiempo total que transcurre desde que enviamos una información, hasta que la misma llega a un receptor. Su valor de medición se hace en milisegundos, también se conoce como *Ping* y está presente en actividades que realiza cotidianamente como jugar en línea o hacer videollamadas.

¿Altera la latencia la velocidad de la conexión? La velocidad de conexión influye en la latencia una vez que pasamos de un rango predeterminado. Poniéndolo en un ejemplo, si tenemos una conexión a Internet de 1 Mbps y la comparamos con una conexión de 100 Mbps, dependiendo del tamaño del paquete se notará una mejora grande en la velocidad. Otros factores que importan a la hora de hablar de latencia son el estar conectado a internet por Wi-fi o un cable, si tiene servicio de fibra óptica o qué tanta distancia hay entre su ordenador y un Router, etc.

La latencia en la conexión también es la suma de otros retardos:

- De procesamiento: se define en el tiempo que tardan los Routers en examinar la cabecera y a su vez, la respuesta en determinar a dónde hay que enviar cualquier paquete haciendo una previa comprobación de sus tablas de enrutamiento.
- De cola: tiempo de espera del paquete para poder ser transmitido a través de un enlace físico. Cabe anotar que no podemos saber previamente si va a haber un retardo de cola o no, ya que este cambia en tiempo real.
- De transmisión: es el tiempo que tarda el paquete en arribar hasta el siguiente nodo o destino final.
- De propagación: es el tiempo que tarda un bit en propagarse desde un punto cualquiera de origen hasta llegar a uno de destino. Su velocidad depende del medio físico por el que se transporte.

El retardo total es la suma de todos los retardos anteriormente enunciados.

Comúnmente se asocia la latencia con la banda de ancha, pero existe una diferencia sustancial entre ambas. Si bien ambas afectan la velocidad de la conexión, la banda ancha permite que se pueda transmitir una gran cantidad de datos, mientras la latencia determina a qué velocidad se transmite esa cantidad de datos.

Si bien, tener red nos permite estar conectados, tenemos una gran limitación por las velocidades de conexión a las que tendremos acceso según el tipo de medio físico que usemos para conectarnos, así como el número de dispositivos con los que compartamos la conexión. Las redes inalámbricas parecen ser omnipresentes, pero las velocidades de interconexión dejan mucho que desear como veremos a continuación.

Redes alámbricas se comunica a través de cables de datos (generalmente basada en Ethernet). Los cables de datos, conocidos como cables de red de Ethernet o cables con hilos conductores (CAT5), conectan computadoras y otros dispositivos que forman las redes. Las redes alámbricas son mejores cuando se necesita mover grandes cantidades de datos a altas velocidades, como medios multimedia de calidad profesional.

Ventajas

- Costos relativamente bajos
- Ofrece el máximo rendimiento posible
- Mayor velocidad - cable de Ethernet estándar hasta 1 Gbps[14]
- Mayor rendimiento de Voz sobre IP.
- Mejores estándares Ethernet en la industria.
- Mayor capacidad de ancho de banda por cables.
- Aplicaciones que utilizan un ancho de banda continúo.

Desventajas

- El costo de instalación siempre ha sido un problema común en este tipo de tecnología, ya que el estudio de instalación, canaletas, conectores, cables y otros suman costos muy elevados en algunas ocasiones.
- El acceso físico es uno de los problemas más comunes dentro de las redes alámbricas. Ya que para llegar a ciertos lugares, es muy complicado el paso de los cables a través de las paredes de concreto u otros obstáculos.
- Dificultad y expectativas de expansión es otro de los problemas más comunes, ya que cuando pensamos tener un número definido de nodos en una oficina, la mayoría del tiempo hay necesidades de construir uno nuevo y ya no tenemos espacio en los Switches instalados.

Hoy en día se puede hacer la siguiente clasificación de las redes de protocolo Ethernet para cable y fibra óptica:

[14] El término bps se refiere a transmitir Bits por segundo (se requieren 8 Bits para formar un Byte). Por eso el término de 1,000 Mbps es equivalente a 125 MB/s.

- Ethernet, que alcanza no más de 10 Mbps de velocidad
- Fast Ethernet, que puede trabajar con hasta 100 Mbps
- Gigabit Ethernet, alcanza hasta 1 Gbps (1000 Mbps aprox.)
- 2.5 y 5 Gigabit Ethernet hasta 2.5 Gbps si usamos cableado Cat 5a y nada menos que 5 Gbps con cableado Cat 6
- 10 Gigabit Ethernet, que puede alcanzar hasta los 10 GMbps
- 1000GbE para alcanzar la Ethernet Terabit (125 Gbytes)
- El récord mundial en mayo del 2022 de transmisión en fibra óptica es de 1.02 petabits por segundo enviados a través de 51.7 kilómetros (que podría transmitir hasta 10 millones de canales por segundo de vídeo a resolución 8K), que rompe al récord anterior de 319 terabits por segundo sobre una distancia de 1,800 millas (con el estimado de descarga de 80,000 películas simultáneamente en un segundo).

La categoría del cable o el tipo de fibra óptica determina la velocidad máxima soportada por cada tipo de cable, pero aunque haya cables con la misma velocidad hay otros factores que determinan su usabilidad, como el ancho de banda o la frecuencia. La frecuencia o ancho de banda determina la potencia de la red, a mayor frecuencia mayor ancho de banda y menor perdida de datos. Este factor es importante si vamos a conectar varios equipos al mismo cable de red o vamos a hacer una gran tirada de cable, ya que cuanto más largo sea el cable de red más potencia perderá. Siempre tendrá más velocidad un cable corto que un cable largo, pero si el ancho de banda es amplio tardará más metros en perder potencia y velocidad.

Ethernet es el estándar que domina la gran mayoría de mercado, presente de manera casi exclusiva tanto en mercado doméstico, como en pequeña y mediana empresa representa también un porcentaje muy significativo en grandes centros de datos. Pero existen otros protocolos que se pueden situar en el mismo nivel de calidad que Gigabit Ethernet como InfiniBand.

InfiniBand es un bus serie bidireccional de comunicaciones de alta velocidad en las que las rápidas comunicaciones entre servidores son críticas para el rendimiento, llegando a ofrecer velocidades de hasta 2.0 Gbps netos en cada dirección del enlace en un nodo simple, 4 Gbps netos en un nodo doble y hasta 8 Gbps netos en un nodo quadruple. Estos nodos a su vez se pueden

agrupar en grupos de 4 ó 12 enlaces llegando a velocidades de hasta 96 Gbps netos en un grupo de 12 nodos cuádruples. El factor de velocidad neta viene relacionado con que Infiniband de cada 10 bits que transmite 8 de ellos son datos, basándose en la codificación 8B/10B.

Recientemente se han implementado sistemas en los que ya no se utiliza esta codificación 8B/10B sino la 64B/66B que permite mejorar el porcentaje de datos útiles por trama enviada y que ha permitido los nodos FDR-10 (Fourteen Data Rate-10 a 10 Gbps), FDR (Fourteen Data Rate a 13.64 Gbps) y EDR (Enhanced Data Rate a 25 Gbps). Este último en un grupo de 12 nodos proporciona hasta 300 Gbps. Los últimos desarrollos de Gigabit Ethernet, proporcionan hasta 100 Gbps por puerto.

Estas enormes velocidades de conexión hacen que Infiniband sea una conexión con una muy importante presencia en superordenadores y clústers, por ejemplo del top 500 de superordenadores en 2020, 226 están conectados internamente con Infiniband, 188 lo están con Gigabig Ethernet y el resto con Myrinet, Cray, Fat Tree u otras interconexiones a medida.

Una de las principales ventajas de Infiniband sobre Ethernet es su bajísima latencia, por ejemplo y basándonos en los datos del estudio de Qlogic "Introduction to Ethernet Latency, an explanation to Latency and Latency measurement", la latencia en 10 Gpbs Ethernet se sitúa en 5 microsegundos mientras que la de Infiniband se sitúa por debajo de los 3 microsegundos.

Los sistemas de conmutadores inteligentes InfiniBand de NVIDIA Mellanox ofrecen el mayor rendimiento y densidad de puertos para computación de alto rendimiento (HPC), IA, Web 2.0, Big Data, nubes y centros de datos empresariales. La compatibilidad con configuraciones de 36 a 800 puertos a hasta 200 Gpbs por puerto permite que los clústeres de cómputo y los centros de datos convergentes funcionen a cualquier escala, lo que reduce los costos operativos y la complejidad de la infraestructura.

Redes inalámbricas: es una red en la que dos o más terminales (ordenadores, tabletas, teléfonos inteligentes, etc.) se pueden comunicar sin la necesidad de una conexión por cable. Se basan en un enlace que utiliza ondas electromagnéticas (radio e infrarrojo) en lugar de cableado estándar. Permiten que los dispositivos remotos se conecten sin dificultad, ya se encuentren a unos metros de distancia como a varios kilómetros.

Asimismo, la instalación de estas redes no requiere de ningún cambio significativo en la infraestructura existente como pasa con las redes cableadas.

Tampoco hay necesidad de agujerear las paredes para pasar cables ni de instalar porta cables o conectores. Esto ha hecho que el uso de esta tecnología se extienda con rapidez.

Tipos de redes inalámbricas

- LAN Inalámbrica: Red de área local inalámbrica. También puede ser una red de área metropolitana inalámbrica.
- GSM (Global System for Mobile Communications): la red GSM es utilizada mayormente por teléfonos celulares.
- D-AMPS (Digital Advanced Mobile Phone Service): está siendo reemplazada por el sistema GSM.
- Fixed Wireless Data: Es un tipo de red inalámbrica de datos que puede ser usada para conectar dos o más edificios juntos para extender o compartir el ancho de banda de una red sin que exista cableado físico entre los edificios.
- Wi-Fi[15]: es uno de los sistemas más utilizados para la creación de redes

[15] Notemos que una conexión de WiFi tradicional sin obstáculos con una intensidad de banda de 2.4 GHZ puede tener un alcance máximo de 46 metros y para la banda de 5.0 GHz un alcance de 15 metros. Paro hay muchos objetos que interfieren con la señal del Router del WiFi y el sitio en el que colocamos nuestro dispositivo inalámbrico como computadora o teléfono inteligente:

- Superficies y/o objetos de metal o vidrio blindado
- Refrigeradores, lavadoras y radiadores
- Hornos de microondas, cámaras Web, monitores de bebés y teléfonos inalámbricos
- Paredes y muros
- Dispositivos arquitectónicos que funciones como una jaula de Faraday (es un contenedor recubierto por materiales conductores de electricidad como mallas metálicas, papel aluminio, cajas o cestos de basura de acero que funciona como un blindaje contra los efectos de un campo eléctrico proveniente del exterior).

En caso de que varios dispositivos se conecten a la red, se puede optar por colocar el Router en un punto medio, para que ninguna zona quede sin cobertura. Por otra parte, para una mejor conexión, también procura que el Router se encuentre en una zona elevada, pues esto mejorará el alcance de la señal inalámbrica. Una excelente opción para mejorar la calidad del internet inalámbrico cuando hay obstáculos es utilizar un repetidor de WiFi, este dispositivo es muy útil para amplificar la señal y llevarla a más sitios de nuestra red.

inalámbricas en computadoras, permitiendo acceso a recursos remotos como internet e impresoras. Utiliza ondas de radio.

Ventajas

- La instalación de redes inalámbricas suele ser más económica.
- Su instalación también es más sencilla.
- Permiten gran alcance; las redes hogareñas inalámbricas suelen tener hasta 100 metros desde la base transmisora.
- Permite la conexión de gran cantidad de dispositivos móviles. En las redes cableadas mientras más dispositivos haya, más complicado será el entramado de cables.
- Posibilidad de conectar nodos a grandes distancias sin cableado, en el caso de las redes inalámbricas corporativas.
- Permiten más libertad en el movimiento de los nodos conectados, algo que puede convertirse en un verdadero problema en las redes cableadas.
- Permite crear una red en áreas complicadas donde, por ejemplo, resulta dificultoso o muy caro conectar cables.

Desventajas

- Calidad de Servicio: La velocidad que posee la red inalámbrica no supera la cableada, ya que esta puede llegar a los 10 Mbps, frente a 100 Mbps que puede alcanzar la cableada. Hay que tomar en cuenta la tasa de error debida a las interferencias.
- Costo: En algunos casos, puede ser más barato cablear una casa/oficina que colocar un servicio de red inalámbrica.
- La señal inalámbrica puede verse afectada e incluso interrumpida por objetos, árboles, paredes, espejos, entre otros.

La velocidad máxima de transmisión inalámbrica de la tecnología 802.11b es de 11 Mbps. Pero la velocidad típica es solo la mitad: entre 1.5 y 5 Mbps dependiendo de si se transmiten muchos archivos pequeños o unos pocos

archivos grandes. La velocidad máxima de la tecnología 802.11g es de 54 Mbps. Pero la velocidad típica de esta última tecnología es solo unas 3 veces más rápida que la de 802.11b: entre 5 y 15 Mbps. Resumiendo, las velocidades típicas de los diferentes tipos de red son:

Estándar	V.Máxima	V.Practica	Frecuencia	Ancho Banda	Alcance
802.11	2Mbit/s	1Mbit/s	2.4Ghz	22MHz	330 metros
802.11a(WiFi5)	54Mbit/s	22Mbit/s	5.4Ghz	20MHz	390 metros
802.11b	11Mbit/s	6Mbit/s	2.4Ghz	22MHz	460 metros
802.11g	54Mbit/s	22Mbit/s	2.4Ghz	20MHz	460 metros
802.11n	600Mbit/s	100Mbit/s	2.4Ghzy5.4Ghz	20 y 40MHz	820 metros
802.11ac	6.93Gbps	100Mbit/s	5.4Ghz	80 o hasta 160MHz	Poco alcance, pero sin interferencias
802.11ad	7.13Gbit/s	Hasta6Gbit/s	60Ghz	2MHz	300 metros
802.11ah	35.6Mbps	26.7Mbps	0.9Ghz	2MHz	1000 metros
802.11ax(WiFi6)	9,6Gbps	6,9Gbps	2.4Ghz y 5.4Ghz	20MHz	1000 metros

Como puedes ver, los principales factores que influyen en la calidad de una conexión WiFi, son la frecuencia, el ancho de banda y el alcance total. Considerando que, todo esto junto con la velocidad máxima y la velocidad práctica, se congregan en lo que es cada versión de este tipo de conexión a la red. Además, la conexión se degradará inexorablemente con la cantidad de dispositivos conectados y su consumo de datos.

Velocidad de los Proveedores cuando se contrata el servicio de internet, la velocidad de interconexión del mismo depende de cuánto sea el cobro, pero en la mayoría de los casos se tendrá una velocidad de descarga mayor a la velocidad de carga y nuestra red será una intranet (dirección de IP dinámica) compartida con otros miles de usuarios abonados al servicio de internet del proveedor. También es posible contratar una interconexión con velocidades homogéneas para carga y descarga dedicada, el costo del mismo se puede hasta triplicar con respecto a uno no homogéneo.

En caso de requerir una dirección de internet homologada o pública, el costo de contratar el servicio aumenta considerablemente, pero de esta forma nuestros equipos son visibles en el Internet (esto conlleva un aumento de riesgos al estar nuestros equipos más vulnerables a ataques informáticos).

1.2 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias

Elegir la base de datos más adecuada para un proyecto supone enfrentarse a todo tipo de posibilidades de elección. Quizá la primera, y la más importante, es elegir una base de datos de código abierto o una propietaria. Es decir, una de código abierto, generalmente gratuita, o una propiedad de una empresa. A la vista de esto, la primera ventaja de las bases de datos de código abierto salta a la vista: evitan tener que hacer frente a un desembolso económico fuerte, puesto que no hace falta comprar su licencia a un fabricante. A no ser que tengas necesidades muy concretas y específicas, pero la proliferación de bases de datos de código abierto en los últimos años, así como de diversos complementos para ellas, hace que resulten cada vez más atractivas incluso para fines muy concretos.

Mientras tanto, la popularidad de los sistemas de bases de datos con licencia comercial, es decir, de las propietarias, ha ido cayendo a lo largo de los últimos años. Y según la lista de sistemas de gestión de bases de datos de DB-Engines, en muchos casos han sido superadas por las bases de datos de código abierto en lo que respecta a la adopción en empresas.

Ventajas de las Bases de Datos de Código Abierto Una base de datos de código abierto es un sistema de gestión de bases de datos abierto y de uso gratuito, y quizá una de las pocas desventajas que tiene frente a las bases de datos propietarias es que no cuenta con un servicio de soporte técnico por parte de un proveedor, algo con lo que sí cuentan las bases de datos propietarias. Pero precisamente de esta desventaja nace una de sus ventajas.

Las bases de datos de código abierto cuentan con una nutrida comunidad de usuarios que, en muchos casos no solo comparten sus conocimientos en foros y grupos de usuarios especializados, sino que en muchos casos están dispuestos a ayudar a otros usuarios con problemas y necesidades concretas con ellas.

Además, la información que se guarda en una base de datos de código abierto es de quien la inserta en ella. Sin ningún tipo de restricción. También puedes modificarla para adaptarla a tus necesidades, aunque para ello necesitas contar con ciertos conocimientos de desarrollo. Puedes crear aplicaciones y complementos con base en el Software de código abierto que elijas, de una manera muy parecida a como sucede por ejemplo con el CMS Word-

Press. Por lo tanto, las bases de datos de código abierto te ofrecen un sinfín de posibilidades de personalización y ampliación.

Las Principales Bases de Datos de Código Abierto El número de bases de datos de código abierto crece cada vez más a medida que pasan los años. No obstante, entre todas las posibilidades, hay varias que, según TechTarget, o bien por su sencillez y robustez, o bien por sus posibilidades y facilidades de ampliación y personalización, son las que consiguen una mayor adopción y despliegue en las empresas. Entre ellas están las seis siguientes, quizá las más populares y utilizadas.

MySQL Sin duda esta es la más extendida de las bases de datos de código abierto, y durante muchos años, la opción por defecto para muchos desarrolladores a la hora de optar por una base de datos. Oracle se hizo con ella hace ya bastantes años cuando compró Sun Microsystems, y es un sistema de gestión de bases de datos relacional. Esto quiere decir que utiliza tablas para almacenar los datos, y los tipos de datos almacenados deben estar relacionados entre sí de alguna manera. Cuenta con una edición de código abierto y algunas versiones de pago, y debido a su popularidad hay muchas herramientas de terceros para ella, así como una cantidad importante de documentación centrada en facilitar su aprendizaje.

La base de MySQL se originó a partir de un sistema de base de datos propietario antiguo, mantener la mayor parte de su compatibilidad y convertir los resultados en código abierto. Sus creadores se centraron en la velocidad, y en la actualidad sigue estando considerada entre las opciones más rápidas de bases de datos. Además, de la velocidad, cuenta con un Script que contribuye a la mejora de la seguridad de las bases de datos de los usuarios. Esta base de datos es compatible, entre otros, con los lenguajes de programación C, C++, Java, Python y Ruby.

No obstante, se trata de una base de datos con licencia dual. Esto implica que algunas de sus funciones y Plugins solo son accesibles en ediciones propietarias de pago. En cuanto a su curva de aprendizaje, no es muy elevada. No es necesario saber mucho SQL para utilizarla, y puedes trabajar con la base de datos desde la línea de comandos. MySQL tiene una compatibilidad elevada prácticamente con cualquier sistema operativo. Se trata de una base de datos sólida, rápida y versátil, por lo que es adecuada para multitud de casos de uso.

MariaDB Cuando Oracle compró Sun, y con ella MySQL, hubo quien no se lo tomó muy bien, porque aunque Oracle ha mantenido MySQL como código abierto, la compañía no tiene mucha reputación que digamos como defensora del Software abierto. Uno de los que no se creyó a Oracle fue uno de los propios fundadores de MySQL, que dejó la compañía para crear MariaDB, un fork de MySQL con una compatibilidad casi completa con ella.

Se puede utilizar como un sustituto prácticamente para todo de MySQL, aunque cuenta con varias funciones únicas que la distinguen de ella. Para empezar, MariaDB usa el motor de almacenamiento Aria para gestionas querys de SQL complejas. Esto hace que la base de datos sea todavía más veloz que MySQL. Además, puedes utilizar filas dinámicas como columnas de tabla, lo que la hace más flexible y adaptable a diversas situaciones.

Además, cuenta con motores de almacenamiento especializados para casos de uso concretos que no tiene MySQL. Por ejemplo, permite implementar el almacenamiento distribuido o las transacciones distribuidas. Como hemos mencionado, puedes utilizar MariaDB en los mismos supuestos y casos que MySQL, pero no hay compatibilidad entre las dos, por lo que tendrás que elegir cuál de las dos quieres utilizar.

PostgreSQL Esta base de datos relacional y de código abierto es muy utilizada en ciencia de datos y creación de gráficos, y también en el sector de la Inteligencia Artificial. Esto último se debe a que PostgreSQL está especialmente pensada para aplicaciones de Python y Ruby. No obstante, también es compatible con PHP. Se ha desarrollado pensando en la elegancia, y cuenta con ciertas funciones bastante diferenciadas. Entre ellas están la posibilidad de implementar la replicación asíncrona, y el soporte nativo para el almacenamiento de documentos de estilo JSON o XML.

Además, PostgreSQL permite hacer búsquedas completas de texto en la base de datos, y cuenta con varios tipos de datos integrados de gran valor para algunas aplicaciones, como los rangos, los arrays y la geolocalización. Eso sí, no es excesivamente adecuada para aplicaciones que requieran operaciones intensivas de lectura, y la creación de informes de datos con regularidad puede hacer que el almacenamiento de documentos no esté muy fino si tiene que almacenar un conjunto de datos muy grande.

SQLite También relacional, SQLite es una base de datos de código abierto compuesta por una librería ligera y pequeña que ofrece un motor de

base de datos. A pesar de su ligereza, permite generar bases de datos de cientos de Terabytes, y un tamaño máximo de filas de un Gigabyte. Incluso con archivos de este tamaño, SQLite sigue siendo rápida. Además, es compatible con Java, Python, C o C++, entre otros lenguajes.

Cuenta con decenas de casos de uso, y por ejemplo, la encontrarán muy útil los desarrolladores de aplicaciones sencillas. Está integrada en smartphones y en muchos ordenadores. También es muy valorada en aplicaciones de Internet de las Cosas y Webs de poco tráfico. Eso si, por su estructura, no está indicada para sitios con mucho tráfico, ya que su rendimiento se resiente. Además, cuenta con ciertas limitaciones que en algunos casos pueden ser importantes.

Estas hacen que, entre otras cosas, no esté indicada en aplicaciones a las que tienen que acceder varios usuarios con permisos de acceso especiales. SQLite lee y escribe en un archivo de disco común y corriente, por lo que los únicos permisos de acceso aplicables para esta base de datos son los incluidos habitualmente en el sistema operativo.

MongoDB Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades. Si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Redis Esta base de datos de código abierto es distinta a las más utilizadas, ya que se utiliza sobre todo en situaciones completamente distintas a ellas. Se trata de una base de datos que permite estructurar datos como

pares de valor clave. Por tanto, se trata de un sistema parecido a los arrays asociativos de PHP o los diccionarios de Python. En realidad, Redis es un sistema para enlazar datos para su referencia, más adelante, con rapidez. Está considerada como la mejor elección como base de datos para cacheado, y también para el trabajo con datos distribuidos.

Para empezar, es una solución de almacenamiento en memoria, enteramente en RAM. Esto quiere decir que sus velocidades de lectura y escritura son muy rápidas. Para aprender sus bases tampoco necesitas mucho tiempo, ya que puedes conocerlas en minutos y empezar con rapidez con el almacenamiento de objetos en ella. Eso sí, puede que no sea muy útil para aplicaciones complejas, pero basta con utilizarla junto con otras bases de datos, como MariaDB, para el resto de la aplicación que se esté desarrollando.

1.3 Agradecimientos

Este texto es una recopilación de múltiples fuentes, nuestra aportación -si es que podemos llamarla así- es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
http://132.248.181.216/Herramientas/

Además, la documentación y los diferentes ejemplos que se presentan en este trabajo, se encuentran disponibles en dicha liga, para que puedan ser copiados desde el navegador y ser usados. En aras de que el interesado pueda correr dichos ejemplos y afianzar sus conocimientos, además de que puedan ser usados en diferentes ámbitos a los presentados aquí.

2 Sistemas Operativos

Actualmente tenemos 3 grandes sistemas operativos en el mercado[16]:

- Windows
- Unix
- GNU[17]/Linux

De los cuales, sus dignos representantes son: Windows, macOS, iOS, Android, Chrome OS y GNU/Linux con todas sus diferentes distribuciones[18]. Y sin temor a equivocarnos aseguramos que Android es la distribución de GNU/Linux más popular e iOS es el más popular de los UNIX.

¿Qué es un Sistema Operativo? El conjunto de programas informáticos que permiten la administración eficaz de los recursos de una computadora es conocido como sistema operativo o Software de sistema. Estos programas comienzan a trabajar apenas se enciende el equipo, ya que gestionan el Hardware desde los niveles más básicos y permiten además la interacción

[16] Cuotas de mercado de diferentes sistemas operativos:

https://gs.statcounter.com/os-market-share/desktop/worldwide
https://netmarketshare.com

[17] GNU -es un acrónimo recursivo de «GNU no es UNIX»- es un sistema operativo de Software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU además de Software libre publicado por terceras partes con distintas licencias que conforman una distribución.

[18] Una distribución de Linux es un sistema operativo compuesto por el Kernel de Linux, herramientas GNU, Software adicional y un administrador de paquetes. También puede incluir un servidor de pantalla y un entorno de escritorio que se utilizarán como sistema operativo de escritorio normal. El término es distribución de Linux (o distribución en forma abreviada) porque una entidad como Debian o Ubuntu 'distribuye' el Kernel de Linux junto con todo el Software y las utilidades consideradas por cada entidad como necesarias (como administrador de red, administrador de paquetes, entornos de escritorio, etc.) para que pueda ser utilizado como sistema operativo. Sus distribuciones también asumen la responsabilidad de proporcionar actualizaciones para mantener el Kernel y otras utilidades.

Entonces, Linux es el Kernel, mientras que la distribución de Linux es el sistema operativo. Esta es la razón por la que también se les conoce como sistemas operativos basados en Linux (hay otros Kernels como son FreeBSD, NetBSD y Hurd).

con el usuario. Cabe destacar que los sistemas operativos no funcionan sólo en las computadoras. Por el contrario, este tipo de sistemas se encuentran en la mayoría de los dispositivos electrónicos que utilizan microprocesadores: el Software de sistema posibilita que el dispositivo cumpla con sus funciones -por ejemplo, un teléfono móvil o un reproductor de DVD-.

El sistema operativo cumple con cinco funciones básicas:

- Proporciona la interfaz del usuario -gráfica o de texto-
- La administración de recursos
- La administración de archivos
- La administración de tareas
- El servicio de soporte y utilidades

En cuanto a la interfaz del usuario, el sistema se encarga de que el usuario pueda ejecutar programas, acceder a archivos y realizar otras tareas con la computadora. La administración de recursos permite el control del Hardware, incluyendo los periféricos y la red. El Software de sistema también se encarga de la gestión de archivos, al controlar la creación, la eliminación y el acceso a los mismos, así también, de la administración de las tareas informáticas que ejecutan los usuarios finales. Por último, podemos mencionar que el servicio de soporte se encarga de actualizar las versiones, mejorar la seguridad del sistema, agregar nuevas utilidades, controlar los nuevos periféricos que se agregan a la computadora y corregir los errores del Software.

Tipos de Sistemas Operativos en Función de la Administración de las Tareas Podemos distinguir dos clases de sistemas operativos en función de cómo administran sus tareas, pueden ser:

Sistemas Operativos Monotarea: son sistemas operativos que únicamente cuentan con la capacidad para realizar una tarea al mismo tiempo. Son los sistemas más antiguos, que también llevan aparejados un CPU de menor capacidad. En estos casos, si el equipo está imprimiendo, no atenderá a las nuevas órdenes, ni será capaz de iniciar un nuevo proceso hasta que el anterior haya finalizado.

Sistemas Operativos Multitarea: son los sistemas operativos más modernos, con capacidad para el procesamiento de varias tareas al mismo tiempo. Cuentan con la capacidad para ejecutar varios procesos en uno o más procesadores, por lo que existe la posibilidad de que sean utilizados por varios usuarios al mismo tiempo, y podrían aceptar múltiples conexiones a través de sesiones remotas.

Tipos de Sistemas Operativos en Función de la Administración de los Usuarios También es posible realizar una división de los sistemas operativos en función de la forma en la que se administran los usuarios, como vemos a continuación:

Sistema de Administración Monousuario: sólo pueden gestionar un usuario al mismo tiempo. Así, a pesar de que varios usuarios pueden tener acceso al sistema, solo un usuario puede acceder para realizar y ejecutar operaciones y programas.

Sistemas de Administración Multiusuario: se refiere a todos aquellos sistemas operativos que permiten el empleo de sus procesamientos y servicios al mismo tiempo. Así, el sistema operativo cuenta con la capacidad de satisfacer las necesidades de varios usuarios al mismo tiempo, siendo capaz de gestionar y compartir sus recursos en función del número de usuarios que estén conectados a la vez.

¿Qué Sistema Operativo Usar? ¿Mac o Microsoft? ¿Windows o Linux? ¿Android o iOS? Son preguntas frecuentes que todos nos hemos hecho alguna vez, y es que elegir un sistema operativo, una computadora o un dispositivo móvil no es tan simple. O al menos no lo era años atrás. En la actualidad las diferencias entre sistemas operativos de escritorio son cada vez menos, hasta el punto que prácticamente cualquier servicio Online es compatible con Windows, Mac y GNU/Linux y las principales firmas de Software crean aplicaciones para las tres plataformas principales, salvo excepciones. Lo mismo empieza a ocurrir con el Hardware.

Poco tendremos que decir del sistema operativo de Apple, Mac o iOS (ambos son derivados de Darwin BSD que es un sistema operativo tipo UNIX), ya que son los sistemas operativos más bonitos y que mejores resultados han dado a todos los usuarios que los han probado. Mac es un sistema pensado

para los profesionales de los sectores que necesitan de un equipo de cómputo que sea capaz de todo, como los desarrolladores, programadores, diseñadores, periodistas, fotógrafos, músicos, DJ's y muchos más empleos que se benefician de este sistema operativo.

Después tenemos a Windows, un sistema operativo versátil pensado sobre todo para un uso doméstico, aunque eso no quita que muchas empresas utilicen Windows en sus equipos de cómputo ya que es un sistema operativo que puede dar muy buenos resultados en este aspecto.

Sin embargo, llegamos a Linux, el gran desconocido por muchos. Un sistema operativo mucho más versátil que Windows y que puede ser igual o más profesional que Mac. Sin embargo, la ventaja que tienen estos dos sistemas operativos, es que vienen ya preparados y configurados para el tipo de mercado al que van dirigidos, pero GNU/Linux no.

Esto es una ventaja y una desventaja al mismo tiempo, ya que si tenemos práctica, podemos hacer que el sistema operativo se adapte a nuestras necesidades sin problemas, pero si no tienes práctica, puede que sea demasiado lo que tienes que configurar.

Cuota de Mercado para los Sistemas Operativos Agosto es uno de los meses en los que miles de compañías analizan el tráfico que les llega de usuarios a sus páginas Web y desde que plataformas llegan, según un informe de International Data Corporation (https://www.idc.com) y The Linux Foundation (https://www.linuxfundation.org) en el año 2024 tenemos:

- En el segmento de los sistemas operativos de escritorio basados en Linux ha subido su cuota de mercado llegando al 3%, esto no parecerá mucho, pero si nos fijamos bien, vemos que Mac tiene un 7.5% -basado en Unix-, Chrome OS -usa el Kernel de Linux- tiene 10.8% y Windows el resto.

- En el segmento de teléfonos inteligentes (SmartPhones) y tabletas basadas en Android -usa el Kernel de Linux- tiene 86 % , iOS tiene 13.9 % -basado en Unix- y menos del 1% el resto de los sistemas operativos.

- En el segmento de servidores se estima que más del 60% de los servidores a nivel mundial usan Linux, 1% usan Unix y el resto Windows. Pero en los principales servidores del mundo (un millón) 96 % usan Linux.

- El 90% de toda la infraestructura Cloud corre usando Linux. Es de destacar que en el servicio de servidores Microsoft Azure, el sistema predominante es Linux.

- En el segmento de supercomputadoras, Linux tiene la cuota más importante del mercado; es utilizado en los Top 500 sistemas de supercómputo de alto desempeño del mundo[19].

Hay que decir, que hoy en día y tal y como están las cosas, no existe un sistema operativo que sea definitivo. Así que la pregunta de si GNU/Linux[20] es mejor que Windows o Mac no tiene sentido, ya que cada sistema operativo tiene sus pros y sus contras.

Pero la disyuntiva sigue ahí. ¿Debemos usar Windows en nuestro equipo de cómputo?, ¿nos conviene pasarnos a Linux?. Hay razones a favor y en contra para todos los gustos.

Número de Líneas del Código Fuente de un Sistema Operativo
Pese a que existen múltiples variantes de cada sistema operativo, se han dado a conocer los números de líneas de código fuente que componen la vertiente más usada de algunos sistemas operativos:

- Microsoft Windows 3.1 (Abril de 1992): 3 millones de líneas ($200 USD en 1992)

- Microsoft Windows 95 (Agosto de 1995): 15 millones de líneas (Home $109.95 y Pro$ 209.95 USD en 1995)

- Microsoft Windows NT 4.0 (Julio 1996): 12 millones de líneas (5 usuarios $809, 10 usuarios $1,129 USD en 1996)

[19] Existe el Ranking de las 500 supercomputadoras más poderosas del mundo (esta se actualiza cada seis meses en junio y noviembre) y puede ser consultada en:
https://top500.org
La cuota de supercomputadoras con GNU/Linux ha sido de: 2012 (94%), 2013 (95%), 2014 (97%), 2015 (97.2%), 2016 (99.6%), 2017 (99.6%), 2018 (100%), 2019 (100%), 2020 (100%).

[20] Los resultados de GNU/Linux son muy satisfactorios para los desarrolladores y partícipes de la comunidad Linux, pero todavía hace falta mucho por hacer para que tenga una cuota significativa en el escritorio y esto sólo será posible si los distribuidores de equipo generan un esquema más agresivo para vender máquinas con Linux preinstalado.

- Microsoft Windows 2000 (Febrero 2000): 29 millones de líneas (Pro $319 USD en 2000)
- Microsoft Windows XP (Octubre 2001): 45 millones de líneas (Home $175 US y Pro $ 255 USD en 2001)
- Microsoft Windows Vista (Enero 2007): 50 millones de líneas (Home Premium $239.99 USD en 2007)
- Microsoft Windows 7 (Octubre 2009): 40 millones de líneas (Home Premium $199.99 USD en 2009)
- Microsoft Windows 8 (Octubre 2012): 60 millones de líneas (Home $119.99 US y Pro $ 199.99 USD en 2013)
- Microsoft Windows 10 (Julio 2015): 60 millones de líneas sin Cortana y 65 millones de líneas con Cortana ($139.99 USD en 2019)
- Sun Solaris (Octubre de 1998) 7.5 millones de líneas
- Red Hat Linux 6.2 (Marzo de 2000): 17 millones de líneas
- Red Hat Linux 7.1 (Abril de 2001): 30 millones líneas
- Red Hat Linux 8.0 (Septiembre de 2002): 50 millones de líneas
- Fedora Core 4 (Mayo de 2005): 76 millones de líneas
- Fedora 9 (Mayo del 2008): 205 millones de líneas
- Debian GNU/Linux 3.0 "Woody" (Julio de 2002): 105,000,000 líneas
- Debian GNU/Linux 3.1 "Sarge" (Junio de 2005); 229,500,000 líneas
- Debian GNU/Linux 7 "Wheezy" (Mayo 2013): 419 millones de líneas
- Debian GNU/Linux 10 "Buster" (Julio 2019): 1,077,110,982 líneas
- Debian GNU/Linux 11 "Bullseye" (Agosto 2021): 1,152,960,944 líneas
- Debian GNU/Linux 12 "bookworm" (Junio 2023): 1,341,564,204 líneas
- Kernel Linux 0.01 (Septiembre 1991): 8,413 líneas

- Kernel Linux 1.0 (Marzo 1994): 176,250 líneas
- Kernel Linux 2.6 (Diciembre 2003): 5,475,685 líneas
- Kernel Linux 4.12 (Julio 2017): 24 millones líneas
- Kernel Linux 5.8 (Agosto 2020): 28 millones líneas
- Kernel Linux 5.14 (Julio 2021): 29.7 millones de líneas

El Kernel o Núcleo Es un componente fundamental de cualquier sistema operativo. Es el encargado de que el Software y el Hardware de cualquier equipo de cómputo puedan trabajar juntos en un mismo sistema, para lo cual administra la memoria de los programas y procesos ejecutados, el tiempo de procesador que utilizan los programas, o se encarga de permitir el acceso y el correcto funcionamiento de periféricos y otros elementos físicos del equipo.

Kernel de Linux el núcleo del sistema operativo Linux/Unix (llamado Kernel) es un programa escrito casi en su totalidad en lenguaje C, con excepción de una parte del manejo de interrupciones, expresada en el lenguaje ensamblador del procesador en el que opera, el Kernel reside permanentemente en memoria y alguna parte de él está ejecutándose en todo momento. Pero es muy común confundir al Kernel de Linux con una distribución como Debian y Ubuntu, Linux solo es un núcleo (hay otros como son FreeBSD, NetBSD y Hurd).

Durante mucho tiempo el núcleo Linux solo funcionaba en la serie de máquinas x86 de Intel, desde el 386 en adelante. Sin embargo, hoy día esto ya no es cierto. El núcleo Linux se ha adaptado a una larga y creciente lista de arquitecturas. Siguiendo esos pasos, la distribución Debian GNU/Linux se ha adaptado a estas plataformas. En general este proceso tiene un comienzo difícil (hay que conseguir que la *libc* y el enlazador dinámico funcionen sin trabas), luego sigue un trabajo relativamente largo y rutinario, de conseguir recompilar todos los paquetes bajo las nuevas arquitecturas.

Debian GNU/Linux es un sistema operativo, no un núcleo (en realidad es más que un SO, ya que incluye miles de aplicaciones). Para probar esta afirmación, aun cuando la mayor parte de adaptaciones se hacen sobre núcleos Linux, también existen adaptaciones basadas en los núcleos FreeBSD, NetBSD y Hurd.

Linux es multiprogramado, dispone de memoria virtual, gestión de memoria, conectividad en red y permite bibliotecas compartidas. Linux es multiplataforma y es portable a cualquier arquitectura siempre y cuando está disponga de una versión de GCC compatible.

La parte de un sistema operativo que se ejecuta sin privilegios o en espacio de usuario es la biblioteca del lenguaje C, que provee el entorno de tiempo de ejecución, y una serie de programas o herramientas que permiten la administración y uso del núcleo y proveer servicios al resto de programas en espacio de usuario, formando junto con el núcleo el sistema operativo.

En un sistema con núcleo monolítico como Linux la biblioteca de lenguaje C (*libc*) consiste en una abstracción de acceso al núcleo. Algunas bibliotecas como la biblioteca de GNU proveen funcionalidad adicional para facilitar la vida del programador y usuario o mejorar el rendimiento de los programas. En un sistema con micronúcleo la biblioteca de lenguaje C puede gestionar sistemas de archivos o controladores además del acceso al núcleo del sistema.

A los sistemas operativos que llevan Linux se les llama de forma genérica distribuciones Linux. Estas consisten en una recopilación de software que incluye el núcleo Linux y el resto de programas necesarios para completar un sistema operativo. Las distribuciones más comunes son de hecho distribuciones GNU/Linux o distribuciones Android. El hecho de que compartan núcleo no significa que sean compatibles entre sí. Una aplicación hecha para GNU/Linux no es compatible con Android sin la labor adicional necesaria para que sea multiplataforma.

Las distribuciones GNU/Linux usan Linux como núcleo junto con el entorno de tiempo de ejecución del Proyecto GNU y una serie de programas y herramientas del mismo que garantizan un sistema funcional mínimo. La mayoría de distribuciones GNU/Linux incluye software adicional como entornos gráficos o navegadores Web así como los programas necesarios para permitirse instalar a sí mismas. Los programas de instalación son aportados por el desarrollador de la distribución. Se les conoce como gestores de paquetes. Los creadores de una distribución también se pueden encargar de añadir configuraciones iniciales de los distintos programas incluidos en la distribución.

Las distribuciones Android incluyen el núcleo Linux junto con el entorno de ejecución y herramientas del proyecto AOSP de Google. Cada fabricante de teléfonos dispone de su propia distribución de Android a la cual modifica, elimina o añade programas extra: interfaces gráficas, tiendas de aplicaciones y clientes de correo electrónico son algunos ejemplos de programas suscepti-

bles de ser añadidos, modificados o eliminados. Además de las distribuciones de los fabricantes de teléfonos existen grupos de programadores independientes que también desarrollan distribuciones de Android. LineageOS y Replicant son dos ejemplos de distribuciones Android independientes.

Los usuarios de Linux/Unix estamos acostumbrados a hablar y oír hablar sobre su Kernel[21], el cual puede actualizarse y manipularse en cualquier distribución. Sin embargo, en un sistema operativo tan centrado en el usuario y la sencillez como Windows, su Kernel es un gran desconocido.

Kernel de Windows en la década de los noventa Microsoft estaba basando sus sistemas operativos en los Kernel Windows 9x, donde el código básico tenía muchas similitudes con MS-DOS. De hecho necesitaba recurrir a él para poder operar. Paralelamente, Microsoft también estaba desarrollando otra versión de su sistema dirigido a los servidores llamada Windows NT.

Ambas versiones de Windows fueron desarrollándose por separado. Windows NT era más bien una jugada a largo plazo, una tecnología para ir desarrollando para los Windows del mañana, y en el año 2000 dieron un nuevo paso en esa dirección. A la versión 5.0 de NT la llamaron Windows 2000, y se convirtió en un interesante participante en el sector empresarial.

Tras ver la buena acogida que tuvo, Microsoft decidió llevar NT al resto de usuarios para que ambas ramificaciones convergien. Lo hicieron en octubre del 2001 con la versión 5.1 de Windows NT, que llegó al mercado con el nombre de Windows XP. Por lo tanto, esta versión marcó un antes y un después no sólo por su gran impacto en el mercado, sino porque era el principio de la aventura del Kernel Windows NT en el mundo de los usuarios comunes.

Desde ese día, todas las versiones de Windows han estado basadas en este Kernel con más de 20 años de edad. La versión 5.1.2600 fue Windows XP, la 6.0.6002 fue Windows Vista, y la 6.1.7601 Windows 7. Antes hubo otros

[21] En el caso de los sistemas derivados de Unix y Linux el Kernel lo podemos encontrar en el directorio /boot/, este directorio incluye todos los ejecutables y archivos que son necesarios en el proceso de arranque del sistema y deben ser utilizados antes que el Kernel empiece a dar las órdenes de ejecución de los diferentes módulos del sistema, aquí también es donde reside el gestor de arranque.

En algunas distribuciones al usar un gestor de volúmenes lógico (Logical Volume Manager, LVM) se genera un esquema de particiones con el directorio boot en una partición aparte.

Windows Server 2008 y 2003, y después llegaron las versiones de NT 6.2.9200 llamada Windows 8, la 6.3.9600 o Windows 8, la NT 10.0, también conocida como Windows 10 y finalmente Windows 11.

La principal característica del Kernel de Windows NT es que es bastante modular, y está basada en dos capas principales, la de usuario y la de Kernel. El sistema utiliza cada una para diferentes tipos de programa. Por ejemplo, las aplicaciones se ejecutan en el modo usuario, y los componentes principales del sistema operativo en el modo Kernel. Mientras, la mayoría de los Drivers suelen usar el modo Kernel, aunque con excepciones.

Es por eso que se refieren a él como Kernel híbrido, pero sobre todo también porque permite tener subsistemas en el espacio del usuario que se comunicaban con el Kernel a través de un mecanismo de intercomunicación de procesos IPC (Interprocess Communication).

Cuando ejecutas una aplicación, está accede al modo usuario, donde Windows crea un proceso específico para la aplicación. Cada aplicación tiene su dirección virtual privada, ninguna puede alterar los datos que pertenecen a otra y tampoco acceder al espacio virtual del propio sistema operativo. Es por lo tanto el modo que menos privilegios otorga, incluso el acceso al Hardware está limitado, y para pedir los servicios del sistema las aplicaciones tienen que recurrir a la interfaz de programación de aplicaciones API (Application Programming Interface) de Windows.

El modo núcleo en cambio es ese en el que el código que se ejecuta en él tiene acceso directo a todo el Hardware y toda la memoria del equipo. Aquí todo el código comparte un mismo espacio virtual, y puede incluso acceder a los espacios de dirección de todos los procesos del modo usuario. Esto es peligroso, ya que si un Driver en el modo Kernel modifica lo que no debe, podría afectar al funcionamiento de todo el sistema operativo.

Este modo núcleo está formado por servicios Executive, como el controlador de Caché, el gestor de comunicación, gestor de E/S, las llamadas de procedimientos locales, o los gestores de energía y memoria entre otros. Estos a su vez están formados por varios módulos que realizan tareas específicas, controladores de núcleo, un núcleo y una capa de abstracción del Hardware HAL (Hardware Abstraction Layer).

Diferencias entre los Kernel de Linux y Windows La principal diferencia entre el Kernel de los sistemas operativos Windows y el de Linux está en su filosofía. El desarrollado por el equipo de Linus Torvalds es de

código abierto y cualquiera puede usarlo y modificarlo, algo que le sirve para estar presente en múltiples sistemas operativos o distribuciones GNU/Linux. El Kernel de Microsoft[22] en cambio es bastante más cerrado, y está hecho por y para el sistema operativo Windows.

En esencia, en Linux adoptaron los principios de modularidad de Unix y decidieron abrir el código y las discusiones técnicas. Gracias a ello, Linux ha creado una comunidad meritocrática de desarrolladores, una en la que todos pueden colaborar y en la que cada cambio que se sugiere se debate con dureza para desechar las peores ideas y quedarse con las mejores. También se halaga a quienes consiguen mejorar las funcionalidades más veteranas.

Mientras, en Windows no funciona así, los responsables del Kernel no ven con buenos ojos que se hagan propuestas que se desvíen del plan de trabajo, y asegura que hay pocos incentivos para mejorar las funcionalidades existentes que no sean prioritarias.

Esto hace, a ojos de ese antiguo desarrollador, que al dársele mayor importancia a cumplir planes que a aceptar cambios que mejoren la calidad del producto, o al no tener tantos programadores sin experiencia, el Kernel de Windows NT siempre esté un paso por detrás en estabilidad y funcionalidades.

A nivel técnico existen similitudes entre ambos. Los dos núcleos controlan el Software del sistema de bajo nivel y las interacciones con el Hardware del ordenador a través de la capa de abstracción de Hardware (HAL). El HAL es un elemento del sistema que funciona como interfaz entre Software y Hardware, y como las API, permite que las aplicaciones sean independientes del Hardware.

Los dos están escritos principalmente en C, y son capaces de manejar

[22] Para conocer la información del Kernel de Windows usando la línea de comandos podemos utilizar el siguiente comando en un cmd shell:

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Y en powershell:

```
Get-CimInstance Win32_OperatingSystem | Select-Object Caption, CS-
DVersion, ServicePackMajorVersion, BuildNumber | FL
```

o

```
[System.Environment]::OSVersion.Version
```

el almacenamiento en Caché, los controladores de dispositivos, la memoria virtual, los sistemas de archivos, los protocolos de red y las llamadas de sistema. En esencia sus funcionalidades son las mismas, aunque la manera de llevarlas a cabo es diferente.

Así como el Kernel de Windows tiene dos modos, y por lo tanto se le considera híbrido, la gran diferencia es que el de Linux sólo tiene una capa, o sea que es un núcleo monolítico. Eso sí, pese a ser más sencillo en este aspecto, para funcionar correctamente tiene su núcleo dividido en tres subcapas diferentes.

Ambos gestionan los problemas de memoria de forma parecida. Tienen sistemas de "Swapping" para mover un proceso o parte de él temporalmente de la memoria principal a una secundaria de almacenamiento en el caso de que en la principal haya poco espacio. Windows lo hace en los ficheros Pagefile.sys y Swapfile.sys, mientras que Linux lo suele hacer en una partición, aunque también lo puede hacer en uno o varios ficheros o deshabilitarlo.

Por lo tanto, podemos decir que la principal diferencia entre ambos es la manera en que se desarrolla cada uno. Además, el Kernel de Linux es mucho más sencillo, lo cual es bueno para los desarrolladores. Mientras, el de Windows intenta poner una capa de protección en su modo usuario para que los usuarios con menos conocimientos tengan menos posibilidades de dañar el sistema, y su estructura lo hace más estable frente, por ejemplo a fallos del Driver gráfico.

Pero todo esto ya está cambiando, en las últimas versiones de Windows 10 y 11, Microsoft está integrando el Kernel de Linux a su propio Kernel y esto ha permitido usar Linux dentro de Windows de forma nativa gracias al llamado Windows Subsystem for Linux (WSL, WSL2 WSLg), lo cual ha permitido mejorar la estabilidad y desempeño de Windows.

Kernel de Android Durante la última conferencia de Linux Plumbers 2021, Google dio a conocer sobre el éxito de la iniciativa de mover la plataforma Android para usar un Kernel normal de Linux en lugar de usar su propia versión del Kernel, que incluye cambios específicos para la plataforma Android.

Google menciona que dicho cambio de desarrollo es debido a la decisión de pasar después del año 2023 al modelo «Upstream First», que implica el desarrollo de todas las funciones nuevas del Kernel requeridas en la plataforma Android directamente en el Kernel principal de Linux y no en sus ramas separadas (la funcionalidad será primero se promocionará al Kernel principal

y luego se usará en Android, y no al revés).

Para 2023 y 2024, también se planea transferir al núcleo principal de todos los parches adicionales que quedan en la rama del Kernel común de Android.

En cuanto a un futuro próximo, para la plataforma Android 12 prevista para principios de octubre, se ofrecerán compilaciones del Kernel «Generic Kernel Image» (GKI), lo más parecido posible al Kernel 5.10 habitual.

Para estas compilaciones se proporcionará un lanzamiento regular de actualizaciones, que se colocarán en el repositorio ci.android.com. En el Kernel de GKI, las adiciones específicas de Android, así como los controladores relacionados con el Hardware de los fabricantes de equipos originales, se mueven a módulos de Kernel separados.

Esta nueva interfaz, conocida como Kernel Module Kjos, garantizará que la principal diferencia entre la imagen genérica del Kernel de Android (GKI) y la línea principal de Linux, sean solo los ganchos para todos los módulos específicos del proveedor.

Estos módulos no están vinculados a la versión principal del Kernel y se pueden desarrollar por separado, lo que simplifica enormemente el mantenimiento y la transferencia de dispositivos a nuevas ramas del Kernel. Las interfaces necesarias para los fabricantes de dispositivos se implementan en forma de ganchos que le permiten cambiar el comportamiento del Kernel sin realizar cambios en el código.

En total, el Kernel Android 12-5.10 ofrece 194 ganchos comunes, similares a los puntos de seguimiento, y 107 ganchos especializados que le permiten ejecutar controladores en un contexto no atómico. En el Kernel de GKI, los fabricantes de Hardware tienen prohibido aplicar parches específicos al Kernel principal, y los proveedores deben suministrar los componentes para el Hardware de soporte sólo en forma de módulos de Kernel adicionales, en los que se debe garantizar la compatibilidad con el Kernel principal.

Debemos recordar que la plataforma Android desarrolla su propia rama del Kernel: el «Android Common Kernel», sobre la base del cual se forman las compilaciones específicas separadas para cada dispositivo.

Con lo cual, a partir de cada rama de Android, se proporciona a los fabricantes múltiples diseños de Kernel para sus dispositivos. Por ejemplo, Android 11 ofreció una opción de tres núcleos base a la vez: 4.14, 4.19 y 5.4, y para Android 12, se ofrecerán los núcleos base 4.19, 5.4 y 5.10. La variante 5.10 está diseñada como una imagen de Kernel genérica, en la que las capacidades necesarias para los OEM se transfieren al flujo ascendente, se mueven a módulos o se transfieren al Kernel común de Android.

Antes de la llegada de GKI, el Kernel de Android pasó por varias etapas de preparación:

- La primera de ellas era sobre la base de los principales Kernels LTS (3.18, 4.4, 4.9, 4.14, 4.19, 5.4) y de los cuales se creó una bifurcación del «Android Common Kernel», al que se transferían parches específicos para Android (anteriormente, se alcanzaba el tamaño de los cambios varios millones de líneas).
- Después de ello sobre «Android Common Kernel», los fabricantes de Chips como Qualcomm, Samsung y MediaTek forman el SoC Kernel, que incluye complementos para admitir Hardware.
- Finalmente en el «Kernel de SoC», los fabricantes de dispositivos crean el «Kernel de dispositivo», incluidos los cambios relacionados con la compatibilidad con equipos adicionales, pantallas, cámaras, sistemas de sonido, etc.

Este enfoque complicó significativamente la entrega de actualizaciones con la eliminación de vulnerabilidades y la transición a nuevas ramas del Kernel. Si bien Google publica regularmente actualizaciones para su núcleo común de Android, los proveedores a menudo tardan en enviar estas actualizaciones o usan un solo Kernel durante todo el ciclo de vida del dispositivo, generando una alta fragmentación en el ecosistema Android, una obsolescencia anticipada y en el peor de los casos brechas de seguridad.

Otros Kernels GNU/Linux Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvoltura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado (Liquorix viene optimizado para multimedia y Juegos, por otro lado XanMod tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más para

cálculos intensivos) de las últimas versiones estable del Kernel. Estos se pueden instalar[23] mediante el uso de los comandos *dpkg* o *apt* (después de agregarlo a nuestro repositorio */etc/apt/sources.list.d/*), de esta forma siempre podremos tener la última versión del Kernel junto con la actualización básica de nuestro sistema GNU/Linux.

Además, sí instalamos cualquiera de los distintos Kernels, siempre podemos seleccionar alguno de los instalados al momento de arrancar nuestro equipo para usarlo de acuerdo a las actividades requeridas en ese momento. Y en caso necesario, es fácil su desinstalación y continuar usando el Kernel que teníamos por defecto.

Por otro lado, existe una versión completamente libre del Kernel de GNU/Linux (Linux-Libre) el cual es un Kernel despojado de elementos de Firmware y controladores que contienen componentes no libres o fragmentos de código cuyo alcance está limitado por el fabricante.

Linux-libre es el núcleo recomendado por la Free Software Foundation y una pieza principal de las distribuciones GNU totalmente libres de fragmentos privativos o Firmwares incluidos en Linux sirven para inicializar los dispositivos o aplicarles parches que solventan fallas del Hardware que no pudieron ser corregidas antes de ser puestos a disposición de los usuarios.

Además, Linux-libre deshabilita las funciones del Kernel para cargar componentes no libres que no forman parte del suministro del Kernel y elimina la mención del uso de componentes no libres de la documentación. El Kernel de Linux-libre se utiliza en distribuciones como Dragora Linux, Trisquel, Dyne, Bolic, gNewSense, Parabola, Musix y Kongoni.

Estabilidad del Kernel La estabilidad de un núcleo no es tan difícil. El Kernel de Unix de Mac o el Kernel de Linux están diseñados de manera diferente pero resuelven el mismo problema. El sistema operativo Windows es igualmente robusto. Eso es evidente.

Pero la estabilidad real del día a día depende de otros factores. Particularmente en los controladores que conectan el sistema operativo al Hardware. Aquí es donde surgen las diferencias.

Apple tiene el momento más fácil, porque solo admiten un pequeño conjunto de Hardware seleccionado. Eso facilita su trabajo, el objetivo es pequeño. Apple ocasionalmente estropea esto, pero en su mayor parte, OS X

[23] Para ver las opciones de optimización del Kernel y como instalarlo ver la página Web de cada proyecto: Liquorix, XanMod, Linux-Libre.

ofrece una notable estabilidad diaria para las aplicaciones de escritorio.

Windows tiene un trabajo mucho más difícil. El sistema operativo admite una gama simplemente gigantesca de Hardware, y los fabricantes de Hardware hacen todo lo posible para proporcionar controladores de alta calidad. Este es el valor real de Windows, como paquete de controladores, es prácticamente imbatible.

Linux también intenta admitir una gran variedad de Hardware. Pero muchos fabricantes de Hardware son absolutamente indiferentes a la compatibilidad con Linux. Por lo tanto, el soporte de Hardware en Linux es mucho más impredecible. Si está ejecutando un servidor en Linux y el sistema solo se comunica con un disco duro y un adaptador de red, es probable que tenga una estabilidad impecable que supere a la industria.

Pero, si instala Linux en una computadora portátil y espera que funcione con la función de reposo / activación, la GPU, la tarjeta de sonido y un montón de extravagantes periféricos, entonces podría alejarse del rango de la estabilidad. Hay algunos controladores de código abierto, pero estos son significativamente peores que las versiones de los fabricantes. Por ejemplo, una GPU puede ser 4 o 5 veces más lenta.

Como usuario de escritorio de Linux, es probable que también instale aplicaciones que hacen cosas interesantes, de diversos proveedores, que pueden no estar del todo de acuerdo con las mejores prácticas.

Las Vulnerabilidades y Exposiciones Comunes El mundo está cada vez más interconectado y, como resultado de esto, la exposición a las vulnerabilidades de seguridad también ha aumentado dramáticamente. Las complejidades de mantener las plataformas de cómputo actuales hacen que sea muy difícil para los desarrolladores cubrir cada punto de entrada potencial. En 2019 hubo un promedio de más de 45 vulnerabilidades y exposiciones comunes registradas por día y estas siguen en aumento año con año.

Las vulnerabilidades y exposiciones comunes (Common Vulnerabilities and Exposures, CVE https://cve.mitre.org) que tienen los distintos sistemas operativos, es una lista de información registrada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación CVE-ID, descripción de la vulnerabilidad, que versiones del Software están afectadas, posible solución al fallo (si existe) o como configurar para mitigar la vulnerabilidad y referencias a publicaciones o entradas de foros o Blogs donde se ha hecho pública la vulnerabilidad o se demues-

tra su explotación. Además suele también mostrarse un enlace directo a la información de la base de datos de vulnerabilidades (https://nvd.nist.gov, https://openssf.org y https://docs.aws.amazon.com/security), en la que pueden conseguirse más detalles de la vulnerabilidad y su valoración.

El CVE-ID ofrece una nomenclatura estándar para identificación de la vulnerabilidad de forma inequívoca que es usada en la mayoría de repositorios de vulnerabilidades. Es definido y es mantenido por The MITRE Corporation (por eso a veces a la lista se la conoce por el nombre MITRE CVE List) con fondos de la National Cyber Security Division del gobierno de los Estados Unidos de América. Forma parte del llamado Security Content Automation Protocol.

Mitos en torno a Linux/Unix Hay varios mitos en torno a Linux/Unix y al Software libre, a saber:

- Linux/Unix se puede usar para revivir un equipo de cómputo viejo. La realidad es que si bien, hay múltiples distribuciones de Linux/Unix que corren en una gran cantidad de procesadores antiguos y actuales, los Drivers necesarios para reconocer periféricos como tarjetas gráficas, de red alámbrica e inalámbrica, entre muchos otros, no tienen soporte en Linux/Unix, lo cual hará imposible su uso en Linux/Unix. Esto es cierto en cualquier computadora no importa de cuál generación es el equipo de cómputo. La verdad de todo esto, es que los fabricantes están enfocados en producir Hardware y Drivers que corran en los sistemas operativos con mayor cuota de mercado y por el momento Linux/Unix en equipos personales no son de ellos.

- La compatibilidad del Hardware depende en gran medida de la versión de Kernel de GNU/Linux instalado, es de esperarse que en versiones anteriores del Kernel cierto Hardware no se pueda detectar, pero lo contrario también pasa, hay Drivers que solo corren correctamente en versiones anteriores del Kernel y no en las últimas versiones, lo que ocasiona que muchos usuarios se desesperen al tratar de usar sus equipos con GNU/Linux. Y en caso de logar que funcione el Hardware, se fuerza a los usuarios a usar una determinada versión del Kernel (y todas las aplicaciones de la distribución) no actualizable, por la imposibilidad de hacer funcionar el Hardware del equipo en una más moderna con la consiguiente obsolescencia del Software instalado en el equipo.

- Si tengo un Software ahora y quiero ejecutarlo dentro de cinco o diez años en el futuro ¿Por qué no debería ser capaz de hacerlo? Parte de la belleza del Open Source es que el código fuente está disponible, por lo que es más fácil mantener operativo el Software, de modo que no deje de funcionar cuando alguien deja de mantenerlo. Excepto que mantener el Software en Linux/Unix se está convirtiendo en un desafío tan grande que daría igual que fuese privativo. Porque sería complicado hacerlo funcionar en un tiempo razonable, incluso siendo desarrollador, podría costar mucho trabajo y es posible dejar algo sin funcionar en el camino.

- La retrocompatibilidad[24] es un enorme dolor de cabeza, tomar Software hecho para Linux/Unix de hace 10 o 5 años y ejecutarlo en una distribución moderna. Cualquier cosa de mínima complejidad o que use una GUI, simplemente no funciona. Mientras la retrocompatibilidad en Windows es simplemente increíble. En Linux/Unix somos dependientes de los repositorios en línea, y cuando una aplicación depende de ciertas librerías que empiezan a desaparecer de esos repositorios, nos encontramos en una pesadilla. Y mientras más viejo el Software, peor.

2.1 Windows

Microsoft Windows (véase [3]), conocido generalmente como Windows o MS Windows es el nombre de una familia de Software propietario (véase apéndice 12.1) de distribuciones de Software para PC, Smartphone -que perdió cuota de mercado con Android hasta desaparecer-, servidores y sistemas empotrados, desarrollados y vendidos por Microsoft y disponibles para múltiples arquitecturas, tales como x86, x86-64 y ARM.

Desde un punto de vista técnico, no son sistemas operativos, sino que contienen uno (tradicionalmente MS-DOS, o el más actual, cuyo núcleo es Windows NT) junto con una amplia variedad de Software; no obstante, es usual (aunque no necesariamente correcto) denominar al conjunto como sistema operativo en lugar de distribución.

[24] Siempre estamos en posibilidad de usar una Máquina Virtual que nos permite usar un programa desarrollado hace años o décadas en su entorno original, corriendo en un equipo moderno con un sistema operativo de última generación con todas las actualizaciones de seguridad pertinentes.

La versión más reciente de Windows es Windows 11 para equipos personales (que se ofrece como actualización gratuita para los equipos con licencia válida de Windows 10 que cumplan con los requisitos mínimos de Hardware exigidos), Windows Server 2022 para servidores.

Windows 11 tiene al menos siete ediciones con diferente conjunto de características y Hardware previsto, algunas de ellas son: Home, Pro, Pro Education, Pro for Workstations, Enterprise, Education, Mixed Reality. Además habrá un modo SE para hacer frente al Chrome OS de los Chromebooks de Google.

Windows 11 también se puede descargar como una imagen ISO para instalaciones nuevas, ejecución en máquinas virtuales, además de ser la versión de referencia que los fabricantes OEM que usarán para preinstalaciones en equipos nuevos.

Por su parte, Windows 10 tiene al menos doce ediciones con diferente conjunto de características y Hardware previsto, algunas de ellas son: Home, Pro, Enterprise, Enterprise LTBS/LTSC, Education, Mobile, S, Pro for Workstation, Team, Pro Education, loT (Embedded), N y KN. Se espera que cuente con soporte y actualizaciones hasta el 2025.

Todas las ediciones mencionadas tienen la capacidad de utilizar los paquetes de idiomas, lo que permite múltiples idiomas de interfaz de usuario. A pesar de la múltiple cantidad de ediciones, solamente Windows Home y Pro están orientadas para el común de los usuarios y vienen instaladas en equipos nuevos. Las demás ediciones se adquieren mediante otros tipos de compra.

Por su parte, Windows 10 llegó de forma oficial y gratuita a usuarios con licencia genuina de Windows 7, Windows 8.1 y Windows 8 así como a Insiders, siendo la primera versión que buscaba la unificación de dispositivos (escritorio, portátiles, teléfonos inteligentes, tabletas y videoconsolas) bajo una experiencia común, con lo que se esperaba eliminar algunos problemas que se presentaron con Windows 8.1.

Además está Windows PE que es un pequeño sistema operativo usado para instalar, desplegar y reparar Windows 10 en todas sus versiones de escritorio, servidor y para otras ediciones de Windows. En concreto, podemos preparar el disco antes de la instalación, instalar Windows con apps desde un disco local o una red, instalar imágenes de Windows, hacer modificaciones en Windows sin iniciar sesión, recuperar datos perdidos y un largo etcétera.

Seguridad Una de las principales críticas que reciben los sistemas operativos Windows es la debilidad del sistema en lo que a seguridad se refiere y el alto índice de vulnerabilidades críticas. El propio Bill Gates, fundador de Microsoft, ha asegurado en repetidas ocasiones que la seguridad es objetivo primordial para su empresa.

Partiendo de que no existe un sistema completamente libre de errores, las críticas se centran en la lentitud con la que la empresa reacciona ante un problema de seguridad que pueden llegar a meses o incluso años de diferencia desde que se avisa de la vulnerabilidad hasta que se publica la actualización que corrija dicha vulnerabilidad (parche). En algunos casos la falta de respuesta por parte de Microsoft ha provocado que se desarrollen parches que arreglan problemas de seguridad hechos por terceros.

Uno de los pilares en que se basa la seguridad de los productos Windows es la seguridad por ocultación, en general, un aspecto característico del Software propietario que sin embargo parece ser uno de los responsables de la debilidad de este sistema operativo debido a que, la propia seguridad por ocultación, constituye una infracción del uno de los principios de Kerckhoffs, el cual afirma que la seguridad de un sistema reside en su diseño y no en una supuesta ignorancia del diseño por parte del atacante.

Windows 11 Microsoft presentó el 24 de junio del 2021 la siguiente generación de su sistema operativo y que se puede descargar como actualización a partir del 5 de octubre del 2021. Windows 11 con un nuevo diseño que potencia las formas redondeadas y las transparencias, y enfocado a la productividad. Este diseño, que incluye un modo oscuro y uno claro, está pensado para transmitir sensación de calma y sobre todo para que el usuario tenga la información siempre a mano. Esto se aprecia en nuevas funciones, como los Widgets, que pueden personalizar tarjetas de información sobre el tiempo, el tráfico, o información local.

La compañía también ha rediseñado el menú de inicio y la barra de tareas con el acceso a aplicaciones como Teams, que ahora está integrado en Windows. Con Snap Layouts, Windows 11 permite personalizar la apariencia y la disposición de las ventanas. El usuario también podrá personalizar escritorios según el uso, ya sea para el trabajo, para el estudio o el ocio, con configuraciones separadas.

La nueva experiencia con las aplicaciones y son el 'Docking' –cuando el portátil se conecta a un monitor– hacen posible retomar el trabajo, incluso

con varias aplicaciones abiertas, en el mismo lugar donde el usuario lo dejó antes de apartarse del ordenador. Windows 11 también está diseñado para que siempre se sienta igual tanto si se usa en una tableta y su pantalla táctil como si se usa en un ordenador con teclado. Además, mejora la experiencia con el lápiz óptico y teclado táctil en pantalla.

Otro elemento que presenta novedades es Microsoft Store, que ha sido rediseñada para facilitar la búsqueda de aplicaciones, e incorpora también las aplicaciones de Android, que pueden colocarse en la barra de tareas. Con Windows 11 se tendrá una nueva tienda de aplicaciones, una que abrirá sus puertas a todo tipo de aplicaciones y juegos. Esto quiere decir que a partir de ahora la nueva generación de Windows contará con PWA (aplicaciones Web progresivas), UWP (aplicaciones universales) y los programas clásicos Win32 en un mismo lugar.

En cuanto al tema de actualizaciones, Microsoft abandona definitivamente el programa de entrega de actualizaciones semestrales de Windows 10 que definitivamente no pudo concretar con la estabilidad requerida. Windows 11 solo recibirá una actualización anual de características, funciones y soporte de nuevas tecnologías, lo que debería otorgarle tiempo suficiente para desarrollo y pruebas, entregando versiones más pulidas. Las actualizaciones acumulativas de seguridad y corrección de errores si mantendrán el actual ciclo de entrega mensual.

En octubre del 2022 se publicó[25] que Windows 10 se mantiene con una cuota de mercado de 71.29%, mientras que Windows 11 tiene una cuota de 15.44%, esta diferencia deduce por los altos requisitos mínimos de Hardware exigidos -4 GB RAM, TPM 2.0 y 60 GB de Disco-. Por otra parte, las cuotas de mercado de las versiones sin soporte de Windows son: Windows 8.1 de 2.45%, Windows 8 de 0.69%, Windows 7 de 9.61% y Windows XP de 0.39%.

Windows 365 Microsoft presentó en julio del 2021 la nueva versión de Windows en la nube que llevará el nombre de Windows 365. Este nuevo servicio de suscripción, está especialmente dirigido a empresas, que permitirá acceder a nuestra sesión de usuario desde cualquier equipo (el PC, el Mac, la tableta o teléfono Android, etc.), pues el Software y el sistema de

[25] Cuotas de mercado de diferentes sistemas operativos:

https://gs.statcounter.com/os-market-share/desktop/worldwide
https://netmarketshare.com

archivos están alojados en una máquina virtual remota, por lo que la configuración, documentos y herramientas disponibles serán idénticos desde donde accedamos.

Así, desde cualquier navegador (o bien usando la aplicación de Escritorio Remoto de Windows) podremos acceder a un Windows 11/10 disfrutando de una experiencia de arranque casi instantáneo, pero esa no será la única ventaja de esta plataforma. Aún más importante será la posibilidad de contar con varios 'ordenadores' en una misma cuenta, cada uno con distinta potencia (RAM, núcleos de procesador...) y capacidad de almacenamiento contratada, según el trabajo que necesitemos llevar a cabo en cada momento.

Microsoft ya ha confirmado que ofrecerá 12 configuraciones de Hardware distintas para sus equipos virtualizados (iniciando en $130 pesos mexicanos por mes). Así, las empresas podrán 'crear PCs' en cuestión de minutos y asignar cada uno a un empleado, eliminando los inconvenientes que conlleva el hecho de manejar Hardware físico.

Windows 11 SE Microsoft anunció en noviembre del 2021 el lanzamiento de una nueva versión de Windows 11 que hará compañía a las versiones 'Pro' y 'Home': su nombre es Windows 11 SE, llevábamos oyendo rumores al respecto desde junio y desembarca ahora para hacer frente al Chrome OS de los Chromebooks de Google, por lo que se destinará únicamente en portátiles escolares de bajo costo.

El sistema estará optimizado para su uso con MS Edge, MS Office y el resto de servicios de la nube de Microsoft, estará abierto a muchas más aplicaciones de terceros. En palabras de Paige Johnson, directora de marketing educativo de Microsoft, "Windows 11 SE también es compatible con aplicaciones de terceros, incluidas Zoom y Chrome, porque queremos dar a las escuelas la opción de usar lo que funcione mejor para ellas".

Winget Windows Package Manager (*winget*) es el gestor de paquetes de Windows 11 y 10 que permite usar comandos desde la terminal para instalar aplicaciones de forma rápida y sencilla (al más puro estilo de Linux). El único requisito para instalar *winget* es contar con Windows 10 1890 o versión posterior, es posible usar para actualizar una o todas las aplicaciones del sistema usando:

```
C: winget upgrade –all
```

también es posible usar *WInstall* interfaz gráfica no oficial de *winget* para elegir programas en un click desde una Web y luego instalarlos en Windows con *winget* con la misma rapidez y automatización.

Microsoft Open Source En Agosto del 2020 presentó la empresa de Redmond el nuevo sitio Microsoft Open Source en que el público puede navegar a través de todo el ecosistema de código abierto que ha estado construyendo en los últimos años. La Web no solo muestra los proyectos Open Source de Microsoft sino que cuenta con secciones para colaborar con la comunidad, descargar herramientas, explorar su código, y hasta encontrar oportunidades de trabajo.

Las dos partes más importantes de este nuevo sitio son las secciones "Get involved" y "Explore proyects". En la primera se puede revisar toda la actividad reciente en los proyectos Open Source de Microsoft alojados en GitHub, y además se cuenta con una larga lista de recursos para aprender a colaborar con proyectos de código abierto, y no necesariamente solo los que mantiene Microsoft.

La segunda sección es la lista de proyectos, y ahí nos encontramos los principales proyectos Open Source mantenidos por los ingenieros de Microsoft y la comunidad. La lista de proyectos es larga y podemos encontrar los proyectos de los empleados de la empresa patrocinados a través de Microsoft FOSS Fund.

Linux Dentro de Windows Desde el 2018 se inició la integración de GNU/Linux en Windows 10, con la actualización de Windows 10 Fall Creator Update con WSL (Windows Subsystem for Linux), se permitía instalar consolas de diversas distribuciones de GNU/Linux como un programa más. Y en el 2020, con la llegada de Windows 10 Build 2020 con WSL2, el cual cuenta con su propio Kernel de Linux que permite instalar de manera casi nativa diversas distribuciones de GNU/Linux con todo el ambiente gráfico permitiendo tener lo mejor de ambos mundos en un mismo equipo -sin hacer uso de programas de virtualización-, incluso es posible ejecutar varias distribuciones de Linux al mismo tiempo en pantalla.

Para usarlo hay que tener todas las actualizaciones de Windows y activar el Subsistema de Windows para Linux (WSL[26]). Reiniciando el sistema, ya podemos usar distribuciones de Linux desde Microsoft Store.

[26] https://docs.microsoft.com/en-us/windows/wsl/install-win10

En el Windows Insider Preview Build 20150 ha incluido soporte para GPU de Intel, AMD y NVIDIA y es compatible con Direct ML (una API de bajo nivel para aprendizaje automático soportado por DirectX 12) permitiendo el uso de las capacidades de computación por GPU de WSL para Linux.

En abril del 2021 se anunció la llegada de WSLg en la que se pueden correr aplicaciones (como gedit, Audacity, etc.) e IDEs con soporte para X11 y Wayland, PulseAudio de Linux con GUI (Graphical User Interface) con soporte para gráficos 3D acelerados por Hardware de forma independiente de la distribución de Linux en la que se instalaron, esta novedad está disponible a partir de la versión Windows Insider Preview Build 21364 y para todos los usuarios en Windows 10 May 2020 Update.

Android Dentro de Windows En el Windows Build 20185 ha incluido soporte para que Windows 10 permite no sólo sincronizar teléfonos Android, sino además mediante "your Phone" permite integrar las aplicaciones, notificaciones, mensajes, fotos, llamadas y otras opciones de teléfonos inteligentes en Android directamente en Windows, ejecutando las aplicaciones sin tener que abrirlas en el teléfono, aunque siguen proviniendo de ahí.

Además en noviembre de 2020, se ha informado que existe la posibilidad de que Microsoft permita la instalación y ejecución de aplicaciones para Android en Windows 11. Con cambios mínimos o directamente sin modificaciones en el código, los desarrolladores podrían enviar a Microsoft Store sus aplicaciones para que sean descargadas e instaladas en PCs, el proyecto tiene el nombre de Latte.

Microsoft Azure Durante los últimos años, Microsoft ha declarado en conferencias magistrales de eventos y en otros lugares que Linux es un sistema operativo de rápido crecimiento que se usa dentro de Microsoft Azure.

Han declarado con orgullo que el 50 % de las máquinas virtuales nuevas que se ejecutaban en Azure ejecutaban Linux. En Octubre del 2022, las estadísticas reportadas señalan:

- Más del 50 % de los núcleos de máquinas virtuales ejecutan Linux en Azure
- Las imágenes basadas en Linux comprenden el 60 % de las imágenes de Azure Marketplace

- Los 100 principales clientes de Microsoft implementan cargas de trabajo de Linux en Azure
- Azure Tuned Kernels proporciona un rendimiento de red un 25 % más rápido
- Microsoft es compatible con todas las principales distribuciones de Linux, como: Red Hat, SUSE, Ubuntu, Oracle Linux, Debian, CentOS, CoreOS y OpenSUSE (Relacionado: Azure también es compatible con FreeBSD)
- Azure ofrece dos servicios de orquestación de Kubernetes administrados con soporte nativo: Azure Kubernetes Service y Azure Red Hat OpenShift

2.2 UNIX y BSD

Unix (véase [**?**]) es un sistema operativo portable, multitarea y multiusuario; desarrollado en 1969 por un grupo de empleados de los laboratorios Bell de AT&T. El sistema, junto con todos los derechos fueron vendidos por AT&T a Novell Inc. Esta vendió posteriormente el Software a Santa Cruz Operation en 1995, y está, a su vez, lo revendió a Caldera Software en 2001, empresa que después se convirtió en el grupo SCO. Sin embargo, Novell siempre argumentó que solo vendió los derechos de uso del Software, pero que retuvo el Copyright sobre "UNIX". En 2010, y tras una larga batalla legal, esta ha pasado nuevamente a ser propiedad de Novell.

Solo los sistemas totalmente compatibles y que se encuentran certificados por la especificación Single UNIX Specification pueden ser denominados "UNIX" (otros reciben la denominación «similar a un sistema Unix»). En ocasiones, suele usarse el término "Unix tradicional" para referirse a Unix o a un sistema operativo que cuenta con las características de UNIX Versión 7 o UNIX System V o UNIX versión 6.

Berkeley Software Distribution o BSD (en español, «distribución de Software Berkeley») (véase [**?**]) fue un sistema operativo derivado de Unix que nace a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley. En los primeros años del sistema Unix sus creadores, los Laboratorios Bell de la compañía AT&T, autorizaron a la Universidad de Berkeley en California y a otras universidades, a utilizar el código fuente y

adaptarlo a sus necesidades. Durante los años 1970 y 1980 Berkeley utilizó el sistema para sus investigaciones en materia de sistemas operativos.

Cuando AT&T retiró el permiso de uso a la universidad por motivos comerciales, la universidad promovió la creación de una versión inspirada en el sistema Unix utilizando los aportes que ellos habían realizado, permitiendo luego su distribución con fines académicos y al cabo de algún tiempo reduciendo al mínimo las restricciones referente a su copia, distribución o modificación (véase apéndice 12.4).

Algunos sistemas operativos descendientes del sistema desarrollado por Berkeley son SunOS, FreeBSD, NetBSD, OpenBSD, DragonFlyBSD y Mac Big Sur. BSD también ha hecho grandes contribuciones en el campo de los sistemas operativos en general. Además, la licencia permisiva de BSD ha permitido que otros sistemas operativos, tanto libres como propietarios incorporaron código BSD. Por ejemplo, Microsoft Windows ha utilizado código derivado de BSD en su implementación de TCP/IP, y utiliza versiones recompiladas de la línea de comandos BSD para las herramientas de redes. También Darwin, el sistema en el cual está construido Mac Big Sur, el sistema operativo de Apple, está derivado en parte de FreeBSD 5. Otros sistemas basados en Unix comerciales como Solaris también utilizan código BSD.

Algunos proyectos activos descendientes del sistema BSD son:

FreeBSD (https://www.freebsd.org/es/)
Es un sistema operativo para computadoras basadas en las CPU de arquitectura Intel. También funciona con procesadores compatibles como AMD. Está basado en la versión 4.4 BSD-Lite del CSRG (Computer Systems Research Group) y fue escrito en C y C++. Tiene Licencia BSD. Este proyecto ha realizado una gran inversión de tiempo en ajustar el sistema para ofrecer las mejores condiciones de rendimiento con carga real y facilidad de uso al usuario final.

NetBSD (https://www.netbsd.org)
Está basado en un conjunto de aplicaciones Open Source, incluyendo 4.4 BSD-Lite de la Universidad de California en Berkeley, Net/2 (Berkeley Networking Release 2), el sistema gráfico X del MIT y aplicaciones del proyecto GNU. Tiene Licencia BSD. NetBSD ha invertido sus energías en proveer de un sistema operativo estable, multiplataforma, seguro y orientado a la inves-

tigación. Está portado a 56 arquitecturas de Hardware y suele ser el primero en implementar tecnologías nuevas, como IPv6.

OpenBSD (https://www.openbsd.org)
Está basado en 4.4 BSD y es un descendiente de NetBSD. El proyecto tiene el foco puesto de forma particular en la seguridad y criptografía. Los esfuerzos se centran en la portabilidad, cumplimiento de normas, corrección, seguridad y criptografía integrada. Tiene Licencia BSD. La filosofía del proyecto puede ser descrita en tres palabras: "Free, Functional and Secure" (Libre, Funcional y Seguro).

DragonFlyBSD (https://www.dragonflybsd.org)
Tiene como meta ofrecer un alto rendimiento y escalabilidad bajo cualquier entorno, desde computadoras de un solo usuario hasta enormes sistemas de clústeres. DragonFlyBSD tiene varios objetivos técnicos a largo plazo, pero el desarrollo se centra en ofrecer una infraestructura habilitada para SMP que sea fácil de entender, mantener y desarrollar.

2.3 Apple y sus macOS e iOS

Apple a la empresa multinacional estadounidense Apple Inc., dedicada al diseño, la confección y la comercialización de productos electrónicos y de Software, así como de los servicios en línea (a través de Internet) que los atañen. Es considerada como una de las sociedades más apreciables del mundo. Su función principal es la producción de aparatos digitales populares, las marcas más populares de esta compañía son: Mac, iPods, iPads e iPhones. Y cuya gama de productos ha ganado un nicho particular en el área de los Gadgets tecnológicos mediante su estética común, intensa mercadotecnia y pretendida alternatividad respecto a otras empresas hegemónicas como Microsoft.

En la actualidad los principales productos de Apple gozan de una amplia popularidad a nivel mundial, particularmente en lo vinculado con reproductores de música, computadores personales, tabletas, teléfonos, relojes inteligentes, Wearables y toda una red de Software que va desde sistemas operativos, programas de gestión de multimedios (como iTunes) o suites de edición profesional. Se trata de una empresa líder en innovación tecnológica y computarizada.

El sistema de Apple siempre se ha caracterizado por contar con detalles no solo sofisticados, sino también fuera de lo común, tratando de marcar la historia de la tecnología bajo Software con distintas mejoras, asistentes virtuales y muchos elementos que dejan gran satisfacción y que debes conocer.

Realizando un seguimiento de los últimos productos lanzados y presentados por los chicos de la manzanita nos damos cuenta de que sin duda son los mejores creando expectación. En Apple saben que para vender hay que mostrar cosas nuevas e innovar, ya que vivimos en un mundo en el que la tecnología evoluciona constantemente y si no lo haces, te quedas atrás. Pero..., ¿siempre son cosas nuevas las que nos muestran?.

Una de las cosas en las que destaca Apple por encima de sus competidores es en el campo de la investigación y la innovación. Las inversiones que realiza la empresa más valiosa del mundo en investigación son enormes, y no en vano. Porque si algo sabe hacer bien Apple es ir por delante de la competencia. En esto todos estamos de acuerdo, los de Cupertino saben mejor que nadie que innovar significa no tener competencia en el mercado. Sin duda, esto saben aprovecharlo y a veces, de tal manera que ni nos damos cuenta.

Pero en Apple tienen una fea costumbre que analizando un poco las últimas Keynotes nos damos cuenta. Los de Cupertino suelen vender como algo innovador y único cosas que ya hacía antes pero pasaba por alto. ¿Qué significa esto?, que en Apple saben como vender las características de sus productos para que nos parezcan espectaculares.

Las computadoras de Apple tienen un sistema operativo único y otras características que sólo están disponibles en las Macs. El diseño del Hardware de Apple unifica la marca, con productos como el iMac "todo en uno", el iMac original con sus colores brillantes y la gama de computadoras portátiles. Además, las nuevas características del sistema operativo que integran el uso de una computadora Mac, iPad, el teléfono o el iPod con iTunes de Apple y la tienda de aplicaciones, proporcionan a los clientes una experiencia de Apple aerodinámica.

Características de Mac

- Carpetas inteligentes en Finder
- Grabe la actividad de la pantalla macOS en QuickTime
- Activa las esquinas calientes de tu pantalla

- Reproduce música y películas
- Ejecutar Windows con Boot Camp
- Automatiza las tareas repetitivas
- Crea escritorios virtuales
- Ping archivos de forma inalámbrica con AirDrop
- Firmar documentos en Vista previa
- Autocompletar palabras a medida que escribe

Características de iOS

- Notificaciones innovadoras
- Widgets de máxima utilidad
- Puedes borrar las aplicaciones de fábrica
- Siri abre apps de terceros
- Varios idiomas para el teclado
- Reconocimiento facial en sus fotos
- Te permite controlar tu hogar
- 3D touch con muchos usos
- Dispone de mensajería interna: iMessage

Mac OS y macOS Ventura Mac OS (véase [5]) -del inglés Macintosh Operating System, en español Sistema Operativo Macintosh- es el nombre del sistema operativo propietario (véase apéndice 12.1) creado por Apple para su línea de computadoras Macintosh, también aplicado retroactivamente a las versiones anteriores a System 7.6, y que apareció por primera vez en System 7.5.1. Es conocido por haber sido uno de los primeros sistemas dirigidos a un gran público al contar con una interfaz gráfica compuesta por la interacción del Mouse con ventanas, íconos y menús.

Debido a la existencia del sistema operativo en los primeros años de su línea Macintosh resultó a favor de que la máquina fuera más agradable al usuario, diferenciándolo de otros sistemas contemporáneos, como MS-DOS, que eran un desafío técnico. El equipo de desarrollo del Mac OS original incluía a Bill Atkinson, Jef Raskin y Andy Hertzfeld.

Este fue el comienzo del Mac OS clásico, desarrollado íntegramente por Apple, cuya primera versión vio la luz en 1985. Su desarrollo se extendería hasta la versión 9 del sistema, lanzada en 1999. A partir de la versión 10 (Mac OS X), el sistema cambió su arquitectura totalmente y comenzó a basarse en BSD Unix, sin embargo su interfaz gráfica mantiene muchos elementos de las versiones anteriores.

Hay una gran variedad de versiones sobre cómo fue desarrollado el Mac OS original y dónde se originaron las ideas subyacentes. Pese a esto, los documentos históricos prueban la existencia de una relación, en sus inicios, entre el proyecto Macintosh y el proyecto Alto de Xerox PARC. Las contribuciones iniciales del Sketchpad de Ivan Sutherland y el On-Line System de Doug Engelbart también fueron significativas.

Versiones Antes de la introducción de los últimos sistemas basados en el microprocesador PowerPC G3, partes significativas del sistema se almacenaban en la memoria física de sólo lectura de la placa base. El propósito inicial de esto fue evitar el uso de la capacidad de almacenamiento limitada de los disquetes de apoyo al sistema, dado que los primeros equipos Macintosh no tenían disco duro. Sólo el modelo Macintosh Classic de 1991, podía ser iniciado desde la memoria ROM.

Esta arquitectura también permitió una interfaz de sistema operativo totalmente gráfica en el nivel más bajo, sin la necesidad de una consola de sólo texto o el modo de comandos de línea. Los errores en tiempo de arranque, como la búsqueda de unidades de disco que no funcionaban, se comunicaban al usuario de manera gráfica, generalmente con un ícono o con mensajes con el tipo de letra Chicago y un "timbre de la muerte" o una serie de pitidos.

Esto contrastaba con los PCs de la época, que mostraban tales mensajes con un tipo de letra monoespaciada sobre un fondo negro, y que requerían el uso del teclado y no de un ratón, para el acceso. Para proporcionar tales detalles en un nivel bajo, Mac OS dependía del Software de la base del sistema grabado en la ROM de la placa base, lo que más tarde ayudó a garantizar que sólo los equipos de Apple o los clones bajo licencia (con el contenido de la

memoria ROM protegido por derechos de autor de Apple, pudieran ejecutar Mac OS).

Mac OS puede ser dividido en tres familias:

- La familia Mac OS Classic, basada en el código propio de Apple Computer.
- El Sistema Operativo Mac OS X, desarrollado a partir de la familia Mac OS Classic y NeXTSTEP, el cual estaba basado en UNIX.
- MacOS Ventura es el reemplazo de macOS Monterrey (que fue el reemplazo de Mac OS X), disponible a partir de junio del 2022, que usando los procesadores de ARM han mostrado un gran desempeño en comparación con equipos INTEL y AMD de gama alta.

Linux Dentro de IOS Es posible tener un Linux completo en IOS además de poder hacer uso de Secure Shell (SSH) a una computadora con Linux. Para la primera forma, se puede ejecutar un sistema virtualizado utilizando Alpine Linux con iSH, que es de código abierto, pero debe instalarse utilizando la aplicación TestFlight propiedad de Apple.

Alternativamente hay aplicaciones de emulador de terminal de código abierto que proporcionan herramientas de código abierto dentro de un entorno restringido. Esta es la opción más limitada -en realidad no nos permite ejecutar Linux, pero estaremos ejecutando herramientas de Linux- pero brindan algunas funciones de línea de comandos. Por ejemplo:

- Sandboxed Shell, con más de 80 comandos e incluye Python 2 y 3, Lua, C, Clang, etc.
- a-Shell, otorga acceso al sistema de archivos e incluye Lua, Python, Tex, Vim, JavaScript, C yC++, junto con Clang y Clang++; y permite instalar paquetes de Python con pip.
- Blink Shell, permite la conexión con servidores.
- iSH, es un Shell Linux que usa *usermode x86* emulación y traducciones de *syscall*.

2.4 GNU/Linux

GNU[27]/Linux (véase [4]) también conocido como Linux, es un sistema operativo libre (véase apéndice 12.2) tipo Unix; multiplataforma, multiusuario y multitarea. El sistema es la combinación de varios proyectos, entre los cuales destacan GNU (encabezado por Richard Stallman y la Free Software Foundation) y el núcleo Linux (encabezado por Linus Torvalds). Su desarrollo es uno de los ejemplos más prominentes de Software libre: todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera, bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres.

A pesar de que «Linux» se denomina en la jerga cotidiana al sistema operativo, este es en realidad solo el Kernel (núcleo) del sistema. La idea de hacer un sistema completo se remonta a mediados de la década de 1980 con el proyecto GNU, así como una gran cantidad de los componentes que se usan hoy en día -además del núcleo-, que van desde los compiladores de GNU hasta entornos de escritorio. Sin embargo, tras la aparición de Linux en la década de 1990 una parte significativa de los medios generales y especializados han utilizado el término «Linux» para referirse a todo. Esto ha sido motivo de polémicas. Cabe señalar que existen derivados de Linux que no tienen componentes GNU -por ejemplo Android-, así como distribuciones de GNU donde Linux está ausente -por ejemplo Debian GNU/Hurd-.

A GNU/Linux se le encuentra normalmente en forma de compendios conocidos como distribuciones o distros, a las cuales se les ha adicionado selecciones de aplicaciones y programas para descargar e instalar las mismas. El propósito de una distribución es ofrecer GNU/Linux como un producto final que el usuario pueda instalar, cumpliendo con las necesidades de un grupo de usuarios o bien del público en general. Algunas de ellas son: Ubuntu, CentOS, Debian, Linux Mint, Arch Linux, Fedora, Red Hat, Oracle, Zorin, MX Linux, Parrot, Manjaro, Elementary, etc.

Algunas de ellas son especialmente conocidas por su uso en servidores de internet y supercomputadoras -donde GNU/Linux tiene la cuota más importante del mercado. Según el informe de International Data Corporation (IDC), GNU/Linux es utilizado por los más poderosos 500 sistemas de super-

[27] GNU -es un acrónimo recursivo de «GNU no es UNIX»- es un sistema operativo de Software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU además de Software libre publicado por terceras partes con distintas licencias que conforman una distribución.

cómputo de alto desempeño del mundo[28]-, en cuanto a teléfonos inteligentes y tabletas tiene una cuota de 86% y con menor participación, el sistema GNU/Linux también se usa en el segmento de las computadoras de escritorio, portátiles, computadoras de bolsillo, sistemas embebidos, videoconsolas y otros dispositivos.

Creación El proyecto GNU, iniciado en 1983 por Richard Stallman, tiene el objetivo de crear un «sistema de Software compatible con Unix compuesto enteramente de Software libre». El trabajo comenzó en el año 1984. Más tarde, en 1985, Stallman fundó la Free Software Foundation para financiar el desarrollo de GNU, y escribió la Licencia Pública General de GNU en 1989. A principios de la década de 1990, muchos de los programas que se requieren en un sistema operativo -como bibliotecas, compiladores, editores de texto, el Shell Unix, y un sistema de ventanas- ya se encontraban en operación. Sin embargo otros elementos como los controladores de dispositivos y los servicios estaban incompletos.

Linus Torvalds ha declarado que si el núcleo de GNU hubiera estado disponible en el momento (1991), no se habría decidido a escribir su propio núcleo. Aunque no fue liberado hasta 1992 debido a complicaciones legales, el desarrollo de BSD -de los cuales NetBSD, OpenBSD y FreeBSD descienden- es anterior al de Linux. Torvalds también ha declarado que si BSD hubiera estado disponible en ese momento, probablemente no habría creado Linux.

En 1991 Torvalds asistía a la Universidad de Helsinki. Usuario de MINIX y de los programas provenientes de GNU, se mostraba interesado por los sistemas operativos. Comenzó a trabajar en su propio núcleo en ese año, frustrado por la concesión de licencias que utilizaba MINIX, que en ese momento se limitaba a uso educativo.

El núcleo Linux maduró hasta superar a los otros núcleos en desarrollo. Las aplicaciones GNU también reemplazaron todos los componentes de MINIX, porque era ventajoso utilizar el código libre del proyecto GNU con el nuevo sistema operativo. El código GNU con licencia bajo la GPL puede ser reutilizado en otros programas de computadora, siempre y cuando también se liberen bajo la misma licencia o una licencia compatible. Torvalds inició un cambio de su licencia original, que prohibía la redistribución comercial a la GPL. Los desarrolladores de ambas partes trabajaron para integrar com-

[28] Top500.org informó, en su lista de noviembre de 2017 -y así ha continuado hasta ahora-, que las 500 supercomputadoras más potentes del mundo utilizan Linux.

ponentes de GNU con el núcleo Linux, consiguiendo un sistema operativo completamente funcional.

Para darnos una idea del frenético crecimiento del Kernel de Linux, por ejemplo, en la versión 4.10 se añadieron 632,782 líneas de código nuevo y en el Kernel 4.12 se añadieron más 1.2 millones de líneas de código nuevas, teniendo un total de 24,170,860 líneas de código. El número de desarrolladores involucrados fue de 1821 colaboradores y 220 empleados hicieron un promedio de 231 cambios por día, casi 10 cambios por hora, diariamente se añadieron casi 20 mil líneas de código, y casi 800 líneas por hora en dicha versión.

Hay que precisar que, si bien el código alojado en el repositorio del Kernel es cuantioso, sólo una pequeña parte del mismo afectará a nuestras propias instalaciones de GNU/Linux, pues gran parte del código fuente es específico para cada una de las (múltiples) arquitecturas de Hardware compatibles con Linux.

De hecho, a principios de 2018, Greg Kroah-Hartman (responsable de mantenimiento del código), afirmó que "un portátil promedio usa alrededor de 2 millones de líneas de código del Kernel para funcionar correctamente", cuando en aquel momento, el Kernel completo ya contaba con 25 millones de líneas de código (que ya han aumentado a más de 28 millones en la versión 5.8).

GNU/Linux puede funcionar tanto en entorno gráfico como en modo consola. La consola es común en distribuciones para servidores, mientras que la interfaz gráfica está orientada al usuario final del hogar como empresarial. Así mismo, también existen los entornos de escritorio, que son un conjunto de programas conformado por ventanas, íconos y muchas aplicaciones que facilitan el uso de la computadora. Los entornos de escritorio más populares en GNU/Linux son: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, entre muchos otros.

¿Qué es lo que está llevando a la gente a probar distribuciones de GNU/Linux y a utilizarlas como sistema operativo principal en sus equipos de cómputo? A continuación, vamos a exponer una lista con las razones por las que deberías probar una distribución de GNU/Linux -ya que es una sabia elección- como sistema operativo principal en tu equipo de cómputo:

Software Libre y Código Abierto muchos usuarios de internet no conocen el significado principal del Software libre ni del código abierto. Software libre son esos programas que se automanifiestan, por parte de sus autores, que puede ser copiado, modificado y redistribuido con o sin cambios o mejoras. El concepto de código abierto, es el Software desarrollado y distribuido libremente. Tiene beneficios prácticos ya que si alguien tiene una idea o piensa que puede mejorar el código puede modificarlo sin problemas.

Seguridad no descubrimos el agua tibia diciendo que el sistema operativo de Microsoft es el más atacado por virus y Malware y además, se han descubierto varios virus para Mac OS, unos que llevan ocultos mucho tiempo. Pero con GNU/Linux eso no pasa, ya que es un sistema suficientemente seguro y que no tenemos muchos registros de ataques a esta plataforma.

Aunque hay compañías Linuxeras, como Oracle, Novell, Canonical, Red Hat o SUSE, donde el grueso de distribuciones y Software Linux está mantenido por usuarios y colectivos sin ánimo de lucro. A diferencia de Microsoft y Windows, detrás de Linux no es habitual encontrarnos con una empresa con intereses empresariales, de manera que es más fácil evitar problemas de tipo legal o violaciones de nuestra privacidad o seguridad por parte de quienes han programado esa aplicación o versión de GNU/Linux que usamos. Un ejemplo es la recopilación de datos de uso. A diferencia de los sistemas operativos comerciales, en GNU/Linux no es habitual toparse con este problema.

Es Gratis aunque Mac OS X también es gratuito, está pensado para funcionar solamente en equipos de cómputo Apple. En cuanto a Windows, a pesar de la tendencia, sigue siendo de pago, a pesar de las muchas ofertas que hizo para cambiar de Windows 7 a Windows 10.

Si adquieres una computadora nueva con Windows, el precio incluye la licencia de compra. Por otro lado, todo el mundo sabe que los sistemas operativos de GNU/Linux son totalmente gratuitos y puedes instalarlos en cualquier equipo de cómputo. Las distribuciones más populares puedes descargarlas desde sus páginas oficiales e instalarlas las veces que quieras y en el número de equipos de cómputo que necesites. Además, no tendremos que pagar por utilizar el Software, sin embargo, podremos donar lo que nos plazca al proyecto para que sigan mejorándolo.

Fácil de Utilizar muchos de nosotros hemos utilizado un sistema operativo basado en GNU/Linux y no lo sabíamos. Aeropuertos, estaciones de tren, sistemas de gestión empresarial y ahora en el espacio con SpaceX, etc. Muchos de estos sistemas están basados en GNU/Linux.

Una de las barreras que durante años ha evitado a muchos usar Linux es su complejidad. O al menos lo era cuando la mayoría de tareas debías hacerlas desde la línea de comandos.

En la actualidad, distribuciones GNU/Linux como Ubuntu, Mint, Manjaro, Debian u OpenSUSE ofrecen una interfaz similar a Windows y con todas las herramientas y aplicaciones necesarias para empezar a disfrutar desde el primer día.

Si necesitas un nuevo Software, la mayoría de distribuciones cuentan con su propia tienda de aplicaciones o herramienta de gestión de aplicaciones. Todo está pensado para que cualquiera pueda manejarse sin problemas.

Está claro que existen versiones de GNU/Linux complejas, pero están enfocadas a un público muy concreto. Las distribuciones domésticas cumplen con creces con los requisitos de usuarios amateurs o recién llegados.

Versatilidad configurar un sistema a nuestro gusto, en Windows o en Mac OS X, es algo realmente difícil, pero con los sistemas operativos basados en GNU/Linux se puede tener un sistema operativo totalmente único y totalmente personalizable.

La naturaleza de GNU/Linux y su filosofía de código abierto y libre hace posible que contemos con cientos de versiones diferentes. Esto implica que podamos elegir una versión de GNU/Linux, o distribución, en función de para qué la queremos. ¿Para educación? ¿Para niños? ¿Para uso doméstico? ¿Para gestión de redes? ¿Para temas de seguridad? ¿Para reciclar un PC antiguo? Incluso las hay para arreglar problemas de Windows.

Esta variedad significa que no sólo podemos emplear GNU/Linux en una computadora doméstica. Los ejemplos más claros son Raspberry Pi, Jetson Nano, Pine64 y Arduino[29], son soluciones baratas y diminutas para montar tu propia computadora personal, tu centro multimedia o cualquier artilugio electrónico que desees diseñar. Y para hacerlo funcionar, cuentas con varias distribuciones Linux enfocadas a dicho Hardware.

[29] Son ordenadores del tamaño de una tarjeta de crédito que se conectan a un televisor, un teclado y ratón. Es una placa que soporta varios componentes necesarios en un ordenador común y cuyo precio inicial es de 15 dólares.

Usar GNU/Linux significa que puedes cambiar cualquier elemento de tu sistema operativo. Me refiero a ir más allá de los programas y aplicaciones por defecto. GNU/Linux cuenta con diferentes escritorios y gestores de ventanas, de manera que podemos elegir el que queramos, algo que permiten muchas distribuciones GNU/Linux. Mientras que Windows cuenta con un escritorio por defecto, en GNU/Linux podemos elegir entre: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc. En la variedad está el gusto.

Además, cualquier configuración o elemento del sistema operativo es susceptible de ser alterado[30]. La única limitación es que seamos capaces o tengamos los conocimientos adecuados. Pero siempre podemos encontrar en internet un tutorial donde nos explique cómo hacerlo.

Existen distribuciones de Linux de tamaño muy reducido, por ejemplo: BasicLinux ocupa 2.8 MB, requiere un procesador 386 y 3 MB de RAM y cuenta con el escritorio gráfico JWM, Nanolinux ocupa 14 MB, utiliza SLWM como escritorio y cuenta con navegador, procesador de texto, hoja de cálculo, cliente IRC, etc.

Actualizaciones del Sistema Operativo hablando de actualizaciones, sus aplicaciones se actualizan prácticamente al día, en cuanto el desarrollador lanza dicha actualización. Por lo que siempre podemos tener nuestros programas y aplicaciones actualizadas.

Además para los usuarios que así lo requieran existen versiones de soporte a largo plazo (Long-Term Support , LTS) normalmente se asocia con una aplicación o un sistema operativo para el que obtendremos seguridad, mantenimiento y (a veces) actualizaciones de funciones durante un período de tiempo más largo.

Las versiones LTS se consideran las versiones más estables que se someten a pruebas exhaustivas y en su mayoría incluyen años de mejoras en el camino.

[30] BlendOS este prometedor sistema operativo, introduce muchas novedades, empezando porque ahora soporta distintas distribuciones: Arch (el principal), AlmaLinux, Crystal Linux, Debian, Fedora, Kali Linux, Neurodebian Bookworm, Rocky Linux y Ubuntu.

Además de estar disponible en siete entornos gráficos, y que se puede cambiar entre ellos con un sencillo comando. Los entornos en los que está son GNOME, KDE (Plasma), Cinnamon, Xfce, LXQt, MATE y Deepin. El comando para ir cambiando entre los escritorios disponibles es: *sudo system track*. Esta distribución es inmutable, por lo que es difícil que subir de versión estropee algo. Básicamente son imágenes completas a las que se le pueden hacer pequeños retoques, como instalar nuevo software. Pero casi todo va por contenedores.

Es importante tener en cuenta que una versión de Software LTS no implica necesariamente actualizaciones de funciones a menos que haya una versión más reciente de LTS. Sin embargo, obtendrá las correcciones de errores y las correcciones de seguridad necesarias en las actualizaciones de una versión de Soporte a largo plazo.

Se recomienda una versión LTS para consumidores, negocios y empresas listos para la producción porque obtiene años de soporte de Software y sin cambios que rompan el sistema con las actualizaciones. Si observamos una versión que no es LTS para cualquier Software, generalmente es la versión más avanzada con nuevas funciones y un período corto de soporte (por ejemplo, 6-9 meses) en comparación con 3-5 años de soporte en un LTS.

Tiendas de Aplicaciones lo mejor de las distribuciones de GNU/Linux es que tienen una característica en común, sus tiendas de aplicaciones. Ya que vamos a poder instalar cualquier tipo de programa que necesitemos con un Click. Recordamos que esto es algo que Windows está intentando con su propia tienda de aplicaciones, pero no están teniendo muy buenos resultados.

Compatibilidad muchos han experimentado problemas a la hora de actualizar sus sistemas operativos con los programas que tenían instalados. Pero eso con GNU/Linux, no pasa, ya que todas sus actualizaciones tienen retrocompatibilidad a largo plazo dentro de su distribución.

Hoy en día la mayoría de aplicaciones y servicios Online cuentan con versión compatible para cualquier sistema operativo. Siendo más fácil crear una aplicación multiplataforma, por lo que GNU/Linux cuenta con un catálogo de Software que poco o nada tiene que envidiar a Windows o Mac OS X.

En el catálogo destacan las aplicaciones gratuitas y de código abierto, pero también surgen proyectos comerciales, y en la lista se incluyen los juegos, cada vez más presentes en GNU/Linux.

Seguramente hay algún Software no disponible en GNU/Linux, pero es más que probable que encontremos una alternativa o, en su defecto, que podamos ejecutarlo mediante Wine o empleando máquinas virtuales como KVM/QEMU o VirtualBox.

En cuanto al Hardware, la comunidad GNU/Linux ha avanzado mucho en la creación de controladores o Drivers para emplear cualquier dispositivo o componente en GNU/Linux. Podemos encontrarnos con alguna excepción, pero la mayoría de dispositivos cuentan con un controlador compatible por

defecto.

Está en Todas Partes GNU/Linux está presente en la infraestructura de grandes empresas como Amazon, Facebook, Netflix, NASA, SpaceX, el gran colisionador de hadrones o IBM y en el año 2021 llegó a Marte en el sistema operativo del helicóptero que acompaña al rover Perseverance, etc. A nivel de usuario, muchos dispositivos emplean este sistema operativo, bien en alguna de sus versiones o a través de Android, que salvando las distancias, todavía conserva gran parte de su origen Linuxero. Por otro lado, las quinientas principales supercomputadoras emplean Linux como sistema operativo, ya que permite trabajar en todo tipo de entornos y situaciones.

Las grandes empresas de internet hace años que vieron en GNU/Linux una gran oportunidad, y si bien a nivel usuario doméstico no está tan extendido, nunca había sido tan fácil dar el paso. Para hacernos una idea, sólo hay que ver la lista de empresas que apoyan a GNU/Linux a través de The Linux Foundation. Una de las más recientes, la propia Microsoft.

La Comunidad GNU/Linux finalmente, hay que hablar de la fabulosa comunidad de GNU/Linux. Podemos preguntar lo que queramos en sus foros, cambiar el código, enviar tus programas, sin problemas. ¿Trabas en la configuración? Te lo solucionan sin preocupación, ¿consejos sobre Software? Hay cientos de hilos con soluciones. Y nosotros ponemos nuestro granito de arena con este trabajo.

Programas de Windows y macOS en Linux A medida que va pasando el tiempo, las diferencias entre los sistemas operativos se van volviendo irrelevantes. Máquinas virtuales, contenedores y otras tecnologías permiten que podamos utilizar cada día más títulos de nuestros programas preferidos aunque no tenga versión para nuestro sistema operativo.

Wine, la herramienta que actúa como un intérprete entre el núcleo Linux y las aplicaciones Windows ya lleva mucho tiempo entre nosotros. Desde hace poco tiempo, también tenemos una herramienta para los programas de macOS.

Wine para Aplicaciones de Windows Nació inicialmente como un proyecto que buscaba crear un emulador de Windows. Su acrónimo era inicialmente «WINdows Emulator», aunque viendo su evolución, y la forma de funcionar,

este acrónimo fue actualizado por «Wine Is Not an Emulator». Y es que en realidad no es un emulador, sino que este programa está formado por un cargador de programas binarios junto a un conjunto de herramientas de desarrollo que permiten portar en tiempo real el código de las aplicaciones de Windows a Unix. Además, trae por defecto una gran cantidad de bibliotecas y librerías de manera que no tengamos problemas de dependencias.

Principales Características este programa es capaz de ejecutar sin problemas cualquier programa diseñado para cualquier versión de Windows, desde la 3.x hasta Windows 10. Eso sí, solo es compatible con programas Win32 (tanto de 32 bits como de 64 bits), por lo que no vamos a poder ejecutar las apps UWP de la Microsoft Store, al menos por ahora.

Entre toda la variedad de librerías, bibliotecas y recursos, podemos encontrarnos con prácticamente todas las bibliotecas de interrupciones para programas, lo que permite hacer llamadas INT en tiempo real. De esta manera, los programas no saben que se están ejecutando en un sistema operativo que no es Windows, simplemente se ejecutan. Y lo hacen igual que en él. Si algún programa, o juego, tiene dependencias especiales (por ejemplo, una DLL concreta) podemos añadirla fácilmente a Wine. Todas las librerías se encuentran dentro del directorio «~/.wine/drive_c/windows/system32», que equivale al directorio System32 de Windows.

Por supuesto, Wine tiene soporte para una gran cantidad de recursos gráficos. Los programas se pueden dibujar tanto en una interfaz gráfica X11 (el escritorio) como desde cualquier terminal X. Es compatible con las tecnologías OpenGL, DirectX y cuenta con soporte total para GDI (y parcial para GDI32). También permite y gestionar varias ventanas a la vez (del mismo programa, o de diferentes) y es compatible con los temas msstyle de Windows.

También es compatible con los controladores de sonido de Windows, y tiene acceso a los puertos del PC, al Winsock TCP/IP y hasta a los escáneres.

¿Qué Programas y Juegos Puedo Ejecutar con Wine? por desgracia, a pesar de tener una gran compatibilidad, Wine no es capaz de ejecutar el 100% de los programas y juegos de Windows en Linux. Y algunos, aunque se pueden ejecutar, no funcionan del todo bien. Para saber si un programa se puede ejecutar, o no, en Linux, podemos buscarlo en la red del proyecto. Allí vamos a encontrar una gran base de datos que nos va a per-

mitir saber si un programa funciona va a funcionar, si no lo hace, o qué tal lo hace.

Además de poder buscar manualmente cualquier programa o juego, también vamos a encontrar una lista con los Top-10 que mejor funcionan. Los juegos «Platino» son los que funcionan de manera idéntica en Windows que en Linux, los «Oro» los que funcionan bien, pero requieren de alguna configuración especial y los «Plata», los que funcionan, pero tienen pequeños problemas. Los programas o juegos «Bronce» o «Basura» son los que no funcionan.

Saca Todo el Partido a Wine con Estos Programas Wine, al final, es la parte más importante para poder usar programas de Windows en Linux. Sin embargo, su configuración, sobre todo para los programas que no tienen una clasificación de platino, puede ser algo tediosa. Por suerte, existen programas que, aunque se basan igualmente en Wine, nos ayudan a configurar cada uno de estos programas de manera automática para que nosotros no tengamos que hacer nada más.

La instalación y configuración de los programas y juegos de Windows para usarlos en Linux es lo peor. Si no tenemos muchos conocimientos podemos perder mucho tiempo y, además, no conseguiremos que todo funcione del todo bien. Aquí es donde entra en juego *PlayOnLinux*. Este programa, gratuito y de código abierto, busca ayudarnos con la instalación y configuración de los programas y juegos para hacerlos funcionar en este sistema operativo.

PlayOnLinux nos ofrece una completa base de datos de programas con sus correspondientes configuraciones óptimas de manera que nosotros solo tengamos que seleccionar el programa que queremos, cargarle su instalador y dejar que este complete el proceso de puesta en marcha. Nada más. Cuando acabe la instalación ya podremos abrir el programa o juego y empezar a usarlo.

Podemos descargar esta herramienta de forma totalmente gratuita desde su página Web, o desde una termina:

```
# apt install playonlinux
```

Descargar e Instalar Wine hay muchas formas de instalar Wine en Linux. Sus desarrolladores tienen binarios específicos para cada distribución, así como unos completos repositorios desde los que vamos a poder descargar y actualizar el programa desde terminal:

```
# apt install wine
```

WINE no es una aplicación que se inicia por sí sola. Es un backend que se invoca cuando se inicia una aplicación de Windows. Lo más probable es que su primera interacción con WINE ocurra cuando inicie el instalador de una aplicación de Windows.

Ejecutar e Instalar Programas Windows una vez instalado, Wine se ejecutará al hacer doble clic sobre cualquier archivo .EXE. Además, te permitirá instalar programas, como si estuvieras en Windows y pondrá los accesos directos en el menú principal bajo la categoría «Wine».

A pesar de lo que mucha gente cree, Wine sirve no sólo para correr aplicaciones «sencillas» de Windows, sino incluso juegos complejos. Es más, está demostrado que terribles juegazos como Sim 3, Half Life 2, Command & Conquer 3, Star Wars: Jedi Knight, o importantes suites como Microsoft Office funcionan a la perfección.

Darling para Aplicaciones de macOS cumple una función similar a la de Wine con los programas de Windows, solo que no tiene ningún complejo en definirse como un emulador. Lo que hace es actuar como un traductor que permite ejecutar los programas de macOS usando los recursos de Linux. El nombre Darling (Querida) es la primera parte del nombre del núcleo de macOS (Darwin) y las primeras 3 letras de Linux. Supongo que la G final es para construir una palabra fácil de memorizar.

Hay que decir que a los desarrolladores de Darling la cosa les resulta más fácil que a los de Wine. No tienen que hacer ingeniería inversa ni reinventar nada dado que se basan en las partes de Darwin que están bajo licencias abiertas. El propio Darling se distribuye bajo la licencia GPL.

Iniciando Darling el programa no tiene interfaz gráfica. Lo ponemos en marcha desde la terminal con el comando:

```
$ darling shell
```

Al escribirlo, Darling creará un directorio raíz virtual o se conectará con uno existente. Además cargará los módulos del núcleo y construirá el sistema de archivos virtual donde ejecutaremos los programas.

Desde la línea de comandos podemos acceder a dos tipos de sistemas de archivos: el tradicional de macOS que incluye los directorios de nivel superior como */Applications*, */Users* y */System* entre otros. Por otro lado, el del sistema operativo anfitrión lo hallamos en una partición denominada */Volumes/SystemRoot*

Podemos verificar el núcleo con el siguiente comando:

```
$ uname
```

y averiguar la versión de macOS con:

```
$ sw_vers
```

salimos de la terminal con

```
$ exit
```

y apagamos el contenedor con:

```
$ darling shutdown
```

Instalación de Programas Si estás usando Linux en arranque dual con macOS y quieres ejecutar alguno de los programas que tienes instalado en la partición de Mac, puedes hacerlo con el comando:

```
$ /Volumes/SystemRoot/run/media/usuario/Macintosh HD
/Applications/nombre_app.app)
```

Muchos programas para macOS se distribuyen en formato .dmg. Para instalarlos en Darling hacemos:

```
Darling [~]$ hdiutil attach Downloads/aplicación.dmg /Vol-
umes/aplicacion
Darling [~]$ cp -r /Volumes/aplicación/aplicación.app /Ap-
plications/
```

En el caso de aplicaciones almacenadas en archivos comprimidos, lo descomprimimos y copiamos en la carpeta */Applications*. Lo mismo con aplicaciones previamente descargadas de la tienda de aplicaciones.

Por último nos quedan las aplicaciones *.pkg*, el formato de paquete nativo de macOS. Este formato implica ejecutar Scripts durante la instalación. Para poder usarlos debemos hacer:

Darling [~]$ installer -pkg aplicación.pkg -target /

Podemos desinstalar los programas con:

Darling [~]$ uninstaller nombre_del_paquete

Debemos entender que si bien Darling funciona muy bien con aplicaciones para la línea de comandos, solo tiene funcionalidades muy limitadas para las que necesitan una interfaz gráfica.

Instalación de Darling Si utilizas Debian o derivados, la instalación de Darling no tiene mayor problema. Solo tienes que escribir los comandos:

```
# apt install gdebi
# gdebi darling-dkms_X.X.X.testing_amd64.deb
# gdebi darling_X.X.X.testing_amd64.deb
```

reemplaza las X por el número de versión de los paquetes que descargarás o bien se puede descargar los archivos fuentes del proyecto para su compilación e instalación.

Aprender a Usar Linux Existen diversos sitios Web que están enfocados a explorar detalladamente cada distribución actual o antigua, a un nivel técnico acompañado de grandes y útiles análisis técnicos sobre los mismos, lo que facilita el aprendizaje puntual sobre qué distribución usar o empezar a usar sin tanta incertidumbre, algunos de estos lugares son:

- ArchiveOS https://archiveos.org
- Distro Chooser https://distrochooser.de/es/
- Distro Watch https://distrowatch.com
- Linux Distribution List https://lwn.net/Distributions/

¿Qué otros sabores de GNU/Linux hay?

https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

Existen distintas distribuciones de GNU/Linux[31] para instalar, una de las más ampliamente usadas es Debian GNU/Linux[32] y sus derivadas como Ubuntu. La comunidad de GNU/Linux te apoya para obtener, instalar y que de una vez por todas puedas usar GNU/Linux en tu computadora.

Puedes conocer y descargar las diferentes distribuciones desde:

https://es.wikipedia.org/wiki/Anexo:Distribuciones_Linux

https://en.wikipedia.org/wiki/List_of_Linux_distributions

y ver cuál es la que más te conviene:

https://en.wikipedia.org/wiki/Comparison_of_Linux_distributions

o probar alguna versión Live[33]:

https://livecdlist.com/

también las puedes correr como máquina virtual para VirtualBox:

https://www.osboxes.org/

o máquina virtual para QEMU/KVM:

https://docs.openstack.org/image-guide/obtain-images.html

https://github.com/palmercluff/qemu-images

https://bierbaumer.net/qemu/

[31] Una lista de las distribuciones de Linux y su árbol de vida puede verse en la página Web http://futurist.se/gldt/

[32] Algunas de las razones para instalar GNU/Linux Debian están detalladas en su página Web https://www.debian.org/intro/why_debian.es.html

[33] Linux es uno de los sistemas operativos pioneros en ejecutar de forma autónoma o sin instalar en la computadora, existen diferentes distribuciones Live -descargables para formato CD, DVD, USB- de sistemas operativos y múltiples aplicaciones almacenados en un medio extraíble, que pueden ejecutarse directamente en una computadora, estos se descargan de la Web generalmente en formato ISO.

por otro lado, existen diferentes servicios Web que permiten instalar, configurar y usar cientos de sistemas operativos Linux y Unix desde el navegador, una muestra de estos proyectos son:

Distrotest https://distrotest.net
JSLinux https://bellard.org/jslinux
OnWorks https://www.onworks.net

Ahora, Windows 10 Build 2020 con WSL2[34] (Windows Subsystem for Linux), tiene su propio Kernel de Linux que permite instalar de manera casi nativa diversas distribuciones de GNU/Linux permitiendo tener lo mejor de ambos mundos en un mismo equipo.

En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a usar, administrar y optimizar cada uno de los distintos aspectos de Linux, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Sistemas operativos

2.5 Android

Android (véase [6]) es un sistema operativo basado en el núcleo Linux (véase apéndice 12.2). Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Android fue presentado en 2007 junto a la fundación del Open Handset Alliance (un consorcio de compañías de Hardware, Software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Android es el sistema operativo móvil (SmartPhone y tabletas) más utilizado del mundo, con una cuota de mercado del 86% al año 2020, muy por encima del 13.9% de iOS.

El éxito del sistema operativo lo ha convertido en objeto de litigios sobre patentes en el marco de las llamadas guerras de patentes entre las empresas de teléfonos inteligentes. Según los documentos secretos filtrados en 2013 y 2014,

[34] https://docs.microsoft.com/en-us/windows/wsl/install-win10

el sistema operativo es uno de los objetivos de las agencias de inteligencia internacionales.

La versión básica de Android es conocida como Android Open Source Project (AOSP). El 25 de junio de 2014 en la Conferencia de Desarrolladores Google I/O, Google mostró una evolución de la marca Android, con el fin de unificar tanto el Hardware como el Software y ampliar mercados. El 17 de mayo de 2017, se presentó Android Go. Una versión más ligera del sistema operativo para ayudar a que la mitad del mundo sin Smartphone consiga uno en menos de cinco años. Incluye versiones especiales de sus aplicaciones donde el consumo de datos se reduce al máximo.

Arquitectura del Sistema Android los componentes principales del sistema operativo de Android[35]:

Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a las mismas API del entorno de trabajo usadas por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del Framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android. Algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

[35] Android tiene la base de Linux, por ello en cualquier dispositivo que soporte dicho sistema operativo es posible instalar una aplicación para acceder a la terminal de línea de comandos -por ejemplo ConnectBot-, y en ella podemos correr los comandos de BASH como en un sistema GNU/Linux.

Runtime de Android: Android incluye un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android ejecuta su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede ejecutar múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato de ejecutable Dalvik (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida DX. Desde la versión 5.0 utiliza el ART, que se compila totalmente al momento de instalación de la aplicación.

Personalización muchos conocen a Android como el sistema operativo móvil más personalizable. Pero para los que no lo saben, recordamos que está basado en el núcleo de Linux y que muchos desarrolladores están queriendo llevar Android a un sistema operativo de escritorio.

Núcleo Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el Hardware y el resto del Software.

Android sobre KVM (MicroDroid) En enero de 2021 se anunció por parte de Google que trabajan en MicroDroid, una versión minimalista de Android para máquinas virtuales sobre KVM. No es la primera alternativa a Android, nos encontramos con las capas de personalización Android Go, AOSP o las imágenes GSI. En el caso de MicroDroid, estaríamos ante una limitada imagen de Linux basada en Android, para este proyecto, Google está trabajando en adaptar la máquina virtual de Chrome OS (crosvm), que ya se utiliza para ejecutar aplicaciones de Linux en Chrome OS. De esta forma con MicroDroid podría ejecutar pequeñas máquinas virtuales junto a Android, posiblemente para aplicaciones y uso relacionado con DRM, esta modificación constaría con el mínimo de componentes para iniciar el sistema permitiendo aislar datos entre aplicaciones y sistemas operativos en el mismo dispositivo, así como cambiar instantáneamente entre sistemas operativos.

2.6 Chromebook y Chrome OS

Para entender la razón de ser de los Chromebooks, primero tenemos que entender qué es Chrome OS. Se trata de un sistema operativo creado por Google y diferente a Android. Está basado en el Kernel de Linux, y utiliza Chrome como su interfaz de usuario principal. Esto quiere decir que su aspecto es prácticamente idéntico al de Chrome, pero con algunos añadidos como una barra de tareas, un explorador de archivos y otros elementos presentes en cualquier sistema operativo.

Fue anunciado a mediados del 2009 como un intento de crear un sistema basado en la nube y en aplicaciones Web. Esto hacía que, cuando se estaba conectado a internet se pudieran hacer muchas cosas gracias a herramientas como Google Drive o las aplicaciones de la Chrome Web Store, pero que cuando dejaba de tener internet se limitaba mucho sus funciones.

En cualquier caso, y pese a lo limitado que era en sus primeros años, poco a poco Google lo ha hecho evolucionar. Primero se empezaron a añadir opciones a las aplicaciones de Google para poderse utilizar sin conexión, algo que también benefició a los usuarios que usaran Chrome en otros sistemas operativos.

Pero la evolución más grande fue llegando después. El primer gran paso fue el anuncio de la compatibilidad para ejecutar aplicaciones de Android, y se fue implementando directamente la tienda de aplicaciones Google Play de Android para hacer que la experiencia de instalarlas fuera tan nativa como en Android. Aun así, hay que decir que la llegada de Android a Chrome OS ha sido lenta, y han tardado algunos años en hacer que todo vaya funcionando como debería.

Y a mediados de 2018 se anunció que Google Chrome también podrá utilizar aplicaciones creadas para los sistemas GNU/Linux. Con ello, el catálogo de aplicaciones diseñadas para funcionar sin conexión se multiplica beneficiando a la comunidad de desarrolladores libres, aunque también es de esperar que tarde algunos años en estar todo perfectamente integrado, ya que todavía se están lanzando poco a poco mejoras.

Chrome OS es hoy en día un sistema operativo completo. Tiene lo básico, aplicaciones nativas y compatibilidad con Android, que se une al reproductor de medios, gestor de archivos, configuración de impresoras, etcétera. Además, al igual que el navegador, Chrome OS tiene también una versión libre llamada Chromium OS, que pese a no tener la tecnología nativa de Google sirve para que la comunidad de desarrolladores independientes pueda ayudar a

mejorarlo.

Ahora bien, los Chromebook son equipos de cómputo personales que utilizan como sistema operativo Chrome OS, desarrollado por Google y que, a diferencia de Windows, OS X y Linux, están pensados para utilizarse permanentemente conectados a internet, ya que se basan casi completamente en la nube.

Chromebook Apps también se incluye un reproductor multimedia, y todo se sincroniza permanentemente en la nube. Por ello, si pretendemos utilizar un Chromebook sin conexión a internet, su funcionalidad es más limitada que la de otros equipos de cómputo. De hecho, las aplicaciones se instalan a través de Chrome Web Store, la tienda de aplicaciones integrada en Google Chrome, con lo que algunas de las herramientas más habituales (como Office o Skype, por ejemplo) tendrían que verse reemplazadas por Google Drive y Google Hangouts, aplicaciones nativas de Google.

Chrome Web Store no obstante, también se pueden utilizar de forma local sin recurrir a la red, ya que muchos de los servicios de Google disponen de un modo sin conexión que, una vez volvemos a disponer de internet, se sincronizaran sin problemas.

¿Cómo es un Chromebook? en un Chromebook podemos utilizar dispositivos USB sin problemas, como memorias y discos externos, Webcams, teclados y ratones, y por lo general suelen venir con una cantidad de almacenamiento inferior a lo que estamos acostumbrados (ya que lo que se pretende es que todo esté en la nube, y no en nuestro disco duro local). De hecho, al adquirir uno se nos obsequia con 100 GBytes de espacio en Google Drive.

Igualmente, su precio suele ser bastante asequible (desde 179 dólares o 130 euros) y no requieren de un Hardware potente para funcionar, siendo la ligereza de recursos una de sus mayores bondades. Por su parte, los equipos de cómputo portátiles con Chrome OS son lo que llamamos Chromebook, mientras que si preferimos el formato Mini PC, estaremos ante un Chromebox.

El inicio del sistema es prácticamente instantáneo y todo está listo para funcionar en cuestión de segundos, y dadas sus características, un Chromebook es un equipo ideal para navegar por internet ante todo.

Se accede desde la barra de herramientas en la parte inferior de la pantalla a las aplicaciones que tengamos instaladas, que en realidad se trata de un atajo a las apps que tengamos instaladas en Google Chrome.

Chromebook Integración por supuesto, los Chromebook también son multiusuario, con la ventaja de que con simplemente iniciar sesión con otra cuenta de Gmail todo estará tal y como si lo hubiésemos configurado con ella (aplicaciones, servicios, historial y demás), y por este mismo motivo se complementan a la perfección con otros dispositivos (ya sean equipos de cómputo, Smartphones o Tablets) en los que utilicemos los servicios de Google, gracias a la sincronización en la nube.

Además, los Chromebook también presumen de no necesitar antivirus, pues al almacenarse todo en la nube la seguridad está integrada por defecto y corre por parte de Google.

Microsoft en un Chromebook En el 2020 las empresas Parallels[36] y Google llegaron a un acuerdo para ofrecer a los usuarios la posibilidad de ejecutar aplicaciones Windows en Chrome OS. Ellas aseguran que en Chrome OS la integración será completa: las aplicaciones se ejecutarán cada una en su propia ventana, como las nativas, y no dentro de un Windows virtualizado.

Aunque ninguna de las dos compañías ha ofrecido aún una lista de aplicaciones compatibles con esta función que será lanzada en el 2021, John Solomon (vicepresidente de Chrome OS) ha afirmado que Microsoft Office será una de ellas.

El problema es que, por ahora, estas nuevas funcionalidades no estarán disponibles para todos los usuarios de Chrome OS, sino únicamente para los de Chrome OS Enterprise, la versión empresarial del mismo.

Nota: en últimas fechas han aparecido proyectos que permiten instalar diversas distribuciones de GNU/Linux en los Chromebook, esto es debido a que Google deja de dar soporte a sus equipos después de algunos años de que salieron al mercado, pese a que el equipo es totalmente funcional.

Chrome OS Flex En febrero del 2022, Google anunció su nueva versión de Chrome OS para equipos de cómputo PCs y Macs, la propuesta es Chrome

[36] Empresa (propiedad de Corel desde hace un año) desarrolladora del Software homónimo de virtualización que es especialmente popular entre los usuarios de Mac.

OS Flex y cuya descarga es totalmente gratuita, tiene como propósito atender las necesidades de escuelas y empresas rehusando equipo (procesador Intel o AMD 64 bits, 4 GB RAM, 16 GB de almacenamiento, etc). Esta versión tiene la misma interfaz gráfica y herramientas básicas que se encuentran en Chrome OS; entre ellas el navegador Chrome, se ofrece soporte para sincronización de ajustes y marcadores. Además, si nuestro equipo cumple con las especificaciones básicas, tendremos a nuestra disposición a Google Assistant e integraciones diversas con dispositivos Android.

Es de notar que Chrome OS Flex no tiene soporte para la Play Store o para las aplicaciones de Android y al parecer no hay intención de añadir esta compatibilidad. Tampoco se puede ejecutar Windows en una máquina virtual de Parallels Desktop.

Se puede descargar Chrome OS Flex para crear una unidad USB booteable y la instalación reemplazará al sistema operativo del equipo donde se instale (Mac, Windows o Linux). Pero no está optimizado para sacar provecho de todos los puertos, sensores o accesorios que pueden estar presentes en el equipo. Esto nos proporcionará una experiencia lo más cercana posible a Chrome OS sin comprar un Chromebook pese a sus limitaciones.

2.7 Otros Sistemas Operativos

Sistemas Operativos para PC

1. Fuchsia OS.- Es un sistema operativo versátil y adaptable esta basado en el microkernel Zircon en desarrollo por parte de Google, está disponible desde un repositorio de Git y está ya siendo usado en los Nest Hub, se espera su uso en la domótica que prepara Google como parte del internet de las cosas.

2. Dahlia OS.- Este sistema operativo combina lo mejor de GNU/Linux y Fuchsia OS es moderno, seguro, liviano y receptivo. Se mantiene minimalista al incluir solamente las aplicaciones necesarias, pero es posible agregar todos nuestros favoritos de otros sistemas operativos usando aplicaciones Containers y proporciona una tienda para aplicaciones Flutter de terceros.

3. KataOS.- Es un sistema operativo centrado en la seguridad y los sistemas embebidos que está construido casi enteramente con Rust. No emplea Linux ni Fuchsia, sino el micronúcleo seL4, el cual, según Google,

"pone la seguridad al frente y en el centro". Lo que se pretende con este sistema operativo es proporcionar "una plataforma segura verificable que protege la privacidad del usuario porque es lógicamente imposible que las aplicaciones violen las protecciones de seguridad del Hardware incluidas en el kernel, además de que los componentes del sistema son seguros de forma verificable".

4. ToaruOS.- Es un sistema operativo escrito desde cero y provisto con su propio Kernel, cargador de arranque, biblioteca C estándar, administrador de paquetes, componentes de espacio de usuario y una interfaz gráfica con un administrador de ventanas compuesto. Se inició como un proyecto de investigación en la Universidad de Illinois en el 2010 y a partir del 2012 es desarrollado por la comunidad interesada.

5. Essence, es un sistema operativo con su propio Kernel y escritorio construido desde cero por un entusiasta desde 2017 y se destaca por su original escritorio y pila de gráficos que permite dividir ventanas en pestañas, lo que permite trabajar en una ventana con varios programas a la vez y agrupar aplicaciones en ventanas según las tareas a resolver. El administrador de ventanas funciona al nivel del Kernel del sistema operativo y la interfaz se crea utilizando su propia biblioteca gráfica y un motor de Software vectorial que admite efectos animados complejos completamente vectoriales.

6. eComStation.- Seguro que muchos recuerdan el mítico OS/2 de IBM, sistema operativo que perdura con eComStation, derivado de este adaptado al Hardware moderno. A diferencia de otras alternativas de la lista, este no es gratuito y sus precios comienzan desde 145 dólares para la versión doméstica. Muchas aplicaciones libres como Firefox, OpenOffice o VLC han sido portadas a este sistema operativo.

7. Haiku.- BeOS fue un sistema operativo lanzado en el año 1991 con muy buenas intenciones a nivel de optimización e interfaz. Sin embargo, como les sucedió a muchos otros, terminó sucumbiendo en este complicado mercado. Su legado ha sido continuado por Haiku, un sistema de código abierto que lleva ya años en desarrollo.

8. ReactOS.- Es una alternativa a la arquitectura Windows NT de Microsoft totalmente abierta que no utiliza ningún tipo de código propietario. No obstante, es compatible con muchos de los controladores y

aplicaciones de Windows. Como punto negativo, su desarrollo no es tan rápido como muchos esperarían en un entorno tan cambiante como este.

9. FreeDOS.- Alternativa libre a DOS cuyo desarrollo sigue activo en estos momentos. Se trata de un entorno bastante estable, pero que carece de interfaz gráfica o multitarea. Es compatible a todos los niveles con MS-DOS y sus programas.

10. Solaris.- El sucesor de SunOS, de Sun Microsystems, empezó como una distribución propietaria de UNIX, pero en 2005 fue liberado como OpenSolaris. Más tarde, Oracle compró Sun y le cambió el nombre a Oracle Solaris.

11. Illumos.- Basado en Open Solaris, este proyecto nació por parte de algunos de los ingenieros originales del sistema. En realidad, busca ser una base para crear distribuciones de este sistema operativo. OpenIndiana es una de las más conocidas y utilizadas.

12. DexOS.- Un sistema operativo de 32 Bits escrito para la arquitectura x86 en lenguaje ensamblador. Está diseñado para programadores que desean tener acceso directo al Hardware (incluyendo CPU y gráficos) con un código bien comentado y documentado.

13. Syllable.- Sistema operativo nacido como fork de AtheOS, un clon de AmigaOS, aunque comparte mucho código con Linux. No tiene demasiada utilidad para los usuarios domésticos, aunque es compatible con arquitecturas x86.

14. AROS Research Operating System.- Es otro sistema que implementa en código abierto las APIs de AmigaOS, con cuyos ejecutables es compatible a nivel binario en procesadores de 68k, además de ser compatible a nivel de código con otras arquitecturas como x86 para la que se ofrece de manera nativa. Es portable y puede correr hospedado en Windows, Linux y FreeBSD.

15. MenuetOS.- Llamado también como MeOS, su característica más destacada es que está programado completamente en lenguaje ensamblador. Está diseñado para funcionar en equipos muy básicos aunque soporta

hasta 32 GigaBytes de RAM. Con decir que el sistema cabe en un disquete de 1.44 Megabytes, está dicho todo. Aún así se las arregla para incluir un escritorio gráfico y controladores para teclados, video, audio, USB o impresoras.

16. Visopsys.- Se trara de un sistema gratuito y libre bajo GPL que ha estado en desarrollo desde 1997, como hobby de un solo programador, Andy McLaughlin. Soporta arquitecturas x86, está escrito en C y ensamblador y no se basa en ningún sistema preexistente, si bien utiliza código del kernel Linux, ofrece herramientas comunes de GNU y parte de la interfaz gráfica de usuario como los iconos, resultaran familiares a los usuarios de KDE Plasma.

17. mOS.- Sistema operativo usado en centros de datos y para cómputo de alto rendimiento (High Performance Computing HPC), se basa en el Kernel de Linux pero tiene su propio núcleo ligero LWK, el Kernel gestiona un pequeño número de núcleos de la CPU para asegurarse la compatibilidad y el LWK Kernel gestiona el resto del sistema.

18. KolibriOS.- Es un pequeño sistema operativo poderoso y rápido para PCs. Solamente requiere unos pocos megas de espacio en disco y 8 MB de RAM para funcionar, además de incluir varias aplicaciones básicas.

19. SerenityOS es un sistema operativo Unix con aspecto de Windows de los 90s creado por un único programador como un proyecto terapéutico y está pensado para equipos X86 de escritorio.

20. BlendOS este prometedor sistema operativo Linux, introduce muchas novedades, empezando porque ahora soporta distintas distribuciones: Arch (el principal), AlmaLinux, Crystal Linux, Debian, Fedora, Kali Linux, Neurodebian Bookworm, Rocky Linux y Ubuntu. Además de estar disponible en siete entornos gráficos, y que se puede cambiar entre ellos con un sencillo comando. Los entornos en los que está son GNOME, KDE (Plasma), Cinnamon, Xfce, LXQt, MATE y Deepin. Esta distribución es inmutable, por lo que es difícil que subir de versión estropee algo. Básicamente son imágenes completas a las que se le pueden hacer pequeños retoques, como instalar nuevo Software, pero casi todo va por contenedores.

Sistemas Operativos para móviles

1. PinePhone.- Usa un sistema operativo basado en sistemas operativos de código abierto impulsado por la comunidad Linux, ha sido portado a 16 diferentes distribuciones de Linux y 7 diferentes interfaces gráficas de usuario como: Mobian, Manjaro con interfaz plasma, Ubuntu Touch, postmarketOS, LuneOS, Nemo Mobile, Maemo Leste, Tizen, entre otros. Además la compañía Pine64 es el segundo fabricante de teléfonos (después de OpenMoko) que ofrece el arranque desde una tarjeta microSD, que permite a los usuarios probar múltiples sistemas operativos, antes de instalarse en la memoria Flash interna.

2. HarmonyOS.- Sistema operativo desarrollado por Huawei para reemplazar a Android en sus equipos, es un sistema operativo similar a la idea de Fuchsia OS, con la idea que pueda instalarse tanto en un ordenador, como en un teléfono, tableta, relojes, como en un coche conectado, en donde todos estos dispositivos se conecten entre sí con una sola cuenta, dando así un paso hacia adelante en la utopía de la convergencia.

3. PostmarketOS.- Sistema operativo de Software libre y código abierto en desarrollo principalmente para teléfonos inteligentes y tabletas -es una idea genial, la persecución de tener Linux en los dispositivos Smartphone, como otra alternativa a los sistemas Android e iOS-, haciéndose las primeras pruebas en teléfonos que ya no tienen uso. Distribución basada en Alpine Linux. Puede usar diferentes interfaces de usuario, por ejemplo Plasma Mobile, Hildon, LuneOS UI, MATE, GNOME 3 y XFCE.

4. Plasma Mobile.- Es un sistema en fase de desarrollo por KDE que permite la convergencia con los usuarios de KDE para escritorio.

5. Lomiri.- Sistema operativo basado en Linux que soporta dos sabores: Ubuntu Touch y Manjaro. Ambos basados en Unity 8 que están en constante desarrollo.

6. Windows Phone.- Sistema operativo móvil desarrollado por Microsoft, como sucesor de Windows Mobile. A diferencia de su predecesor fue enfocado en el mercado de consumo en lugar del mercado empresarial.

7. Symbian OS.- Era un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson y otros, el objetivo de Symbian fue crear un sistema operativo para terminales móviles.

8. BlackBerry OS.- Es un sistema operativo móvil desarrollado por Research In Motion para sus dispositivos BlackBerry.- Es multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la trackwheel, trackball, touchpad y pantallas táctiles.

9. HP webOS.- Se trata de un sistema operativo multitarea para sistemas embebidos basado en Linux, desarrollado por Palm Inc., ahora es propiedad de Hewlett-Packard Company.

10. GrapheneOS.- Es un sistema operativo móvil centrado en la privacidad y la seguridad con compatibilidad con aplicaciones Android, que está desarrollado como un proyecto de código abierto sin ánimo de lucro. Se centra en la investigación y el desarrollo de tecnología de privacidad y seguridad, incluyendo mejoras sustanciales en el Sandboxing, la mitigación de exploits y el modelo de permisos.

11. Sailfish OS.- Es un Sistema Operativo móvil seguro y optimizado para funcionar en Smartphones y tabletas, y también fácilmente adaptable a todo tipo de dispositivos integrados y casos de uso. Es el único Sistema Operativo móvil independiente basado en el código abierto, sin ningún vínculo con las grandes corporaciones, respaldado por unos sólidos derechos de propiedad intelectual, que incluyen todos los derechos de propiedad intelectual y marcas comerciales. En resumen, es una plataforma abierta con un modelo de contribución de código abierto activo.

12. Bada.- Fue un sistema operativo para teléfonos móviles desarrollado por Samsung (Bada «océano» o «mar» en coreano). Diseñado para cubrir teléfonos inteligentes de gama alta como gama baja.

3 Seguridad, Privacidad y Vigilancia

Ante el constante aumento -explosivo- en el uso de dispositivos conectados a internet como computadoras personales, Laptops, tabletas y teléfonos celulares, expertos en ciberseguridad advierten un entorno propicio para que prosperen los cibercriminales y que, tanto individuos como empresas, se encuentren expuestos a múltiples amenazas de ciberseguridad.

En la actualidad, aproximadamente la mitad de la población mundial accede de algún modo a internet. Con tantos accesos concurrentes a la red de redes, la posible amenaza de seguridad a los sistemas informáticos crece y se complejiza, a pesar de las diversas y especializadas maneras de contrarrestarlas. Por su propia naturaleza, una conexión a internet hace que tu equipo de cómputo quede expuesto a ataques y pueda ser accesible por otro equipo, llegando a tener acceso a tu información.

¿Por qué?

- En muchas ocasiones, los usuarios no cuentan con la suficiente sensibilización sobre su exposición al riesgo, o bien, no están familiarizados con las herramientas y/o capacitación de sus organizaciones para prevenir y enfrentar amenazas de ciberseguridad.

- Los usuarios de internet a menudo utilizan redes Wi-Fi no seguras y usan dispositivos que no están configurados con los controles de políticas de seguridad básicos, lo cual los vuelve excepcionalmente vulnerables a ataques cibernéticos.

- Es común que los usuarios compartan dispositivos (computadoras, tabletas, teléfonos) para educación, trabajo y esparcimiento aumentando con ello su exposición a amenazas de ciberseguridad.

- Los piratas informáticos eligen como blanco la dependencia, cada vez mayor, de las personas con respecto a las herramientas digitales, además de que más tiempo en línea incrementa la potencial exposición de los usuarios a amenazas de ciberseguridad.

- Un error común es creer que los ciberatacantes utilizan únicamente herramientas muy avanzadas y técnicas para Hackear las computadoras o cuentas de las personas. Los atacantes han aprendido que la forma

más sencilla de robar la información de los usuarios, comprometer sus cuentas o infectar los sistemas es simplemente engañar al usuario para que lo hagan por ellos con una técnica denominada ingeniería social.

- Los piratas informáticos son extremadamente creativos al idear formas de aprovecharse de los usuarios y de la tecnología para acceder a contraseñas, redes y datos, a menudo sirviéndose de herramientas de ingeniería social y de temas y tendencias populares para tentar a los usuarios a tener comportamientos inseguros en línea.

Todo lo anterior, ha creado una enorme superficie de exposición a ataques cibernéticos dirigidos a los usuarios, la red, la computadora portátil, el teléfono inteligente, la tableta, etc. con la intención de cometer delitos informáticos. Por ello estos lineamientos generales de ciberseguridad son sugeridos para el uso seguro de redes y dispositivos de telecomunicaciones en apoyo al usuario, contiene recomendaciones sencillas y prácticas para fomentar una buena salud cibernética de los dispositivos utilizados para el trabajo a distancia, educación y diversión; y promover la ciberseguridad entre las personas y sus organizaciones.

La seguridad de la información en la red es más que un problema de protección de datos, y debe estar básicamente orientada en asegurar la información importante de las personas, de las organizaciones y la propiedad intelectual. Los riesgos de la información están presentes cuando confluyen fundamentalmente dos elementos: las amenazas de ataques, y las vulnerabilidades de la tecnología; conceptos íntimamente relacionados donde no es posible ninguna consecuencia sin la presencia conjunta de estos.

3.1 ¿Qué es la Privacidad y por qué es Importante?

Consideramos que la privacidad debería ser un derecho básico y una protección necesaria en la era digital para evitar la victimización y la manipulación. Una sociedad no puede tener libertad sin privacidad. Puede parecer un lujo, pero es importante para el bienestar de una sociedad libre y justa. Muchos gobiernos y empleadores hacen vigilancia[37] activa a sus ciudadanos o empleados para monitorear ideas, discusiones o disensiones no deseadas. Los

[37] ¿Qué es vigilancia? Vigilancia es el seguimiento de las comunicaciones, acciones o movimientos de una persona por un gobierno, empresa, grupo o persona.

¿Cuándo es legal la vigilancia? En general, cuando es selectiva, se basa en indicios suficientes de conducta delictiva y está autorizada por una autoridad estrictamente inde-

infractores son luego procesados o reeducados para alinearse con lo que las autoridades consideran apropiado. Sin el beneficio del anonimato, el deseo de los ciudadanos de expresar sus pensamientos se reprime efectivamente.

En la era digital, la privacidad va más allá del anonimato, al proteger a las personas de la victimización y la manipulación. La sociedad ha adoptado la tecnología para educarse, comunicarse, realizar negocios y formar relaciones. Nuestros puntos de vista y opiniones están fuertemente influenciados por lo que aprendemos de las fuentes de noticias locales, nacionales e internacionales.

Contribuimos en gran medida al panorama digital a través de nuestras acciones y decisiones. Nuestras huellas digitales están en todas partes. Cuentan una historia de a dónde vamos, qué hacemos, quién nos gusta o no, y qué pensamos. Se crean con cada Clic que hacemos y con cada archivo, aplicación y dispositivo que usamos. Cuando esos datos se agregan, pueden proporcionar información tremendamente poderosa sobre una persona o comunidad, lo suficiente como para construir personas complejas y precisas.

Esta información se usa comúnmente para manipular las creencias y comportamientos de las personas. Las compras en línea son un ejemplo perfecto: el Marketing dirigido y la publicidad basada en datos es un gran negocio porque logra que la gente gaste dinero. Todo se reduce a saber lo que las personas hacen, piensan, dicen, consumen y miran[38]. Tener acceso a grandes

pendiente, como un juez.

¿Qué es la vigilancia masiva? La vigilancia masiva indiscriminada es el control de las comunicaciones por internet y telefónicas de un gran número de personas -a veces de países enteros- sin que existan indicios suficientes de conducta delictiva. Este tipo de vigilancia no es legal.

¿Qué datos recogen? Algunos gobiernos almacenan y analizan historiales de navegación, búsquedas en internet, mensajes de correo electrónico, mensajes instantáneos, conversaciones por Webcam y llamadas telefónicas. También reúnen metadatos -datos sobre datos-, como destinatarios de correo electrónico, horas de llamadas y registros de ubicación.

¿Qué hacen con mis datos? Se guardan en grandes centros de datos donde unos algoritmos informáticos pueden hacer búsquedas en ellos y analizarlos. También están a disposición de las autoridades de las agencias de seguridad a través de potentes bases de datos como XKeyscore.

[38]Has notado que si buscas algo en un buscador comercial -como Google, Yahoo, Bing, Ask, Baidu, etc.-, minutos después en tus distintas redes sociales aparecerán mensajes relacionados a tu búsqueda. Esto ocurre porque los buscadores comerciales en cada búsqueda que haces, vídeo o los anuncios que ves o en los que haces Clic comparten tu ubicación, los sitios que visitas, los dispositivos, navegadores y aplicaciones que usas para acceder a los servicios del buscador con fines comerciales.

cantidades de datos privados les brinda a los anunciantes la capacidad de crear mensajes oportunos y significativos que atraigan a las personas a los comportamientos deseados.

Pero si los minoristas pueden hacer que las personas compren cosas que no necesitan, ¿para qué más se pueden usar los datos privados? ¿Qué tal cambiar lo que piensa la gente, a quiénes apoyan, sus opiniones políticas, qué debería convertirse en ley y en qué creer? El uso de información privada se ha aprovechado durante mucho tiempo para promover, vilipendiar o perseguir a diversas religiones, partidos políticos y líderes.

En las últimas décadas, ha cambiado la forma en que los ciudadanos del mundo reciben sus noticias. Los segmentos de noticias y entretenimiento han comenzado a mezclarse, a menudo informando hechos con adornos e historias obstinadas para influir en las opiniones públicas. Cuanta más información privada se conozca, más fácil será influir, convencer, engatusar o amenazar a las personas.

Según informes de Oxford Internet Institute, a pesar de los esfuerzos para combatir la propaganda computacional, el problema ha crecido a gran escala. El mayor crecimiento proviene de propaganda política que es difundida junto con desinformación y noticias basura alrededor de los períodos electorales. Se incrementa el número de campañas políticas a nivel mundial que usan Bots, noticias basura y desinformación para polarizar y manipular a los votantes.

Un velo de privacidad puede proteger tanto los beneficios como los abusos. La tendencia actual es establecer y ampliar los derechos de privacidad en beneficio de los ciudadanos. Esto reduce la victimización, manipulación y explotación digitales al proteger los datos confidenciales y permite actividades que promueven la libertad y la libertad de expresión.

Sin leyes, los gobiernos y las empresas han desarrollado prácticas que aprovechan el poder de recopilar información confidencial y utilizarla en su propio beneficio. Las nuevas leyes de privacidad están cambiando el panorama y muchas empresas éticas reducen sus esfuerzos de cobranza para ser más conservadoras y respetuosas. También muestran flexibilidad en la forma en que tratan, protegen y comparten dichos datos.

Algunos gobiernos y agencias también están reduciendo la recolección, limitando la retención o terminando los programas domésticos que los ciudadanos consideran invasivos. Al mismo tiempo, los organismos encargados

Una opción para nuestra privacidad es usar buscadores que no dejan rastro de nuestras búsquedas como puede ser: DuckDuckGo.

de hacer cumplir la ley quieren conservar la capacidad para detectar e investigar delitos, para proteger la seguridad de los ciudadanos.

También se hace un mal uso de la privacidad. Es la herramienta preferida por quienes cometen delitos y permite que los actos atroces contra otros no se detecten. Puede ocultar actos terribles y permitir una coordinación generalizada entre el fraude, abuso y terror.

Se argumenta que las puertas traseras digitales, las claves maestras y los algoritmos de cifrado que obtienen acceso a los sistemas y la información privada ayudarían en la detección legal de actividades delictivas y en las investigaciones para identificar a los terroristas. Aunque suena como una gran herramienta contra los delincuentes, es una caja de Pandora.

Las puertas traseras y las llaves maestras no limitan el acceso para una investigación específica donde existe una causa probable, sino que permiten una vigilancia generalizada[39] y la recolección de datos de toda una población, incluidos los ciudadanos respetuosos de la ley. Esto viola el derecho de las personas a la privacidad y abre la puerta a la manipulación y el enjuiciamiento político. La capacidad de leer cada texto, correo electrónico, mensaje y conversación en línea para "monitorear" a la población crea un camino claro hacia el abuso. El riesgo de control y explotación es real.

Incluso para aquellos que no tienen ninguna objeción a que su gobierno tenga acceso, debemos considerar el hecho de que tales puertas traseras y

[39] El 5 de junio de 2013, el exanalista de la CIA y de la NSA Edward Snowden decidió revelar la existencia de programas de vigilancia sobre las comunicaciones de millones de ciudadanos de todo el mundo. A través de The Guardian y The Washington Post, supimos que, en nombre de la seguridad y sin ningún control judicial, la NSA y el gobierno británico habían rastreado e-mails, llamadas telefónicas y mensajes encriptados. Empresas como Facebook, Google y Microsoft habían sido obligadas a entregar información de sus clientes por órdenes secretas de la NSA. Esta misma agencia grabó, almacenó y analizó los 'metadatos' de las llamadas y de los mensajes de texto enviados en México, Kenia y Filipinas. Incluso llegaron a espiar el móvil de la canciller alemana Angela Merkel.

Snowden hizo las filtraciones a la prensa desde Hong Kong y actualmente vive en Rusia, donde se le concedió asilo. No puede volver a su país porque está acusado de revelar información clasificada a personas no autorizadas y de robo de propiedad del gobierno federal. Para unos es un héroe y para otros un traidor, gracias a las revelaciones de Snowden la opinión pública es más consciente de su derecho a la privacidad y ha reaccionado oponiéndose al espionaje masivo.

Aunque queda un largo camino para asegurar que los Gobiernos no invadan la vida privada de las personas, los tribunales han declarado ilegales algunos aspectos de estos programas y las empresas tecnológicas han tenido que posicionarse ante un escándalo capaz de dañar seriamente su reputación.

claves maestras serían buscadas por ciberdelincuentes y otros actores del estado-nación. Ningún sistema es infalible. Con el tiempo, los delincuentes encontrarían y utilizarían estas herramientas en detrimento de la comunidad digital mundial. Algunas puertas traseras podrían valer decenas de miles de millones de dólares para el comprador adecuado, ya que podrían desbloquear un poder inimaginable para apoderarse de la riqueza, afectar a la gente, dañar naciones, socavar la independencia y reprimir el pensamiento libre.

Proteger la privacidad no se trata de ocultar información. Se trata de la capacidad de liberarse de influencias no deseadas, de la tiranía y de comunicarse con los demás de formas que desafíen el statu quo. La privacidad protege a las personas, pero también los cimientos de una sociedad libre. La privacidad no es un tema fácil y no existe una solución perfecta. Es una situación dinámica y seguirá cambiando con el sentimiento público.

Todos quieren cierto nivel de discreción, confidencialidad y espacio. Nadie quiere que se expongan sus contraseñas, finanzas familiares, detalles de relaciones personales, historial médico, ubicación, compras y discusiones privadas. Las personas tampoco disfrutan de verse inundadas de Spam, Phishing y llamadas de ventas incesantes. La privacidad no se trata necesariamente de ocultar algo, sino de limitar la información a quienes tienen derecho a saber.

Muy poca privacidad puede socavar la libertad de expresión, la libertad y la denuncia de victimizaciones. También empodera a entidades poderosas para manipular el mundo digital de las personas para coaccionarlas, manipularlas y victimizarlas. Demasiada privacidad puede permitir que los actores criminales prosperen y se escondan de las autoridades. Debe lograrse un equilibrio[40].

[40]Los gobiernos nos enfrentan a un dilema falso: seguridad o libertad. En un estado de derecho, donde las leyes equilibran ambos conceptos, las personas son inocentes hasta que se demuestra lo contrario y tienen derecho a que se respete su vida privada. Por tanto, antes de violar estos derechos, los gobiernos deben de tener indicios de que se está cometiendo un delito. No pueden buscar pruebas aleatoriamente en nuestras comunicaciones privadas antes de que se cometa ese delito.

Varios países dan "carta blanca" a la vigilancia indiscriminada en sus leyes. En Francia se permite interceptar masivamente comunicaciones, retener información durante largos períodos de tiempo y se ha eliminado la autorización judicial previa. También Reino Unido ha introducido en su legislación mayores poderes de espionaje. Polonia ha otorgado poderes de vigilancia incompatibles con respecto a la privacidad a la policía y o otras agencias. Otros países como China o Rusia también vigilan internet con total desprecio a la intimidad de las personas.

Algunos gobiernos y empresas tecnológicas quieren que aceptemos que no tenemos derechos cuando estamos en internet. Que cuando usamos el teléfono o entramos en nuestra cuenta de correo electrónico, todo lo que hacemos o decimos les pertenece. No permitiríamos este grado de intrusión en nuestra vida fuera de internet, así que no debemos permitirlo dentro.

Cuando los gobiernos no protegen los derechos humanos, sólo la acción de la gente común y corriente puede hacer que las cosas cambien y que los responsables de los abusos rindan cuentas. Muchas organizaciones trabajan para que los gobiernos prohíban la vigilancia masiva y el intercambio ilegal de información confidencial. Esta será una larga lucha no exenta de dificultades, pero cuyo impulso va creciendo con el tiempo.

¿Qué son los datos biométricos? en general entenderemos por datos biométricos a las características físicas, fisiológicas, morfológicas, de comportamiento y rasgos de personalidad distintivas de cada persona que permite distinguir ciertas singularidades e identificar al individuo en cuestión.

El uso de datos biométricos no es nuevo e incluso es algo en lo que constantemente se ven envueltos las redes sociales[41], aunque los fines de uso son diversos. Recientemente Texas demandó a Meta, dueña de Facebook, por almacenar millones de estos datos y comercializarla con terceros.

Las Amenazas a los Usuarios en un entorno dinámico de interconectividad, pueden venir de cualquier parte, sea interna o externa, e íntimamente relacionada con el entorno de las organizaciones. Las vulnerabilidades son una debilidad en la tecnología o en los procesos asociados con la información, y como tal, se consideran características propias de los sistemas o de la infraestructura que los soporta.

Las personas o usuarios de internet que emplean las tecnologías para vulnerar los sistemas en la red, robar información y/o infectarlos con comportamientos dudosos, son comúnmente conocidos como *Crackers*.

[41] Por ejemplo en el caso de X antes Twitter, podrá hacerse de tu foto de perfil, así como otros datos que incluyen historial de empleo, educación, preferencias de empleo, capacidades y habilidades, así como búsqueda de trabajo, datos que utilizaría para ofrecer servicios similares a los de Linkedln, "recomendarte potenciales trabajos, compartir con potenciales empleadores cuando solicitas un trabajo, permitir que los empleadores encuentren potenciales candidatos y mostrarte publicidad más relevante".

El término Hacker[42] en el sentido más filosófico tiende a promover una conciencia colectiva de la libertad del conocimiento y la justicia social, por lo que muchas veces se los encuentra en situaciones de activismo (llamado en este caso Hacktivismo) en pos de dicha ideología. Su forma de actuar, por lo general, determina su clasificación en:

- *Hacker* de Sombrero Blanco (White Hat): éticos, expertos en seguridad informática, especializados en realizar test de intrusión y evaluaciones de seguridad.
- *Hacker* de Sombrero Negro (Black Hat): también conocidos como *Crackers*, vulneran los sistemas de información con fines maliciosos.
- *Hacker* de Sombrero Gris (Grey Hat): en ocasiones vulneran la ley, y de forma general no atacan malintencionadamente o con intereses personales, sino que sus motivaciones se relacionan a protestas o desafíos personales.
- *Hacker* de Sombrero Verde (Green Hat): son novatos que pueden evolucionar y buscan convertirse en expertos.
- *Hacker* de Sombrero Azul (Blue Hat): son personas expertas que se les paga para probar Software en busca de errores antes de que éste salga al mercado.

Para una entidad, la fuga de información provocada por el actuar de algunos de estos usuarios de la red, puede ocurrir deliberadamente como resultado de una acción intencional de algún empleado descontento, como consecuencia de un ciberataque, o inadvertidamente, por un colaborador desprevenido víctima de un Software malicioso.

Una de las principales amenazas para los dispositivos tecnológicos utilizados para el trabajo y estudio a distancia es el Malware, también conocido como Software o código malicioso. Éste se define como cualquier programa informático que se coloca de forma oculta en un dispositivo, con la intención de comprometer la confidencialidad, integridad o disponibilidad de los datos, las aplicaciones o el sistema operativo.

[42] Existen algunos otros términos en el ciberespacio, por ejemplo: Newbie, que significa principiante; Lammer, persona que presume tener conocimientos que realmente no posee; Phreaker, Hacker orientado a los sistemas telefónicos; Script Kiddie, quien utiliza programas creados por terceros sin conocer su funcionamiento.

Los tipos más comunes de amenazas de Malware incluyen Virus, gusanos, troyanos, Rootkits y Spyware. Las amenazas de Malware pueden infectar cualquier dispositivo por medio del correo electrónico, los sitios Web, las descargas y el uso compartido de archivos, el Software punto a punto y la mensajería instantánea.

Además, existen amenazas relacionadas con la ingeniería social como el Phishing, Smishing y Vishing, por medio de los cuales los atacantes intentan engañar a las personas para que revelen información confidencial o realicen ciertas acciones, como descargar y ejecutar archivos que parecen ser benignos, pero que en realidad son maliciosos.

3.2 Las Vulnerabilidades y Exposiciones Comunes

De manera global, la última década ha sido testigo del cambio de paradigma en que los atacantes buscan explotar vulnerabilidades dentro de las organizaciones y las infraestructuras de redes. Con el fin de contrarrestar estos ataques, las políticas de seguridad persiguen constantemente aprender de ellos para estar preparados lo mejor posible, y en este sentido, intentar garantizar confianza y tranquilidad a los usuarios de la red, sobre el empleo de sus datos, finanzas y propiedades intelectuales.

Los ataques de seguridad vienen en muchas formas y usan varios puntos de entrada. Cada tipo de ataque viene en varios tipos, ya que generalmente hay más de una forma en que se pueden configurar o camuflar en función de la experiencia, los recursos y la determinación del pirata informático.

Las Vulnerabilidades y Exposiciones Comunes (Common Vulnerabilities and Exposures, CVE[43]) que tienen los distintos sistemas operativos (productos), es una lista de información registrada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación llamada CVE-ID, descripción de la vulnerabilidad, que versiones del Software están afectadas, posible solución al fallo (si existe) o como confi-

[43] https://www.cvedetails.com/
https://www.cvedetails.com/top-50-products.php
https://thebestvpn.com/vulnerability-alerts/
Hay que notar que en la base de datos se diferencian distintos productos de Windows (como Windows 7.0 u 8.1) pero no se hace lo mismo con los demás productos (por ejemplo para Debian GNU/Linux están los errores desde 1999 a la fecha) con lo que es engañosa la comparación del número vulnerabilidades para un sistema operativo con tantas versiones generadas en dicho tiempo con otro con un tiempo de vida corto.

gurar para mitigar la vulnerabilidad y referencias a publicaciones o entradas de foros o Blogs donde se ha hecho pública la vulnerabilidad o se demuestra su explotación. Además suele también mostrarse un enlace directo a la información de la base de datos de vulnerabilidades (https://nvd.nist.gov, https://openssf.org y https://docs.aws.amazon.com/security), en la que pueden conseguirse más detalles de la vulnerabilidad y su valoración.

El mundo está cada vez más interconectado y, como resultado de esto, la exposición a las vulnerabilidades de seguridad también ha aumentado dramáticamente. Las complejidades de mantener las plataformas de cómputo actuales hacen que sea muy difícil para los desarrolladores cubrir cada punto de entrada potencial. En 2019 hubo un promedio de más de 45 vulnerabilidades y exposiciones comunes registradas por día y estas siguen en aumento año con año.

El CVE-ID ofrece una nomenclatura estándar para identificación de la vulnerabilidad de forma inequívoca que es usada en la mayoría de repositorios de vulnerabilidades.

Es definido y es mantenido por The MITRE Corporation (por eso a veces a la lista se la conoce por el nombre MITRE CVE List) con fondos de la National Cyber Security Division del gobierno de los Estados Unidos de América. Forma parte del llamado Security Content Automation Protocol.

Por otro lado, como parte de las amplias revelaciones sobre vigilancia masiva filtradas en 2013, 2014 y años posteriores, se descubrió que las agencias de inteligencia estadounidenses y británicas, la Agencia de Seguridad Nacional (NSA, por sus siglas en inglés) y el Cuartel General de Comunicaciones del Gobierno (GCHQ, por sus siglas en inglés), respectivamente, tienen acceso a los datos de los usuarios de dispositivos Android. Estas agencias son capaces de leer casi toda la información del teléfono como los SMS, geolocalización, correos, notas o mensajes.

Documentos filtrados en enero de 2014, revelaron que las agencias interceptan información personal a través de internet, redes sociales y aplicaciones populares, como Angry Birds, que recopilan información para temas comerciales y de publicidad. Además, según The Guardian, el GCHQ tiene una wiki con guías de las diferentes aplicaciones y redes de publicidad para saber los diferentes datos que pueden ser interceptados. Una semana después de salir esta información a la luz, el desarrollador finlandés Rovio, anunció que estaba reconsiderando sus relaciones con las distintas plataformas publicitarias y exhortó a la industria en general a hacer lo mismo.

Las informaciones revelaron que las agencias realizan un esfuerzo adi-

cional para interceptar búsquedas en Google Maps desde Android y otros teléfonos inteligentes para recopilar ubicaciones de forma masiva. La NSA y el GCHQ insistieron en que estas actividades cumplen con las leyes nacionales e internacionales, aunque The Guardian afirmó que «las últimas revelaciones podrían sumarse a la creciente preocupación pública acerca de cómo se acumula y utiliza la información, especialmente para aquellos fuera de Estados Unidos de Norte América, que gozan de menos protección en temas de privacidad que los estadounidenses».

3.3 Alfabetismo Digital

Según la organización Common Sense Media, el alfabetismo digital es la capacidad de encontrar, identificar, evaluar y usar la información encontrada en medios digitales de manera efectiva. Básicamente, es la misma definición tradicional de alfabetismo, pero adaptada a la era digital y a fuentes no tradicionales de información.

El anuario del 2016 de la UNESCO en "Alfabetización mediática e informacional para los objetivos de desarrollo sostenible" hace referencia a las "Cinco leyes de la alfabetización mediática e informacional":

1. La información, la comunicación, las bibliotecas, los medios de comunicación, la tecnología, el internet y otras fuentes de información se encuentran en la misma categoría. Ninguna es más relevante que la otra ni debe ser tratada como tal.

2. Cada ciudadano es un creador de información o conocimiento y tiene un mensaje. Todas las personas deben estar facultadas para acceder a nueva información y expresarse.

3. La información, el conocimiento y los mensajes no siempre están exentos de valores o prejuicios. Cualquier conceptualización, uso y aplicación de alfabetismo digital debe presentar este hecho de manera transparente y comprensible para todos los ciudadanos.

4. Todo ciudadano desea conocer y comprender información, conocimientos y mensajes nuevos, así como comunicarse; y sus derechos nunca deben ser comprometidos.

5. El alfabetismo digital es un proceso dinámico de experiencias vividas. Se considera completa cuando incluye conocimientos, habilidades y actitudes, cuando abarca el acceso, la evaluación, el uso, la producción y la comunicación de información, de contenido mediático y tecnológico.

De igual manera, en el anuario de la UNESCO se exponen 10 habilidades que deben desarrollarse para lograr la alfabetización digital, o como lo define el anuario, alfabetización mediática e informacional. Estas habilidades son:

- Interactuar con información referente a los medios y la tecnología.
- Ser capaz de aplicar habilidades técnicas de comunicación de información para procesar la información y producir contenido mediático.
- Utilizar, de manera ética y responsable la información y comunicar su comprensión o conocimiento adquirido a una audiencia o lectores en una forma y medio apropiados.
- Extraer y organizar información y contenidos.
- Evaluar de forma crítica la información y el contenido presentado en los medios y otras fuentes de información, incluyendo medios en línea, en términos de autoridad, credibilidad, propósito y posibles riesgos.
- Localizar y acceder a información de contenido relevante.
- Sintetizar las ideas extraídas del contenido.
- Comprender las condiciones bajo las cuales se pueden cumplir esas ideas o funciones.
- Comprender el papel y las funciones de los medios, incluyendo medios en línea, en la sociedad y su desarrollo.
- Reconocer y articular la necesidad de información y de los medios.

Internet ha sido una herramienta que ha transformado y definido la comunicación en el siglo XXI. A través de sus múltiples interfaces, internet ha tenido éxito, para que tanto individuos como organizaciones se conecten, se comuniquen e intercambien información. Las plataformas tecnológicas y las redes sociales han acelerado la velocidad a través de la cual los usuarios

pueden acceder y recuperar información, simplificando el proceso en el que las noticias se difunden, actualizan e incluso se comunican. Hoy en día, es prácticamente instantáneo darse cuenta de un evento de noticias sin que sea necesariamente comunicado por medios tradicionales como los periódicos o la radio.

La facilidad a través de la cual las personas ahora pueden comunicarse ha traído una sensación de democratización a la libertad de expresión. Transformar la libertad de expresión ha posibilitado nuevas capacidades para crear y editar contenido, generando nuevas oportunidades para el periodismo alternativo; nuevas capacidades de organización y movilización (que apoyan en gran medida otros derechos, como la libertad de asociación); y nuevas posibilidades para innovar y generar desarrollo económico (apoyando los derechos sociales y económicos).

Sin embargo, esta facilidad en el intercambio y la creación de información también presenta desafíos tanto para las organizaciones como para las personas que son usuarias de dichas redes, tanto como fuente, como usuario final. Aunque estos desafíos varían en escala, todos son igualmente significativos. Algunos de los más destacados incluyen el desafío a la calidad superficial de la información, la susceptibilidad a la información errónea y la exposición a ataques cibernéticos. Por lo tanto, la necesidad de mitigar y mantener la integridad de esta información se ha convertido en un área de trabajo creciente para muchas organizaciones públicas y privadas.

En las siguientes secciones se ofrece una variedad de técnicas y mejores prácticas para mitigar y contrarrestar los desafíos mencionados anteriormente. Sin embargo, es importante tener en cuenta, al leer estas recomendaciones, la posición que representas o con la que te asocias. Algunas de las recomendaciones pueden no ser aplicables a una figura pública, como políticos, activistas u otros actores cuyas mejores prácticas en las redes sociales están sujetas a un mayor escrutinio. En este sentido, el ejercicio de los derechos de expresión, reunión y protesta se debe respetar, y debe ser respetado, en el ámbito digital al tiempo que se garanticen prácticas más seguras de internet.

3.4 Amenazas a la Ciberseguridad

La aparición de vulnerabilidades en los sistemas operativos y los métodos de encubrimiento de los atacantes, lo convierten en una práctica en aumento. Algunos de los principales ataques en la red, hacia donde dirigen su mirada

los *Hackers* para vulnerar la seguridad, pueden definirse como:

- Shoulder Surfing: es una técnica mediante la que el ciberdelincuente consigue nuestra información mirando "por encima del hombro" desde una posición cercana, mientras utilizamos los dispositivos sin darnos cuenta.

- Dumpster Diving: se le conoce como el proceso de buscar en nuestra basura para obtener información útil sobre nuestra persona o empresa que luego puede utilizarse contra nosotros para otro tipo de ataques.

- Malware: el término se refiere de forma genérica a cualquier Software malicioso que tiene por objetivo infiltrarse en un sistema para dañarlo. Comúnmente se asocian como tipos de Malware a los virus, gusanos y troyanos.

- Virus: es un código que infecta los archivos del sistema mediante un programa maligno, pero para ello necesita que el usuario lo ejecute directamente. Una vez activo, se disemina por todo el sistema a donde el equipo o cuenta de usuario tenga acceso, desde dispositivos de Hardware hasta unidades virtuales o ubicaciones remotas en una red.

- Gusanos: es un programa que, una vez infectado el equipo, realiza copias de sí mismo y las difunde por la red. A diferencia del virus, no necesita la intervención del usuario, ya que pueden transmitirse utilizando las redes o el correo electrónico. Son difíciles de detectar, pues su objetivo es difundirse e infectar a otros equipos, y no afectan inicialmente el funcionamiento normal del sistema. Su uso principal es el de la creación de redes zombies (Botnets), utilizadas para ejecutar acciones de forma remota como ataque de denegación de servicio (DoS) a otro sistema.

- Troyanos: similares a los virus, sin embargo, mientras que este último es destructivo por sí mismo, el troyano lo que busca es abrir una puerta trasera (Backdoor) para favorecer la entrada de otros programas maliciosos. Su misión es precisamente pasar desapercibido e ingresar a los sistemas sin que sea detectado como una amenaza potencial. No se propagan a sí mismos y suelen estar integrados en archivos ejecutables aparentemente inofensivos.

- Spyware: es un programa espía, cuyo objetivo es recopilar información de un equipo y transmitirla a una entidad externa sin el consentimiento del propietario. Su trabajo suele ser silencioso, sin dar muestras de su funcionamiento, llegando incluso a instalar otros programas sin que se perciban. Las consecuencias de su infección incluyen, además, pérdida considerable del rendimiento del sistema y dificultad para conectarse a internet.

- AdWare: su función principal es la de mostrar publicidad. Aunque su intención no es la de dañar equipos, es considerado por algunos una clase de Spyware, ya que puede llegar a recopilar y transmitir datos para estudiar el comportamiento de los usuarios y orientar mejor el tipo de publicidad.

- Ransomware[44]: este es uno de los más sofisticados y modernos ataques, ya que lo que hace es secuestrar datos (encriptándolos) y pedir un

[44] El Ransomware se ha convertido en la principal amenaza para la ciberseguridad mundial. Desde que el troyano WannaCry afectó en la primavera de 2017 al menos a 200.000 equipos y servidores de 150 países, poniendo 'contra las cuerdas' a importantes empresas, el uso de este tipo de ataques informáticos no ha dejado de aumentar y son cada vez más numerosos, sofisticados, peligrosos y masivos.

Si en sus inicios los atacantes se conformaban infectando ordenadores de consumo a cambio de unos pocos dólares, todos los informes apuntan que los ciberdelincuentes están enfocando su ámbito de actuación preferente al segmento empresarial, organizaciones, administraciones e infraestructuras públicas con Malware tan peligroso como 'CryptoWall', 'Babuk', 'Black Kingdom', 'Ryuk' o 'CryptoLocker' que destacan por su alto nivel de código y su capacidad de control de ordenadores y redes mediante el cifrado de archivos.

El número de víctimas por Ransomware es ya interminable. La última conocida (abril 2021) ha sido la cadena de tiendas Phone House, pero la lista de los últimos meses es amplísima: la multinacional Acer; el estudio CD Projekt Red; la asociación deportiva NBA; el líder en cámaras Canon; el desarrollador japonés Capcom; la firma de seguros Mapfre; municipios y hospitales estadounidenses o la infraestructura del SEPE, el Servicio Público de Empleo Estatal de España que gestiona subsidios y maneja datos personales de millones de desempleados.

Solo es un ejemplo de afectados y, además, se sospecha que otras empresas han recibido ataques, aunque han preferido no divulgarlos públicamente ante la pérdida reputacional que suponen estos incidentes que son una lacra que parece imparable. Realmente, no hay sistema operativo, plataforma, dispositivo o red informática que esté a salvo, porque el Ransomware emplea cualquier tipo de vulnerabilidad, tipo de Malware o de ataque para secuestrar los equipos. Y no en todas las ocasiones es detectado a tiempo por los sistemas de seguridad y Software antimalware.

Teniendo en cuenta que un Ransomware típico puede infectar dispositivo móviles, ordenadores personales, servidores o redes, bloqueando su funcionamiento y/o acceso a una

rescate por ellos. Normalmente, se solicita una transferencia en dinero electrónico (Bitcoins), para evitar el rastreo y localización. Este tipo de ciberataque va en aumento y es uno de los más temidos en la actualidad.

- Stalkerware: se trata de un programa que permite el seguimiento y monitorización de la actividad del usuario en un dispositivo móvil como teléfono, tableta o computadora. El problema de este tipo de programas es que no han sido creados puntualmente para el espionaje y el acoso, sino para el intercambio de datos entre dispositivos de manera más sencilla. Y qué sí se instalan sin el consentimiento de la persona, permite espiar sus comunicaciones y toda su actividad gracias a las funcionalidades y sensores incorporados en el dispositivo.

- Escaneo de Puertos: técnica empleada para auditar dispositivos y redes con el fin de conocer qué puertos están abiertos o cerrados, los servicios que son ofrecidos, así como comprobar la existencia de algún cortafuegos (Firewall), la arquitectura de la red, o el sistema operativo, entre otros aspectos. Su empleo permite al atacante realizar un análisis preliminar del sistema y sus vulnerabilidades, con miras a algún otro tipo de ataque, pues cada puerto abierto en un dispositivo, es una potencial puerta de entrada al mismo.

- Phishing: no es un Software, se trata más bien de diversas técnicas de suplantación de identidad para obtener datos privados de las víctimas, como contraseñas o datos bancarios. Los medios más utilizados son el correo electrónico, mensajería o llamadas telefónicas, y se hacen pasar por alguna entidad u organización conocida, solicitando datos confidenciales, para posteriormente ser utilizado por terceros en su beneficio.

- Whaling: es un método para simular ocupar cargos de nivel superior en una organización con el objeto de conseguir información confidencial u obtener acceso a sistemas informáticos con fines delictivos.

- El Smishing: ocurre cuando se recibe un mensaje de texto corto (SMS)

parte o a todo el equipo apoderándose de los archivos con un cifrado fuerte y exigiendo una cantidad de dinero como "rescate" para liberarlos, el mejor (y casi único) de los consejos en ciberseguridad es la prevención con las copias de seguridad como máximo exponente. La opción de pagar el rescate para recuperar el acceso a los archivos es muy negativa para la industria ya que retroalimenta aún más esta amenaza.

en el teléfono celular, por medio del cual se solicita al usuario llamar a un número de teléfono o ir a un sitio Web.

- El Vishing: es la estafa que se produce mediante una llamada telefónica que busca engañar, suplantando la identidad de una persona o entidad para solicitar información privada o realizar alguna acción en contra de la víctima.

- Juice-jacking o Video-jacking: son los nombres que se han puesto a los procesos por los cuales, a través de un puerto (generalmente USB) nos conectamos a un puerto Hackeado, de esta forma el ciberdelincuente puede instalar, grabar datos, tomar video o sacar datos de nuestros dispositivos móviles. Esto se ha vuelto común en los puertos de recarga de energía públicos a través del puerto USB. Si se hará uso de este tipo de servicios, es recomendable adquirir un dispositivo del tipo PortaPow de carga rápida con adaptador USB que inhabilitan los pines de datos permitiendo sólo la carga del dispositivo.

- Keylogger (registrador de teclas): es un Software malicioso o dispositivo en Hardware -generalmente conectado al teclado- que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente usarlas para robar información privada.

- Criptomining es un Malware diseñado para la extracción de Criptomonedas en nuestros dispositivos. Si el usuario accesa a un sitio Web que esté infectado, el Malware se puede descargar de forma inadvertida a través de una descarga automática y nuestro dispositivo comenzará a desenterrar una moneda criptográfica seleccionada para los Hackers; la infección será notoria por un uso intensivo de nuestro dispositivo.

- Botnets (Redes de robots): Son computadoras o dispositivos conectados a la red (teléfonos inteligentes, tabletas, etc.) infectados y controlados remotamente, que se comportan como robots (Bots) o zombies, quedando incorporados a redes distribuidas, las cuales envían de forma masiva mensajes de correo Spam o código malicioso, con el objetivo de atacar otros sistemas o dejarlos fuera de servicio.

- Denegación de Servicios: tiene como objetivo inhabilitar el uso de un sistema o computadora, con el fin de bloquear el servicio para el que está destinado. Los servidores Web poseen la capacidad de resolver un

número determinado de peticiones o conexiones de usuarios de forma simultánea, en caso de superar ese número, comienzan a ralentizarse o incluso bloquearse y desconectarse de la red. Existen dos técnicas para este ataque: la denegación de servicio o DoS (Denial of Service) y la denegación de servicio distribuido o DDoS (Distributed Denial of Service); la diferencia entre ambos es el número de equipos de cómputo que realizan el ataque. En el primero, las peticiones masivas al servicio se realizan desde una misma máquina o dirección IP, consumiendo así los recursos que ofrece el servicio hasta que no tiene capacidad de respuesta y comienza a rechazar peticiones (denegar el servicio); en el segundo, las peticiones o conexiones se realizan empleando un gran número de computadoras o direcciones IP, todas al mismo tiempo y hacia el mismo servicio objeto del ataque, de forma general, las computadoras que lo realizan se encuentran infestadas, formando parte de una Botnet, y comportándose como zombis.

- Ataque MITM (Man In The Middle): conocido como "hombre en el medio", ocurre cuando una comunicación entre dos sistemas es interceptada por una entidad externa simulando una falsa identidad. En este sentido, el atacante tiene control total de la información que se intercambia, pudiendo manipularla a voluntad, sin que el emisor y el receptor lo perciban rápidamente. Es común que se realice empleando redes WI-FI públicas y abiertas, y es muy peligroso ya que se puede obtener información sensible de las víctimas, y es difícil identificarlo si no se poseen los mínimos conocimientos sobre el tema.

- Rootkit: es un tipo de Malware diseñado para infectar una PC, el cual permite instalar diferentes herramientas que dan acceso remoto al equipo de cómputo. Este Malware se oculta en la máquina, dentro del sistema operativo y sortea obstáculos como aplicaciones antimalware o algunos productos de seguridad. El Rootkit contiene diferentes herramientas maliciosas como un módulo para robar los números de tarjeta o cuentas bancarias, un Bot para ataques y otras funciones que pueden desactivar el Software de seguridad.

- SIM Swapping: es la duplicación del SIM de un número telefónico que permite a un atacante usurpe nuestra identidad, pudiendo autentificarse por medio de SMS en diversos servicios que usen la autentificación de dos pasos, incluyendo los servicios bancarios.

- SIM Jacker: es el envió de un mensaje malicioso al dispositivo destino, que si se abre el enlace adjunto, el dispositivo inteligente queda comprometido y se puede extraer toda la información del mismo incluyendo la ubicación.
- 0-Day (día cero): es una nueva vulnerabilidad para la cual no se han creado parches o revisiones, y que se emplea para llevar a cabo un ataque. El nombre se debe a que no existe ninguna revisión para mitigar el aprovechamiento de la(s) vulnerabilidad(es), estas pueden ser utilizadas para qué troyanos, Rootkits, virus, gusanos y otros Malwares se propaguen e infecten más equipos.
- IP Spoofing: el ciberdelincuente consigue falsear su dirección IP y hacerla pasar por una dirección válida en la cual confiamos, de este modo, consigue saltarse las restricciones y puede hacernos conectar a una Web maliciosa.
- Web Spoofing: consiste en la suplantación de una página Web real por otra falsa. La Web falsa es una copia del diseño original, llegando incluso a utilizar una URL muy similar para que demos nuestras credenciales de acceso.
- Email Spoofing: consiste en suplantar la dirección de correo de una persona o entidad de confianza para solicitarnos datos o que descarguemos archivos maliciosos.
- DNS Spoofing: a través de programas maliciosos específicos y aprovechándose de las vulnerabilidades en las medidas de protección, los atacantes consiguen infectar y acceder a nuestro Router suplantando la DNS (Domain Name System). Así cuando tratamos de acceder a una determinada Web desde el navegador, este nos lleva a otra Web elegida por el atacante.
- Sniffing: se trata de una técnica utilizada para escuchar todo lo que se transmite dentro de una red, de esta forma se monitorea el tráfico y se puede capturar información que viaja de forma no cifrada para analizarla y hacerse de nuestros datos.
- Ataque de inyección de SQL (Structured Query Language): es aprovechar una o más vulnerabilidades de servidores SQL que hace que se divulgue información confidencial de la base de datos que de otra manera

no haría al inyectar código SQL malicioso. Esto representa un riesgo enorme si la base de datos almacena información de identificación personal, como números de tarjetas de crédito, información personal y contraseñas.

- Baiting o Gancho: se sirve de algún medio físico como una unidad USB infectada que el atacante deja al alcance de los usuarios, para que cuando este lo introduzca en su equipo se infecte. También puede usar anuncios en Webs con la que promociona cursos o premios que nos inciten a compartir datos o descargar Software malicioso.

En general, para poder llevar a cabo alguno de estos ataques, los intrusos deben disponer de los medios técnicos, los conocimientos y las herramientas adecuadas, deben contar con una determinada motivación o finalidad, y se tiene que dar además una oportunidad que facilite el desarrollo del ataque, como podría ser un fallo en la seguridad del sistema informático elegido.

3.5 Dicen que si no Pagas por un Producto, Entonces el Producto Eres Tú.

¿Cuál es el modelo de negocio de empresas tecnológicas como Facebook? usar Google, Facebook, Messenger, Instagram y WhatsApp es completamente gratis. Y pese a ello, por ejemplo Facebook saca cada vez más dinero por usuario. Esto es posible gracias al uso que Facebook hace de nuestros datos. Teóricamente, y escándalos por fallos de seguridad al margen, Facebook no vende nuestros datos a terceros, sino que vende a terceros el acceso a nosotros gracias al uso de nuestros datos.

Así, las empresas tecnológicas (no importa si pagamos o no por un servicio o producto) van detectando nuestros gustos e intereses en base al dispositivo desde el que accedemos, las páginas que seguimos, nuestro historial de navegación y otros factores, y crea un perfil sobre cada uno de nosotros. Luego vende espacios de nuestro Feed a las empresas que busquen gente como nosotros (franja de edad determinada, localización, aficiones...). Y ya se está planteando formas de ir más allá mediante acuerdos con terceros, como uno con la banca para poder saber el dinero que tenemos en nuestra cuenta.

Esta red publicitaria que tan optimizada está merece el aplauso por su logro técnico, pero quizás no sea tan plausible si tenemos en cuenta ese rastreo

tan agresivo y la escasa preocupación de las empresas por la privacidad de sus usuarios.

Por otro lado, el uso generalizado de las redes sociales entraña algunos riesgos que, siguiendo recomendaciones básicas, se pueden evitar. Como cualquier comunidad frecuentada por miles de usuarios (o, como sucede a veces con las redes sociales, por miles de millones), se deben conocer los mecanismos de control y de seguridad para poder utilizarlos con fiabilidad para mantener nuestra privacidad y es por eso que el usuario tiene que ser especialmente cuidadoso con el uso que hace de la red social, a continuación daremos algunas recomendaciones:

- No necesitamos informar de todo y a todos, y menos con información sensible. Por ello, cuanta menos información pongamos, mejor.
- A menudo publicamos notas o mensajes en el muro de un amigo. Esto también es visible para otros usuarios de Facebook o de la red social que sea.
- Se dice que una foto vale más que mil palabras. Estamos dando información sobre nuestros hábitos, nuestros movimientos, a posibles atacantes.
- Hay que asegurarse de que nuestro contenido en las redes sociales sea visible solo para amigos y familiares.
- Google, Twitter, Facebook e Instagram usan el geoetiquetado mediante el GPS para ayudar a los usuarios a marcar la ubicación donde se hizo una foto, vídeo y ayudar a que el perfil sea más «social». Cualquier usuario puede interpretar fácilmente información como nuestro estado económico, estilo de vida, lugares frecuentes y la rutina diaria a través de los medios con etiquetas geográficas.
- En una red social lo normal es que cada usuario se identifique con su nombre y apellido real y que aporte datos personales, como si estudia o trabaja, con quién se relaciona o en qué ciudad vive.Esto hace que su exposición pública sea mucho mayor que antes, esto implica la pérdida del anonimato algo que antes era común y habitual en internet.
- En las redes sociales, una vez que se pulsa el botón de "publicar", esa información es enviada a los contactos del usuario. Eso significa que si

más adelante el usuario se arrepiente de lo dicho, publicado o mostrado y trata de borrarlo, solo conseguirá eliminarlo de su propio perfil pero no de internet.

- Quizás lo más importante de estos consejos para mantener la seguridad en redes sociales es el sentido común en lo que publicamos y comentamos en ellas.

Acuerdos de Privacidad la lectura de los acuerdos de privacidad[45] orientará al usuario sobre qué datos se comparten o no, y también se ofrece la opción de seleccionar o anular las opciones de privacidad, seguridad o administrativas escogidas para proteger la cuenta y el dispositivo.

Al registrarte en una cuenta de redes sociales, por defecto, toda la información anotada en un perfil se hace pública, lo que significa que cualquier persona puede acceder al contenido que hayas registrado en una cuenta. Sin embargo, las necesidades y preferencias de privacidad varían de persona a persona. Mientras que algunos usuarios prefieren tener una mayor exposición y así poder promocionar su contenido en redes sociales, otros prefieren incluir muy poca o ninguna información.

Para lograr una mayor protección del usuario y su información, es importante evaluar en qué medida la persona está dispuesta a incluir información personal en su perfil. Por consiguiente, ten en cuenta lo siguiente al:

- Seleccionar un nombre de usuario: el nombre de usuario es el "nombre digital" que una persona se asigna a sí misma o a su organización para ser identificada en línea. Si existe la preferencia de no ser fácilmente identificada en ninguna plataforma, pero poder continuar usando estas redes, la persona puede asignar y usar un seudónimo que puede estar relacionado o no con esa persona. Además, la persona puede cambiar su nombre de usuario en cualquier momento simplemente ingresando

[45] Si de verdad leyéramos los términos y condiciones de uso de plataformas Online seleccionadas (datos de abril 2020), tardaríamos (con una velocidad de 240 palabras por minuto) aproximadamente:

Microsoft 1:03:30 s (15,260 palabras), Spotify 35:48 s (8,600 palabras), Tik Tok 31:24 s (7,459 palabras), Apple 30:30 s (7,314 palabras), Zoom 30:12 s (7,243 palabras), Tinder 25:54 s (6,215 palabras), Uber 25:36 s (5,658 palabras), Twitter 25:30 s (5,633 palabras), LinkedIn 18:06 s (4,346 palabras), Facebook 17:12 s (4,132 palabras), Amazon 14:12 s (3,416 palabras), YouTube 13:42 s (3,308 palabras, Netflix 11:00 s (2,628 palabras), Instagram 9:42 s (2,451 palabras).

a la configuración de su (s) cuenta (s). El nombre de usuario no tiene que ser coherente en todas las redes sociales; estas pueden variar según las preferencias en cada una.

- Incluir una imagen en la cuenta: el usuario tiene la opción de personalizar una cuenta con la inclusión de una foto del perfil. Cuando un usuario prefiere no ser identificado, se sugiere elegir una imagen en la que no pueda ser identificado y cambiarla cuando sea necesario. Ten en cuenta que cuando se usa la misma imagen en todas las redes sociales, la simple búsqueda de imágenes puede llevar a otras cuentas.

- Incluir una ubicación: cuando se activan los servicios de ubicación en la plataforma de redes sociales, estos permiten a los usuarios rastrear el origen de cualquier actividad de medios en línea. Es importante tener en cuenta que una vez que se activa esta función, permanecerá activa hasta que se elija deshabilitarla en la configuración de privacidad. A pesar de que se permitía que esta característica estuviera activa en el pasado, las plataformas tienen la funcionalidad de deshabilitar la ubicación de cualquier contenido que se haya publicado en sus cuentas.

Sin embargo, aunque un usuario active o desactive la función de compartir la ubicación, potencialmente, la ubicación de un usuario podrá ser descubierta por el contenido que comparta o las imágenes que haya elegido para compartir.

3.6 Datos que Recopila Google

Desde el escándalo de Cambridge Analytica[46] en 2018, poco a poco todos nos hemos ido haciendo más conscientes de nuestra privacidad, y cómo las grandes empresas recopilan nuestros datos casi sin que nos demos cuenta. Ya no solo Facebook y las redes sociales, ya que hay otras empresas como Google que pueden recopilar muchos más datos sobre lo que hacemos cada día.

En parte, esto se debe a que es una empresa que ofrece una gran cantidad de servicios gratuitos que utilizamos casi sin pensar, y en otra, porque es la dueña de todo lo que buscamos en la red (si usamos sus productos y servicios).

[46] Se le acusa de tratar de influenciar los resultados de las elecciones presidenciales de 2016 en los Estados Unidos de Norteamérica.

Para hacerme una mejor idea de hasta dónde llega, podemos descargar todos nuestros datos y sí los revisamos es seguro que tendremos una desagradable sorpresa, que nos llevan a la conclusión de que Google sabe mucho más sobre nosotros que la propia Facebook, como por ejemplo, por dónde he viajado y qué aplicaciones utilizo y cuándo.

Algunos de estos datos los puedes encontrar fácilmente en algunas páginas de datos. Pero para otros, hemos utilizado la herramienta Google Takeout para descargar una copia de seguridad de todo lo que tienen sobre nosotros. Ya es bastante significativo que esta copia ocupe varios Gigas. Algo que debes saber es que puedes configurarlo para que muchos de los datos que Google tiene sobre ti se autodestruyan cuando haya pasado determinado tiempo.

Una cosa que hay que tener en cuenta es que Google no tiene por qué estar vendiendo o revisando minuciosamente estos datos, aunque tampoco hay manera de asegurarse de que no lo hagan con alguno. Sin embargo, en algunos casos nos parece excesivo incluso el simple hecho de que recopilen algunos de ellos. Para que sepas de lo que estamos hablando, aquí tienes una lista del tipo de datos personales que he visto que Google recopila.

Lo primero que llama la atención al mirar los datos de Google, es que sabe por dónde te has estado moviendo. Por ejemplo, tiene una función de cronología en la que puedes ver todos los movimientos que ha ido registrando sobre ti cuando tienes habilitado el historial de ubicaciones de Google Maps. Este suele estar activado por defecto en tus dispositivos, lo que quiere decir que si no cambias la configuración deliberadamente seguirá todos tus pasos a través de tu móvil, tableta o cualquier otro dispositivo con el que estés identificado con tu cuenta de Google.

Todos estos datos Google te los va presentando de forma ordenada y cronológica, mostrándote a qué hora has salido de un sitio, el medio de transporte por el que has ido y qué paradas has hecho. Sorprende la minuciosidad con la que Google lo registra todo. Además, en el caso de que te hayas ido de viaje sacando muchas fotos, Google también va a saber dónde y cuándo sacaste las fotografías con tu móvil, y todo te lo va a mostrar de forma ordenada viendo cómo hiciste el camino y dónde hiciste las fotos. Incluso distingue los trayectos que hiciste en coche y los que has hecho a pie. Esto lo hace utilizando los metadatos de tus fotos y cruzándolos con la información de ubicaciones que recopila.

Y más allá de esta herramienta, en la copia de seguridad de Google también puedes encontrar otras sorpresas. Por una parte, hay una carpeta con un historial de ubicaciones, en la que puedes ver las coordenadas por las que

te has movido en los últimos días y la manera en la que lo has hecho. Es como lo de Google Maps, pero con el código en bruto.

También guarda las actividades que hayas registrado utilizando Google Fit. Aquí puedes encontrar dos carpetas diferentes, una con todos los ejercicios que has hecho utilizando esta aplicación, con horas y coordenadas, y otra con un archivo diario en el que se puede ver todos los movimientos agregados en cada momento. Da igual que lleves años sin utilizar la aplicación guarda los datos obtenidos de aplicaciones de terceros al vincularlas con el servicio.

Google también tiene una sección Mi Actividad en la que recoge y almacena todas las búsquedas que haces en Internet, así como el contenido que consumes dentro de portales pertenecientes a Google. Aquí no solo vas a encontrar las búsquedas hechas a través del buscador, sino las búsquedas realizadas en Chrome y tu Android.

La verdad es que al mirar mi cronología no he podido evitar sentirme un poco incómodo, sobre todo porque todos los datos están perfectamente organizados y van acompañados de enlaces. De esta manera, puedes saber que has buscado determinados términos y volver a pulsar sobre ellos para ver los resultados de búsqueda, pero también puedes saber qué perfiles de Twitter o Facebook has visitado o las páginas a las que has entrado.

Además de esto, Google también recopila cuando utilizas otras plataformas como Stadia. Incluso si simplemente pulsas en el botón de Discover de la aplicación de Google para ver los temas que la propia Google considera importantes para ti, va recopilando cuáles son los temas sobre los que te ha mostrado información cada vez.

Es más, dentro de algunas páginas también te dice en qué secciones has navegado. Esto se vuelve un poco más preocupante con el tema de los foros, ya que te dice el título de cada hilo que has visitado dentro de un mismo foro, e incluso te ofrece enlaces para volver a él.

La Web también registra cuántas veces utilizas cada aplicación móvil y lo que es peor, todos estos datos siguen apareciendo aunque borres el historial de Chrome. He probado borrar las páginas a las que he entrado hoy con el navegador, y estas seguían apareciendo en la página de My Activity.

¿Y qué implicaciones tiene esto? Pues no sólo que Google sabe exactamente todo lo que haces en Internet si utilizas sus productos, sino que cualquiera que se ponga frente a tu computadora, tableta o teléfono inteligente (si tiene acceso a tu cuenta) va a poder saberlo, ya que al entrar a My Activity Google no pide que confirme mi identidad. Así que cualquiera puede con relativa facilidad tener acceso a esta información.

En resumen Google tiene acceso a:

- Tu nombre, tu dirección, tu edad, tu correo electrónico. Tu modelo de teléfono, tu proveedor de telefonía celular, tu plan y tu consumo telefónico y de internet.
- Las palabras que usas con más frecuencia dentro de tus correos electrónicos. Todos los correos que hayas escrito o recibido, incluido Spam. Los nombres de tus contactos y sus direcciones y teléfonos.
- Las fotografías que tomas con tu teléfono Android, aunque las hayas borrado y aunque no las subas nunca a ninguna red social. Los sitios a los que vas, dentro y fuera del país; la fecha en la que fuiste y la ruta que tomaste. Qué tan rápido llegaste. La tarjeta de crédito o débito que usas para pagar.
- Todos los sitios de internet que has visitado en Google, con qué frecuencia y lo que viste dentro de cada uno. En qué idioma buscas. A qué hora navegas. Con quién has hablado vía Hangouts. Qué videos te gustan. Qué música oyes, etc.

Para tratar de minimizar nuestra huella digital, si soy usuario de los servicios de Google puedo desactivar todos los servicios de rastreo a los que me da opción en su página de "Administrar tu cuenta de Google", no es notoria la degradación del servicio por dichas desactivaciones. Pero esto no impide que Google y servicios de terceros recolecten y transmitan nuestros datos, solo no se almacenarán en nuestra cuenta, ni tendremos acceso a los mismos.

También puedo prescindir de los servicios de Google usando otras alternativas no tan invasivas como describimos más adelante en este texto.

3.7 ¿Cómo me Protejo?

Algunas normas básicas de ciberseguridad para nuestra seguridad, tener privacidad y evitar la vigilancia son:

- Usar contraseñas largas, robustas (incorporar mayúsculas, minúsculas, números y símbolos) y diferentes para cada servicio que lo solicite, para facilitar el manejo de las contraseñas es recomendable el uso de gestores de contraseñas.

- Cifrar Dispositivos, Discos y Unidades de Respaldo para mantener la confidencialidad de nuestra información.
- Mantener actualizado el sistema operativo y las aplicaciones (sólo instalar las necesarias) de nuestros dispositivos además de usar mecanismos que coadyuven en la seguridad como cortafuegos, antivirus, entre otras aplicaciones de seguridad.
- Generar respaldos periódicos -compactados y cifrados- de nuestra información y garantizar que es posible restituir nuestros datos de dichos respaldos (una buena estrategia de respaldo es la siguiente: mantener tres copias de cualquier fichero importante -una principal y dos respaldos-, mantener los ficheros en dos tipos distintos de almacenamiento para protegerlos ante distintos riesgos y almacenar una copia de seguridad fuera de nuestra casa u oficina).
- Para ciertas actividades en la red o el uso de programas poco confiables es recomendable el uso de máquinas virtuales que limitan los posibles daños por programas maliciosos.
- Al navegar en internet es necesario hacerlo de forma segura para no exponernos de forma innecesaria, mediante un uso seguro y aceptable de las herramientas en la Nube.
- Conocer las medidas mínimas de protección para una navegación segura en internet.
- Conocer las técnicas usadas en ataques de inteligencia social, para no ser víctimas de la ciberdelincuencia.
- Al hacer uso de videoconferencias conocer las políticas de privacidad y las medidas de seguridad para no exponernos a ciberacoso y pérdida de datos.
- Hacer un uso correcto de los dispositivos personales cuando estos son usados para el trabajo.
- Uso de escritorios remotos y virtuales como una forma de mitigar los riesgos cuando se trabaja de forma remota.

Entre otras tantas cosas que el usuario actual de las Tecnologías de la Información y la Comunicación (TIC) debe dominar. Además, debemos conocer algunas recomendaciones útiles para reducir nuestra huella en internet, como:

- No compartir nuestro correo electrónico y número telefónico, pese a que es habitual utilizarlos para registrarnos en sitios Web, si los compartimos libremente por internet nos exponemos a ser víctimas de Spam, Phishing y todo tipo de ciberataques basados en ingeniería social.
- El uso de cuentas de correo alternativas nos permite registrarnos en diferentes sitios Web sin utilizar datos y/o cuentas personales o de trabajo.
- Usar cuentas de correo seguro, anónimo y bidireccional como el servicio de Tutanota, Mailfence o ProtonMail. O usar el modo confidencial -en el cual podemos establecer fecha de vencimiento y contraseña para cada correo- al enviar correos desde cuentas Google.
- Acceder a las opciones del navegador para eliminar cada cierto tiempo la información almacenada en formas de Cookies, Caché o el historial de navegación.
- El uso de modo privado o incógnito[47] en el navegador nos permite no almacenar información sobre las páginas Web visitadas ni se guardan las Cookies, ya que se eliminarán al salir del navegador.
- Evitar revelar información sensible en redes sociales sobre nosotros, familia y conocidos, en medida de lo posible debemos vigilar lo que publicamos, quién puede verlo y configurar las opciones de privacidad.

[47] En Chrome y otros navegadores Web, el modo incógnito -del cual Google sostiene que este modo- está diseñado para evitar el almacenamiento local de datos, y realmente no protege nuestra privacidad, aunque otros usuarios en el mismo dispositivo no verán la actividad en modo incógnito, los sitios Web y servicios, incluidos los de Google, aún pueden recopilar datos. La actividad, como descargas, favoritos y elementos de la lista de reproducción, se almacenará.

Cabe mencionar que ningún navegador ofrece el 100% de privacidad ni anonimato al usuario, lo que si, es que entre los diferentes navegadores existentes, podremos encontrar navegador web (como por ejemplo Brave) que ofrecen capas adicionales de protección de los datos del usuario, pero esto no los vuelve 100% eficientes en ello, pues incluso navegadores como Tor (para la Dark Web) tiene sus fallos.

- El uso de Webs seguras (https y certificados digitales) aseguran que la información que intercambiamos con ellas, como datos bancarios o contraseñas viajen de forma cifrada.
- Usar navegadores especializados (Tor) y lugares de búsqueda de información (DuckDuckGo) que nos permitan una navegación segura al no guardar lo que buscamos, ni los sitios donde accedemos.
- El uso del GPS en nuestros dispositivos móviles es una gran herramienta en nuestra vida digital, pero debemos tenerlo activado sólo cuando sea necesario, ya que muchas aplicaciones comparte nuestra ubicación en tiempo real sin nuestro conocimiento y alguien puede conocer donde vivimos, los lugares que frecuentamos o cuando no estamos en casa.
- Al tomar fotografías o vídeos desactivar el guardado de datos *Exif* (Exchangeable Image File) también conocidos como metadatos: datos sobre la cámara, datos sobre la fotografía y datos sobre el origen como ubicación por GPS, etc.
- Nunca tomar vídeos o fotos comprometedoras, de carácter íntimo o sexual y menos publicarlas, ya que suponen una gran amenaza a nuestra seguridad y pueden tener consecuencias muy graves, como la exposición pública, extorsión o el ciberacoso.
- No compartir fotos, vídeos o audios de menores.
- No debemos compartir vídeos, fotos o audios de terceros sin su aprobación, en especial los que contienen conversaciones privadas que difundan datos personales o información que podría considerarse como revelación de secretos y que la otra persona preferiría no difundir.
- Al emitir opiniones, quejas o comentarios subidos de tono u ofensivos en medios electrónicos nos expone a ser atacados o censurados y en ciertos casos podrían ameritar acciones legales.
- No publicar documentos personales en forma de fotos, vídeos o archivos (*.Docx*, *.PDF*, etc) ya que con su revelación nos expone a una suplantación de identidad y al uso de nuestros datos de forma fraudulenta.

- Cumplir las normas mínimas de protección de dispositivos personales en redes corporativas.
- Hacer uso correcto de escritorios remotos y virtuales en equipos personales para el trabajo remoto.

4 Búsquedas Básicas en la Red

Buscar en la red es principalmente saber usar el buscador de Google, que nos proporciona información de forma rápida y expedita. Lo cual es un ahorro de tiempo, pero requiere de cierta habilidad. Por otro lado, en la época en la que vivimos, el tiempo es uno de los bienes más preciados. Así que seguro que en el futuro alguno de estos trucos te será útil, pues aquí vamos a explicar algunas formas de realizar búsquedas avanzadas en Google utilizando comandos. En concreto, vamos a explicar los comandos que utilizamos en nuestros primeros acercamientos a una página Web cuando tenemos que realizar una consulta SEO (Search Engine Optimization).

4.1 Algunas Búsquedas Básicas Usando Google

Muchos de estos comandos te serán útiles en el día a día de tus búsquedas y otros posiblemente no lo vayas a utilizar muy a menudo a no ser que te dediques al mundo del SEO y el posicionamiento Web. En cualquier caso, yo incluimos todos los que nos parecen interesantes y te animo a que pruebes la utilización de varios de ellos en conjunción, para que veas lo avanzada que puede resultar tu búsqueda.

Búsquedas es común hacer búsquedas como:

> Ejemplo de búsqueda: video marketing infographic

pero para hacerla más selectiva podemos usar:

> Ejemplo de búsqueda: “video marketing” infographic

Búsqueda Dentro de Una Web Para realizar una búsqueda dentro de una Web el comando que se utiliza es site:www.url.com. Si ponemos site sin las 3w’s en Google, el buscador te ofrecerá todos los resultados que hay en una Web incluyendo sus subdominios. Si ponemos la misma búsqueda pero con las 3w’s, entonces el buscador nos excluirá los subdominios. Para buscar un término o una frase exacta dentro de un dominio lo único que tenemos que hacer es realizar la búsqueda site e incluir la frase que queramos a continuación, pudiendo ser ésta también entrecomillada. Ej: Búsqueda de la convocatoria de becas del Ministerio de Asuntos Exteriores.

La búsqueda site también puede tener muchas otras utilidades dentro del mundo SEO como por ejemplo llevar a cabo un análisis de la indexación que Google hace de una página Web:

Ejemplo de búsqueda: site: google.com
Ejemplo de búsqueda: site: www.google.com
Ejemplo de búsqueda: site: www.unam.mx + convotatoria+de+becas
Ejemplo de búsqueda: site: www.unam.mx "convotatoria de becas"

Búsqueda info el atributo info sirve para obtener mucha de la información que vimos más arriba pero de un solo vistazo, pues Google nos agrupa entre 3 y 5 links donde podemos encontrar (el caché de navegación de la Web), páginas similares (related:), páginas que enlazan a esa Web (link:), las páginas indexadas del sitio (site:) y páginas Web que contengan la URL introducida ("URL"):

Ejemplo de búsqueda: info:www.wikipedia.org
Ejemplo de búsqueda: related:ediciones.sm
Ejemplo de búsqueda: link:google.mx

Búsqueda search supongamos que deseamos encontrar algo relacionado con el desarrollo Web en una página (freeCodeCamp), entonces podemos usar:

Ejemplo de búsqueda: search:freecodecamp Web development

Búsqueda en Red Social podemos hacer la búsqueda restringiéndonos a una red social:

Ejemplo de búsqueda: @twitter
Ejemplo de búsqueda: éxito @twitter

Búsqueda por Precio podemos hacer la búsqueda restringiéndonos a un precio:

Ejemplo de búsqueda: cámara $300
Ejemplo de búsqueda: cámara $30 .. $300

Búsqueda Aproximada ¿no estás seguro de qué palabras necesitas escribir en la búsqueda? En lugar de los términos que no conoces, coloca un asterisco (*) y Google lo reemplazará por cualquier palabra.

Este es un atajo importante para encontrar artículos que tienen un título específico, con un numeral, por ejemplo, pero que no recuerdas cuáles son:

Así que si buscas una lista con consejos de SEO:

Ejemplo de búsqueda: * consejos de SEO

Término Exacto supongamos que estás buscando en Google artículos relacionados con Marketing de Contenidos. En lugar de simplemente escribir Marketing de Contenidos en el cuadro de búsqueda de Google, (lo que te va a regresar resultados con los términos marketing, contenidos y marketing de contenidos) es mejor buscar explícitamente el término. Para hacer esto, simplemente adjunta comillas:

Ejemplo de búsqueda: “Marketing de Contenidos”

Búsqueda Definida cuando queremos que Google nos defina un término, la manera de hacerlo es añadir la expresión define: yahoo. He elegido la palabra Yahoo por puro vicio, ya que remontándose al libro de Jonathan Swift, Los Viajes de Gulliver, la definición que Google nos ofrece al buscar Yahoo es la de una persona ruidosa, violenta y grosera. Lo cual me parece bastante gracioso y un tanto irónico:

Ejemplo de búsqueda: define:yahoo
Ejemplo de búsqueda: define: object oriented programming

Búsqueda Exacta, el Comando ("") si añades comillas a una frase, la búsqueda encontrará exactamente lo que escribes y en ese mismo orden:

Ejemplo de búsqueda: "Crisis en Venezuela"

Excluir Palabras, el Comando (-) digamos que quieres buscar contenido acerca de marketing sustentable, pero deseas excluir cualquier resultado que contenga el término ecología. Para ello, simplemente use el signo (–) justo antes de la palabra que deseas excluir, sin espacio:

Ejemplo de búsqueda: marketing sustentable -ecología
Ejemplo de búsqueda: marketing -strategy -estrategies
Ejemplo de búsqueda: marketing -"marketing strategy"
Ejemplo de búsqueda: marketing -"marketing strategy" -"marketing strategies"

El Comando (+) al añadir un más (+) justo antes de una búsqueda estás diciéndole a Google que te muestre exactamente ese término excluyendo sinónimos, acrónimos y demás. Un ejemplo de a lo que me refiero lo encuentras cuando buscas en Google términos como MJ (Google te devolverá Michael Jordan), u OT (en España Google nos devuelve Operación triunfo). Incluyendo el +MJ estás forzando a Google a que sólo te muestre resultados que contengan MJ sin que él realice ninguna interpretación de lo que tu quieres decir con esa búsqueda:

Ejemplo de búsqueda: MJ
Ejemplo de búsqueda: +MJ
Ejemplo de búsqueda: video marketing + "content strategy"

Buscar Palabras Cercanas a Otras el operador AROUND(X) busca palabras cercanas a otras, donde x indica cuántas palabras pueden estar separadas de los dos términos:

Ejemplo de búsqueda: content marketing around(10) video

Buscar Frases Cercanas a Otras el operador AROUND(X) busca frases cercanas a otras, donde x indica cuántas palabras pueden estar separadas de los dos términos:

Ejemplo de búsqueda: "content marketing" around(5) "video"

Esto O Eso de forma predeterminada, cuando realizas una búsqueda, Google incluirá todos los términos especificados en la búsqueda. Si estás buscando uno o más términos que coincidan, puedes utilizar el operador OR. (Nota: El OR tiene que estar en mayúscula):

Ejemplo de búsqueda: "marketing de contenidos" OR "publicidad"
Ejemplo de búsqueda: content marketing (site:edu OR site:com)

El Comando (|) esta barra horizontal viene a significar esto "o" lo otro. Resulta útil cuando estás mirando por múltiples términos dentro de un determinado conjunto de datos:

Ejemplo de búsqueda: ledezma | ledesma

El Comando (*) el asterisco funciona como un comodín. Cuando lo pones delante de una palabra y dentro de una frase le estás diciendo a Google que busque la frase exacta que le pides pero que puede intercambiar la palabra del asterisco por otra:

Ejemplo de búsqueda: Antonio * Ledesma

El Comando (#) podemos hacer búsquedas sobre Hashtahegs:

Ejemplo de búsqueda: #miercolesdechelas

El Comando (~) la tilde funciona para buscar sinónimos, anteponiendo el signo ~ a la palabra que queramos buscar:

Ejemplo de búsqueda: ~software
Ejemplo de búsqueda: holiday ~decorations

El Comando (..) el uso de dos puntos sirve para buscar intervalos, por ejemplo: si queremos buscar películas de James Bond que se estrenaron entre 2008 y 2012, podemos usar:

Ejemplo de búsqueda: James Bond 2008..2012

También podemos hacerlo con precios:

Ejemplo de búsqueda: Computadora 200..800 dólares

Búsqueda del Cache la búsqueda cache es la copia de seguridad que los robots de Google guardan de todas las páginas Webs que han sido indexadas en algún momento por los robots de búsqueda. Desde el punto de vista de SEO es importante porque te muestra exactamente qué es lo que Google ve cuando visita una Web. Ejemplo de la Cache que Google tiene de Quiwiq. Para realizar la búsqueda de la cache, igual que en anteriores ocasiones, lo único que tenemos que hacer es incluir en la barra de navagación:

Ejemplo de búsqueda: cache: unam.mx

Búsqueda en un Dominio puedes restringir la búsqueda a un solo dominio, por ejemplo: google site:ubuntu.com y enter. Estamos restringiendo la búsqueda de la palabra google solamente al dominio ubuntu.com:

Ejemplo de búsqueda: google site:ubuntu.com

Palabras en el Texto si deseas encontrar una página Web en la que todos los términos que estás buscando aparezcan en el texto de esa página (pero no necesariamente uno junto al otro), escribe allintext: seguido inmediatamente por palabras o frases:

Ejemplo de búsqueda: allintext: cancún cielo casa lago

Palabras en el Texto + Título, URL, etc. si deseas encontrar una página Web en la que aparezca un término dentro del texto y otro que aparezca en cualquier otra parte, como el texto, el título o la URL, escribe el segundo término y luego intext: seguido inmediatamente por el otro término:

Ejemplo de búsqueda: Marketing intext: universidades privadas

Palabras en el Título ¿quieres encontrar una página Web con ciertas palabras contenidas en el título (pero no necesariamente una al lado de la otra)? Escribe allintitle: seguido inmediatamente por palabras o frases:

Ejemplo de búsqueda: allintitle: cafe estrategia

Búsqueda de Canciones podemos restringir buscar palabras solamente en el título de la página:

Ejemplo de búsqueda: intitle:index.of + mp3 + "Radiohead" -html -htm -php -asp apache

Búsqueda de Películas si queremos ver algún review sobre alguna película, lo escribimos de esta forma:

Ejemplo de búsqueda: movie:300

Búsqueda Allintext funciona igual que la búsqueda Allintitle pero en esta ocasión la búsqueda sólo nos muestra aquellas Webs que tienen la palabra o palabras clave en el cuerpo de texto de alguna de sus páginas:

Ejemplo de búsqueda: allintext: cafe estrategia

Búsqueda Allinanchor funciona igual que la búsqueda Allintext pero en esta ocasión la búsqueda sólo nos muestra aquellas Webs que tienen la palabra o palabras clave en el ancho de texto de los enlaces de sus páginas:

Ejemplo de búsqueda: allinanchor: cafe estrategia

Palabras en el Title + Texto, URL, etc. ¿deseas encontrar una página Web en la que aparezca un término en el título de esa página y otro término aparezca en cualquier otra parte como en el texto o en la URL? Escribe el término que quieres en cualquier parte de la página, seguido de intitle: seguido por el término que quieres en el título:

Ejemplo de búsqueda: marcas intitle:inspiring marketing

Búsqueda inurl este tipo de búsqueda nos sirve para buscar palabras o expresiones en las URL's de páginas Webs. Aquí he escogido dos buenos ejemplos: URL's con la letra ñ = 603.000. URL's con las 3W's: 25.200 millones de URL's:

Ejemplo de búsqueda: inurl:ñ
Ejemplo de búsqueda: inurl:www
Ejemplo de búsqueda: inurl:ftp - Instructs Google to only display addresses that contain "ftp" in the URL.
Ejemplo de búsqueda: site:moz.com -inurl:www
Ejemplo de búsqueda: site:amazon.com -inurl:https
Ejemplo de búsqueda: site:moz.com filetype:txt -inurl:robots.txt

Palabras en la URL si deseas buscar páginas con tu consulta de búsqueda mencionada en la URL, escribe allinurl: inmediatamente seguido por tu consulta de búsqueda:

Ejemplo de búsqueda: allinurl: luismaram contenidos

Cómo Buscar en un Sitio Web a menudo, deseas buscar en un sitio Web específico, contenido que coincida con una determinada frase. Incluso si el sitio no admite una función de búsqueda integrada, puedes utilizar Google para buscar el término en su sitio. Simplemente utiliza el modificador de sitio: site:somesite.com:

Ejemplo de búsqueda: site: www.luismaram.com "reputación en línea"

Búsqueda Relacionada si deseas encontrar nuevos sitios Web con contenido similar a un sitio Web que ya conoces, utiliza el modificador related: somesite.com:

Ejemplo de búsqueda: related:ford.com

Una Página que Enlaza con Otra Página digamos que deseas buscar todos los sitios Web que citan un artículo de luismaram en su sitio Web. Para ello, utiliza el comando link: seguido inmediatamente por el nombre de una página. Google te dará todas las páginas que enlazan al sitio Web oficial de luismaram. Cuanto más específica sea la URL, mejores resultados obtendrás, lo que quiere decir que lo puedes usar con páginas específicas:

Ejemplo de búsqueda: link: luismaram.com

si quieres eliminar los links desde luismaram.com, usa el –

Ejemplo: link: luismaram.com -luismaram.com

Definiciones de Palabras si necesitas buscar rápidamente la definición de una palabra o frase, simplemente usa el comando define : Puedes escuchar la pronunciación de la palabra pulsando el icono del megáfono:

Ejemplo de búsqueda: define: marketing de atracción

Palabras que Faltan ¿alguna vez has olvidado una palabra o dos de una frase específica, letra de una canción, cita de una película, o algo más? Puedes utilizar un asterisco * como comodín, lo que puede ayudarte a encontrar la palabra que falta en una frase:

Ejemplo de búsqueda: I Knew You * Trouble

Noticias en una Ubicación Específica si estás buscando noticias relacionadas con una ubicación específica, puede usar el comando *location*: para buscar en Google News las historias que vienen de esa ubicación:

Ejemplo de búsqueda: Rogue One location: México
Ejemplo de búsqueda: marketing agency location:paris
Ejemplo de búsqueda: Elon Musk location:San Francisco

Tipos de Documentos Específicos si estás buscando encontrar resultados que sean de un tipo específico, puedes usar el modificador filetype :. Por ejemplo, es posible que desees encontrar sólo presentaciones de PDF relacionadas con el marketing de contenidos:

Ejemplo de búsqueda: "marketing de contenidos" filetype:pdf
Ejemplo de búsqueda: amor ext:doc OR ext:docx

Algunas opciones:

- 7Z: Compressed File
- BMP: Bitmap Image
- DOC: Microsoft Word
- DOCX: Microsoft Word
- DWF: Autodesk
- GIF: Animated Image
- HTM: Web Page
- HTML: Web Page
- JPG: Image
- JPEG: Image
- KML: Google Earth
- KMZ: Google Earth

- ODP: OpenOffice Presentation
- ODS: OpenOffice Spreadsheet
- ODT: OpenOffice Text
- PDF: Adobe Acrobat
- PNG: Image
- PPT: Microsoft PowerPoint
- PPTX: Microsoft PowerPoint
- RAR: Compressed File
- RTF: Rich Text Format
- TXT: Text File
- XLS: Microsoft Excel
- XLSX Microsoft Excel
- ZIP: Compressed File

Traducciones ¿quieres traducir una palabra o frase sencilla de un idioma a otro? No hay necesidad de ir a un sitio Web de traducción: Basta con buscar traducir [palabra] a [idioma].

Ejemplo de búsqueda: traducir krankenwagen a español

Listado Telefónico digamos que alguien te llama a tu teléfono móvil, y no sabes quién es. Si todo lo que tienes es un número de teléfono, puedes buscarlo en Google usando la función de la agenda:

Ejemplo de búsqueda: telefono: 5339-5626

Búsqueda de Código de Área si todo lo que necesitas es buscar el código de área para un número de teléfono, simplemente ingresa el código de área de tres dígitos y Google te dirá de dónde proviene:

Ejemplo de búsqueda:776

Búsqueda de Código Postal si necesitas buscar el código postal de una dirección, simplemente busca el resto de la dirección, incluyendo el nombre de la ciudad o la ciudad y el estado, provincia o país. Los resultados volverán con un código de área (si corresponde) y hasta un mapa (en muchos casos):

Ejemplo de búsqueda: Caballo Calco 33, Coyoacán, México

Rangos Numéricos este es un consejo raramente usado pero muy útil. Digamos que deseas encontrar resultados que contengan cualquiera de un rango de números. Puedes hacerlo utilizando el comando X..Y (en caso de que esto sea difícil de leer, lo que está entre la X y Y son dos puntos). Este tipo de búsqueda es útil durante años (como se muestra a continuación), precios o en cualquier lugar donde desees proporcionar una serie de números:

Ejemplo de Búsqueda: números romanos 1..100
Ejemplo de Búsqueda: evolución tecnológica 1980 .. 1990

Rangos Fechas podemos usar daterange para acotar las búsquedas:

Ejemplo de Búsqueda: marketing daterange:1890 1950

Valores (Símbolo del Ticker) sólo tienes que introducir un símbolo de ticker válido como término de búsqueda, y Google te dará las finanzas actuales y un gráfico en miniatura rápida para la cotización:

Ejemplo de Búsqueda: Ticker GOOG

te dará los resultados de bolsa de Google, si no sabes el ticker googlea Ticker Google o cualquier compañía; si es pública te devolverá su cotización.

Revisar una Compañía en la Bolsa si nos interesa revisar el estado de la bolsa de una compañía, solo tenemos que escribir stock: delante de la compañia en concreto:

Ejemplo de Búsqueda: stock: Evergrande

Calculadora la próxima vez que necesites hacer un cálculo rápido, en lugar de subir el subprograma Calculadora, sólo escribe su expresión en Google:

Ejemplo de búsqueda: 48512 * 1.02
Ejemplo de búsqueda: calculadora

o algo un poco más complejo (un corazón en 3D):

Ejemplo de búsqueda: 5 + (-sqrt(1-x^2-(y-abs(x))^2))*sin(100*((10-x^2-(y-abs(x))^2))) x is from -1 to 2

Clima ¿vas a viajar? La próxima vez que necesites un pronóstico para un área determinada, busca clima seguido de la ubicación. Google te dará los datos en los primeros resultados de búsqueda:

Búsqueda Ejemplo: clima Ixtapa

Hora de Amanecer y Puesta del Sol si tienes curiosidad cuando saldrá el sol y se pondrá ese día en una ubicación específica, haz una simple búsqueda en Google con la palabra amanecer o puesta de sol junto con el nombre de la ubicación:

Ejemplo de búsqueda: amanecer Ciudad de México
Ejemplo de búsqueda: puesta de sol Cuernavaca

Resultados Deportivos y Horarios ¿quieres saber los últimos marcadores deportivos y los horarios futuros de tus equipos favoritos o match-ups? Busca un solo nombre de equipo o dos nombres de equipo y Google usará Google Sports para darte puntajes y horarios antes de los primeros resultados de búsqueda:

Ejemplo de búsqueda: New England Patriots

Búsqueda de Imágenes por Tamaño podemos usar imagesize para acotar la búsqueda de imágenes:

Ejemplo de búsqueda: imagesize:500x400
Ejemplo de búsqueda: anillo imagesize:50x40

Metrónomo si necesitamos un metrónomo, podemos usar el de Google, para ello:

Ejemplo de búsqueda: metrónomo

Ayuda para la Dieta si deseamos conocer las calorías de algún alimento, podemos usar:

Ejemplo de búsqueda: calorías de una pera

4.2 Algunas Funciones y Cosas que Puedes hacer con IA

Para exprimir al máximo la inteligencia artificial (IA) de Google Gemeni[48] tenemos algunas opciones como:

Haz que te hable en el idioma que quieras IA siempre te va a hablar automáticamente en tu idioma. Si eres usuario español, entonces te hablará en tu idioma sin problemas. Sin embargo, también puedes pedirle que te hable en otro idioma en el caso de que lo prefieras.

IA responde tus dudas Como suele ser habitual, la principal función de este y otras inteligencias artificiales conversacionales es la de dar una respuesta rápida a tus dudas. Con esto, IA pretende ser un sustituto del proceso de buscar una duda en Google y buscar entre las respuestas la que más te convenza.

Tres versiones de respuesta a elegir Le haces una pregunta, y IA te responde. Pero a veces puede que la respuesta no te convenza totalmente, y te de pereza volver a escribir el prompt que le pusiste. Para eso, IA te ofrece tres respuestas distintas en la mayoría de preguntas que le hagas. Para hacer esto, debajo de tu prompt pero encima de la respuesta de IA a la derecha verás un botón llamado *Ver otras versiones*. Si pulsas en él, se desplegará una ventana donde puedes ver las tres versiones de respuestas que te ha generado, y tienes también un botón de recargar para volver a generar otras distintas.

[48] (https://gemeni.google.com/)

Explícamelo como si tuviera cinco años Si IA te da una respuesta que te parece demasiado complicada o técnica, también puedes pedirle que te dé una explicación más básica y sencilla. Para eso, tienes que pedirle que te explique algo como si tuvieras cinco años, y que así sea más entendible.

Puede hacerte tutoriales Dentro de las mil y un consultas que puedes hacer, IA también te va a ayudarte a realizar tareas generando pequeños tutoriales paso a paso para ayudarte, aunque de una manera más básica y con una lista de pasos donde tampoco da tantas explicaciones.

IA es tu traductor Puedes usar Google IA como traductor, y es una alternativa bastante decente a Google Translator. Para traducir una palabra es suficiente con escribir algo como Tradúceme la palabra "teléfono" al inglés, cambiando la palabra que he puesto yo por la que tú quieras traducir. También puedes traducir frases o párrafos y textos con una misma estructura.

Te dice la portada de una Web IA es una inteligencia artificial que está conectada a Internet, y una de las ventajas de esto es que te permite interactuar con el contenido que hay Online. Por ejemplo, puedes echar un vistazo a las noticias de la portada de una Web sin tener que entrar en la Web.

Te resume un artículo Online Otra de las mejores maneras de aprovechar que IA que está conectada a Internet es pedirle que te resuma cualquier artículo de cualquier Web. Ya sabes, si te aparece un texto que crees que es demasiado largo en una Web pero el tema te interesa, simplemente le pides que lo resuma. Para hacer esto, solo tienes que escribir un comando como Resume esto: seguido por la URL de la página o el artículo al que te refieras.

También puede traducirte un artículo Online Otra manera en la que IA puede interactuar con el contenido de una Web es traducirte lo que pone en el texto a otro idioma. Para hacerlo, puedes escribir Traduce al inglés el contenido de esta página Web: http://132.248.181.216/acl/, y te escribirá la respuesta.

Escribe correos por ti IA es una inteligencia artificial de texto a texto muy versátil, y como otras parecidas también puede hacer algunas

cosas por ti que te ayuden en tu trabajo. Por ejemplo, puedes pedirle que escriba un correo electrónico, describiéndole sobre lo que quieras que escriba. Esto puedes aplicarlo a correos personales o profesionales. Para hacer esto, tienes que utilizar un comando que sea parecido a "escribe un email para comunicarle a alguien que su pedido ha sido recibido y que procederemos al envío en los próximos días".

Escribe tus guiones de redes sociales Otra de las cosas que IA puede hacer por ti es ayudarte a escribir guiones para tus vídeos y contenidos en redes sociales. Porque a veces puedes tener la idea de sobre qué quieres hacer un vídeo, pero no terminas de dar con la fórmula de cómo estructurarlo todo. Para hacer esto, tienes que escribir un prompt que incluya la plataforma para la que es el contenido y la temática o el tiempo del vídeo. Por ejemplo, puedes escribir "escribe un guión de TikTok explicando el funcionamiento de un router en tres minutos".

Escribe redacciones o pequeños escritos Y cómo no, una de las cosas por las que muchos utilizan este tipo de inteligencias artificiales es para generar textos. No para hacer los deberes de clase, porque los profesores notarán enseguida la diferencia entre lo que escribes tú y lo que no, pero quizá sí para otros contextos. Aquí, solo tienes que pedirle de una manera coloquial lo que quieres obtener, pudiendo incluir detalles sobre la longitud del texto en número de palabras o caracteres. Por ejemplo, puedes pedir "escribe un texto de 2000 caracteres explicando cómo funciona la gravedad".

Puedes cambiar el tono (o acento) del texto Si crees que el texto que te ha generado tiene un tono demasiado impersonal o aburrido, puedes modificarlo para darle un poco de chispa. Por ejemplo, puedes pedirle que te lo escriba con un tono más informal o uno más formal, dependiendo de para qué o quién escribas el texto. También puedes pedirle un tono más gracioso o que use algún acento para que adopte la manera de hablar de alguna región. No necesitas repetirle la pregunta anterior, solo con decirle algo como "escríbemelo con un tono más informal"

Puede mejorar tus textos Y si no necesitas que escriba por ti, sino simplemente ayudarte a que tus textos queden más claros y mejor escritos, entonces IA también te puede ayudar en esto. El prompt a utilizar podría

ser algo como "mejórame el estilo de este texto para hacerlo más claro:", y añadiendo después de los dos puntos el texto.

Crea plantillas para estructurar textos Otra de las cosas en las que IA te puede ayudar sin tener que escribir textos por ti es a estructurarlos. Por eso, puedes pedirle que cree una plantilla o estructura para un tipo de texto que quieras escribir. Por ejemplo, puedes pedirle algo parecido a "Dime una estructura para un texto sobre el iPhone 54 Pro".

Escribe poemas o letras de canciones IA también puede convertirse en tu Cyrano de Bergerac particular, y escribir poemas con los que sorprender o impresionar a alguna persona. Estos poemas pueden ser sobre el tema que tú quieras. Para esto puedes usar un prompt como "Escribe un poema sobre el amor y una inteligencia artificial".

Crea tus partidas de rol Google también es una útil herramienta para organizar partidas de rol, eligiendo la ambientación y reglas. Suelen ser con historias sencillas, a no ser que especifiques eventos concretos, pero reacciona de una manera muy positiva. Para hacer esto, puedes empezar con un comando como "¿Puedes simular una campaña de un juego de rol con reglas de dragones y mazmorras ambientada en mundo disco?"

IA analiza dispositivos para ti ¿Estás pensando en comprar un nuevo móvil o un portátil y te da pereza ponerte a leer reseñas sobre los modelos que te interesan?. En ese caso, IA también puede salir al paso y ayudarte escribiendo un análisis del dispositivo. Solo tienes que escribirle un prompt como "Hazme un análisis exhaustivo del Pixel 70".

Te ayuda a crear un examen Y aunque al relacionar IA con estudios siempre se piensa en alumnos haciendo trampas, quizá algunos profesores también puedan beneficiarse. Básicamente porque IA también es capaz de hacer plantillas de exámenes del nivel de estudios y temática que quieras. Puedes probar usando un comando como "Crea un examen de matemáticas para secundaria"

Te ayuda a crear fórmulas de Excel Si sueles trabajar con hojas de cálculo, también vas a poder pedirle a IA que genere fórmulas para ellas con

las que ahorrarte algún trabajo. Puedes pedirle que sean para Excel, pero también para otras aplicaciones que quieras. Puedes usar comandos como "quiero una fórmula de excel para obtener el valor más alto de un rango de datos"

Cuidado con los códigos de programación Sí, IA también puede hacer códigos de programación por ti. Pero igual que pasa con la mayoría de inteligencias artificiales conversacionales, muchas veces el código será erróneo. Eso sí, al menos puede darte pistas sobre la estructura a seguir, y también puede echarte una mano a la hora de depurar tu propio código por ti.

Gana una nueva habilidad IA también puede enseñarte a hacer algo nuevo, describiendote los pasos que tienes que dar, las cosas que debes tener en cuenta, o las características de la habilidad que estás buscando dominar. Por ejemplo, puedes pedirle algo como "Cómo tocar la guitarra como Ritchie Blackmore",

Te ayuda con guías de compras Si necesitas comprar algún tipo de producto pero no sabes bien cuáles son las principales alternativas que tienes, IA también puede hacerte una lista de compras. Para eso, puedes usar un prompt como "qué zapatillas deportivas comprar para hacer running".

Te pone en forma IA también puede ponerte en forma diseñándote planes de entrenamiento, en los que vas a poder especificar las zonas que quieres ejercitar. Puedes decirle prompts como "Dime ejercicios para fortalecer las piernas", y te responderá explicándote varios ejercicios que puedes realizar.

Te crea menús y recetas Otra de las cosas en las que puede ayudarte es a crear tus menús y recetas. Para ello, podrás pedirle a IA que haga menús completos o te realice una receta concreta que quieras intentar preparar. Por ejemplo, puedes decirle "Tengo que organizar una cena para 2 personas, que no comen frutos secos ni marisco. ¿Me puedes sugerir un menú de 3 platos?", pudiendo modificar esta petición a tus necesidades o determinando ingredientes concretos que tengas para que te de ideas de qué cocinar con ellos.

También puede darte consejos de salud Y aunque una inteligencia artificial nunca va a ser tan fiable como un médico, en sí que es capaz de darte algunas respuestas genéricas sobre medicina. Por ejemplo, puedes preguntarle por síntomas concretos, con comandos como "cómo saber si tengo depresión".

Te sugiere entretenimiento Otras listas que IA puede hacer de ti son de recomendaciones de entretenimiento, ya quieras ver películas, series, o videojuegos, además de libros y cualquier otro tipo de contenido. Por ejemplo, puedes decirle un prompt como "dime las mejores series si te ha gustado Severance". Cuando hagas esto, IA te devolverá un listado explicándote la sinopsis de cada una de las opciones que le hayas pedido. Aunque es suficiente con decirle géneros, si le dices títulos podrás orientar la búsqueda un poco mejor. También puedes especificar plataformas o temáticas.

Crea juegos para ti Además de ayudarte a elegir entretenimiento, IA también puede diseñar pequeños juegos para ti. Por ejemplo, puedes pedirle algo como "Hazme un trivial sobre tecnología", y la IA creará para ti un juego con preguntas y respuestas. Puedes experimentar con distintos tipos de juego que puedes intentar generar.

Te ayuda a planificar tus vacaciones Y para terminar, Google IA también puede ayudarte a planificar tus vacaciones. Por ponerte un ejemplo, puedes decirle algo como "estoy planeando un viaje a Lisboa, dime sitios imprescindibles para visitar", y la inteligencia artificial te recomendará los mejores sitios. También puedes pedirle otras cosas relacionadas con el viaje, como que te de la información básica para viajar a un sitio u otro. También puedes preguntar cuándo es más recomendable viajar a un país concreto, para que sepas lo mejor de viajar allí en cada estación.

5 Búsquedas Avanzadas en la Red

5.1 Qué es un motor de búsqueda

Un motor de búsqueda o buscador es un mecanismo que recopila la información disponible en los servidores Web y la distribuye a los usuarios por medio del proceso de crawling, en el que las arañas de los buscadores mapean los datos almacenados en la red.

Para encontrar tales archivos, los buscadores Web recurren a la identificación de la palabra clave empleada por la persona que realiza la búsqueda y, como resultado, el usuario obtiene una lista de enlaces que direccionan a sitios Web en los que se mencionan los temas relacionados a la palabra clave.

5.2 Tipos de Buscadores

Los principales tipos de buscadores de internet son los siguientes:

- Buscadores jerárquicos
- Directorios
- Metabuscadores

Buscadores Jerárquicos Este tipo de buscadores son interfaces de interrogación textual. Revisan las bases de datos de las páginas Web a través de sus arañas y estas recopilan la información sobre los contenidos compatibles con la búsqueda del usuario.

Una vez realizan la consulta, clasifican los resultados por la relevancia respecto a la búsqueda concreta y según el historial de navegación que tenga el usuario.

Directorios Los buscadores del tipo directorio son enlaces de páginas que se agrupan por categorías. Son muy sencillos, pero requieren de un soporte humano y de continuo mantenimiento para funcionar.

Estos buscadores Web no recorren los sitios ni almacenan los contenidos, sólo agrupan enlaces por categorías y se organizan por fecha de publicación y no por relevancia o concordancia con una búsqueda hecha por el usuario.

Un ejemplo de este buscador de internet es Open Directory Project, también conocido como Dmoz.

Metabuscadores Estas interfaces funcionan haciendo reenvíos de las búsquedas a varios buscadores al mismo tiempo. Es decir, remiten la consulta a otros sitios para analizar los resultados que estos presentan, para así, ampliar la margen de los mismos resultados, presentar sus propias conclusiones y ordenar los enlaces de acuerdo con el orden definido por el sistema estructural del metabuscador.

Un ejemplo de metabuscador es WebCrawler, un desarrollo de InfoSpace Holdings LLC. Este metabuscador indaga entre sitios Web, noticias, imágenes y videos indexados por Google, Yahoo!, Ask.com, Bing, About.com y otros. Dogpile es otro metabuscador de InfoSpace, que en su caso añade resultados de compras (Shopping).

5.3 Buscadores Verticales

Envían sus robots a un número limitado de páginas Web sobre un tema concreto, lo que permite que tanto la obtención de la información como la creación del índice sean más especializados en el sector del que se trata. Al tratarse de un número de fuentes más reducido que en un buscador genérico los buscadores verticales pueden también actualizar su información con mayor frecuencia. Adicionalmente, un buscador vertical ofrece a los usuarios mecanismos de búsqueda avanzada específicamente diseñados para el sector.

Los buscadores verticales o temáticos suponen una nueva generación después de los modelos de búsqueda horizontal como Google y Yahoo. La revista Business Week ha clasificado los buscadores verticales como parte de un "renacimiento tecnológico". En los EE. UU., The New York Times Company y News Corp's Fox Interactive Media han invertido cinco millones de dólares y trece millones de dólares, respectivamente, a cambio de participaciones minoritarias en buscadores verticales.

- Medicina: https://medes.com/Public/Home.aspx
- Empleo: https://www.infoempleo.com/
- Formación: https://becas.universia.net/
- Deportes: https://sportuniverse.com/Home/Buscar

5.4 Cuántos Motores de Búsqueda Existen

Siendo sinceros, es muy difícil no acudir, en primer lugar, al todopoderoso Google para realizar una búsqueda. No obstante, más allá de Google existen un sin fin de buscadores de internet. Bueno, lo que sucede es que en su mayoría no son conocidos y de hecho, casi podemos contar los que conocemos con los dedos de una sola mano. ¿Acaso no es así?

Te proponemos el reto de contar cuántos nombres de buscadores conoces:

1. Google
2.
3.
4.
5.

Estamos casi seguros que si completaste los 5 nombres fue con mucho esfuerzo. O por lo menos así nos sucedió a nosotros en un principio. Pero está bien, para eso está este texto, para contarte que existen cientos de motores de búsqueda a nivel mundial.

Entre los principales buscadores de internet disponibles hoy en día, se encuentran los siguientes:

1. Google
2. Yahoo!
3. Bing
4. Ask
5. AOL
6. Go
7. Live
8. Snap
9. AURA!
10. Duck duck go
11. MSN Search
12. Yippi
13. Boing
14. Dumbfine
15. GoYams
16. MetaGlossary
17. PlanetSearch
18. SearchTheWeb2
19. Terra

20. Web 2.0
21. You
22.¡Y muchos más!

5.5 Principales Motores de Búsqueda

Sí, son centenares, no mentimos. Sin embargo, ahondaremos en los buscadores con mayor relevancia actual, empezando por el conocido Google.

Google el mayor de los buscadores, el más conocido de todos. Aunque con existencia previa, adquirió su éxito a mediados del 2001 y su concepto se basa en el uso de PageRank, el cual tiene como premisa, que las páginas que son más deseadas, estén enlazadas, en mayor cantidad, a otras páginas.

Basa su proceso de búsqueda en encontrar la importancia y concordancia de los contenidos hallados en la Web, con respecto a las palabras que usa el usuario. Para hacer el filtro, Google usa los diferentes algoritmos para determinar el orden de aparición de los sitios.

En los últimos años Google ha tratado de superar el tradicional modelo de búsquedas con la aplicación Discover, un recurso que interpreta el comportamiento del usuario móvil para recomendarle contenidos.

Bing es el segundo buscador de internet a nivel mundial. Es el sistema de búsqueda nativo por el cual los nuevos dispositivos de Microsoft utilizan, a partir de Windows 8.

Yahoo! fue lanzado en el 2004, y más que un buscador, se puede considerar como un portal Web, gracias a que ofrece el servicio de búsquedas Web, de correo electrónico, de noticias, de tienda, agencia de viajes, juegos, entretenimiento, entre otros.

MSN Search para hacer frente tanto a Google como a Yahoo! ha surgido MSN Search. Un servicio en que las búsquedas se hacen de manera fácil y rápida, con la adición de que cuenta con otros buscadores de internet como el correo de Hotmail y noticias.

Ask es un buscador en el que literalmente preguntas para que te arroje la respuesta a la duda.

You o tú en español, es una suerte buscador que hace eje en el servicio al usuario, implícito en su nombre, donde cada búsqueda se vive de una manera diferente, personalizable e incluso se muestra de una manera como nunca antes. El buscador You.com, que al pie del sitio indica Powered by Microsoft, se jacta además de ser una Web neutral en términos de emisiones de carbono, debido a que sus computadoras y servidores funcionan con energías limpias. En cuanto al motor de búsqueda, a diferencia de la mayoría de los demás, muestra los resultados en paneles horizontales llamados "aplicaciones", en lugar de en una lista vertical. A diferencia de como estamos acostumbrados, You exhibe paneles que están ordenados según sitios Web: uno para Twitter, uno para Amazon y uno para YouTube o Soundcloud, etcétera, dependiendo la búsqueda y lo relevante que haya para mostrar. A través de estas búsquedas, los usuarios de You pueden acomodar hacia arriba o hacia abajo cada panel, con el objeto de personalizar los resultados futuros, mostrando primero las fuentes preferidas de cada usuario.

5.6 Para qué Sirven los Motores de Búsqueda

Ya vimos que existen muchos motores de búsqueda, y vale la pena mencionar que cada uno existe para desempeñar una función en especial. Si quieres obtener buenos resultados, te recomendamos conocer de qué trata cada uno de ellos.

Bueno, aunque es cierto que el 80% de las búsquedas las podemos hacer en Google, existen otro tipo de conocimientos específicos o de mayor complejidad, al que solo podemos acceder ingresando a otros buscadores de internet.

De ahí la importancia de conocer la función que desempeñan:

Buscadores para Dudas Sencillas Si estamos haciendo una búsqueda general es bueno optar por lo siguientes motores de búsqueda:

Google con un banco de datos propio Google es el número uno de los buscadores. Ofrece la mejor experiencia de búsqueda a través de su búsqueda rápida, avanzada y de imágenes.

Yahoo! uno de los buscadores más populares de la Internet, se caracteriza por su función de buscador de imágenes, aquí puedes buscar imágenes

por su tamaño, su color predominante y su tipo, que puede ser GIF, retrato, primer plano, etc.

Bing a pesar de no ser tan popular como Google, este motor de búsqueda ofrece sugerencias. Las sugerencias de "wiki", las búsquedas visuales y las búsquedas relacionadas son muy útiles a la hora de solicitar información.

Wikipedia viene de la palabra hawaiana wiki que significa rápido y una referencia al término enciclopedia. Tiene un banco de datos propio de artículos alojados en esta página sobre los más diversos temas. Estos artículos pueden ser editados por los usuarios.

5.7 Buscadores para Obtener Respuestas Rápidas

Wolfram alpha un sitio especial para responder preguntas sobre matemáticas, astronomía, biología, finanzas, física, química.

Factbites un buscador en inglés que sirve de diccionario, para buscar información sobre universidades y bases de datos. Las respuestas que arroja son de carácter académico.

Yahoo respuestas Es un sitio popular y práctico para responder dudas y tareas sencillas.

También existen otros sitios como Quora y Reddit.

Motores de Búsqueda para Trabajos, Informes e Investigaciones
Si lo que buscas son datos o estadísticas, puedes usar sitios como:

Zanran un localizador de datos estadísticos.

NationMaster este buscador es excelente para encontrar datos estadísticos sobre países. Bebe de la información que brinda el Banco Mundial y permite hacer comparaciones en cuanto a distintas métricas e indicadores.

Buscadores que Respetan la Privacidad

Duck duck go es muy similar a Google, no obstante, su forma de presentar los resultados no se centra en la importancia de la página de acuerdo a la información del contenido, sino que ofrece la mejor respuesta a la pregunta que se formule, aunque provenga de un sitio que no esté bien posicionado.

Search Encrypt este buscador alternativo surge para respetar la privacidad de los usuarios. Con Search Encrypt puedes hacer una búsqueda de forma privada y segura, evitando que sean interceptadas.

Para Resultados Académicos o de Investigación Cuando pienses en resultados de corte académico o de investigación científica, recurre a motores de búsqueda como:

Google Académico este es un buscador especializado en literatura científico-académica. Incluye libros, artículos, resúmenes, trabajos de investigación, tesis, informes técnicos, etc.

Deepdyve Este buscador de internet permite acceso a innúmerables publicaciones científicas mediante la suscripción de un plan. Hay una versión gratuita por 14 días.

Cuando Necesites más de un Motor de Búsqueda a la Vez ¿Recuerdas a los metabuscadores? A continuación algunos de ellos:

Metacrawler Este metabuscador, como todos los otros, trae resultados buscando en la base de datos de otros buscadores de internet lo que permite obtener una infinidad de resultados. Es rápido y bastante eficaz.

Dogpile este otro metabuscador combina diferentes resultados de los motores de búsqueda, eliminando resultados repetidos y ordenándolos por su relevancia, trae una lista editable con categorías. Tiene la opción de búsqueda avanzada, filtros de búsqueda o sugerencias de búsquedas recientes.

Copernic un software para instalar en el escritorio, que te permite buscar en más de 1000 motores de búsqueda al mismo tiempo.

Buscadores Ecológicos

Ecosia es un buscador ecológico que destina el 80% de sus ingresos al plantío de árboles. Este buscador está asociado a organizaciones como WeForest por ejemplo, que colabora con sus programas de plantación de árboles.

GigaBlast podríamos decir que este también es un buscador ecológico. El sitio creado en el año 2000 por Matt Wells informa que el 90% de la energía que utiliza es eólica. Las búsquedas en español dejan un poco que desear, pero si quieres hacer una búsqueda en inglés, puede ser una excelente opción.

Motores de Búsqueda para Encontrar Empresas o Datos sobre Empresas Cuando requieras información relacionada a una empresa como el domicilio social, facturación, entre otros:

Kompass para la búsqueda de empresas B2B de todo el mundo.

Duns and Bradstreet un sitio que cuenta con datos financieros de empresas de todo el mundo.

A partir de estas búsquedas, es posible identificar ideas de contenidos vinculados a diferentes segmentos de mercado y analizar características de la competencia.

Buscadores de Redes Sociales

Búsqueda avanzada de Facebook FaceBook se ha convertido en un gigantesco banco de información, por lo que el uso de su búsqueda avanzada se ha hecho necesaria. Al hacer una búsqueda, aparecerá la opción "Ver todos los resultados para...". Si haces clic aquí te aparecerán todas las opciones para restringir tu búsqueda. Una excelente opción no solo para buscar personas, sino cualquier tipo de información que haya sido publicada o compartida en Facebook.

Búsqueda de LinkedIn en LinkedIn puedes hacer búsqueda de personas, empresas, empleos y respuestas. Como usuario puedes hacer búsquedas con ciertas limitaciones para restringirla. Pero si eres Premium, tienes más opciones para una búsqueda avanzada. Para buscar empleos LinkedIn es una excelente opción.

Búsqueda de Twitter Twitter también tiene un motor de búsqueda bastante eficiente. Su búsqueda avanzada te permite restringir bastante lo que quieres encontrar para que los resultados obtenidos sean los más precisos.

Motores de Documentos, Ebooks y Presentaciones

Scribd en este buscador podrás encontrar una gran cantidad de material informativo, como revistas, libros, audiolibros y otros documentos. Incluso puedes encontrar algunas partituras. Tiene una buena variedad de títulos que puedes acceder online o descargar. Para tener acceso ilimitado puedes adquirir una membresía o hacer la prueba por 30 días.

SlideShare es un buscador de presentaciones, Webinars, documentos y libros en PDF. Está asociada con tu cuenta de LinkedIn y tienes acceso a una gran cantidad de materiales de forma gratuita.

ISBNdb un buscador que permite localizar cualquier libro por nombre, autor, género y con el cual podrás notificarte el n^{o}ISBN. De igual modo, para esto, puedes usar sitios como Amazon o los de las editoriales como tal.

Buscadores en Foros

BoardReader si necesitas hacer búsquedas en foros, este buscador puede serte bastante útil.

Buscadores de Blogs

Blog Search Engine es uno de los motores de búsqueda donde puedes encontrar blogs y post blogs colocando la palabra clave. Trae buenos resultados incluso en español.

Motores de Búsqueda para Canciones y Películas

Allmusic el buscador de cantantes y canciones más complejo que existe en el mundo. La herramienta tiene un recurso de búsqueda avanzada que permite encontrar canciones, álbumes y artistas de cualquier país.

Filmaffinity una buena opción para encontrar películas, pues brinda la cartelera de diversos países como México, Uruguay, Colombia, España, Argentina, Chile, Venezuela, Costa Rica, Bolivia, Ecuador, Perú, Estados Unidos y Reino Unido.

IMDB para realizar búsquedas por directores, actores, series, películas.

Buscadores de Imágenes y Videos

Flickr este conocido buscador de imágenes te permite encontrar las más diversas fotos, ilustraciones e incluso videos. Gran parte del contenido de Flickr tiene los derechos reservados, pero puedes encontrar algunas que no lo tienen, o que puedes compartirlas con atribución.

Pixabay es uno de los mejores buscadores de imágenes con licencia para usar su contenido libremente. Permite que encuentres una variedad de fotos, ilustraciones, infografías y figuras en general libres de derechos. Se recomienda solamente atribuir la imagen a su creador, pero no es obligatorio.

Pinterest es más que un buscador de imágenes, es un buscador de contenidos. A través de sus imágenes expuestas puedes llegar a encontrar valiosos contenidos. Aquí encontrarás fotos, infografías, ilustraciones, etc. que podrás archivar en la propia plataforma.

TinEye este buscador de imágenes funciona con búsqueda inversa. Aquí colocas la imagen y el buscador te dice donde encuentras esta imagen online. La búsqueda puede ser hecha de dos formas, con la imagen que deseas encontrar o colocando la URL donde se encuentra la imagen que quieres. Una excelente herramienta para saber si alguien está utilizando tus imágenes sin tu consentimiento.

Creative Commons Media este buscador no es solo de imágenes, también puedes encontrar audios y videos libres de derechos de autor. Si tu búsqueda es para usos comerciales o si quieres hacer una adaptación, puedes restringir tu búsqueda para este tipo de archivos. El buscador accede al banco de datos de otros buscadores y te da los resultados que estás buscando. Están lanzando una versión Beta con más opciones para que puedas realizar tu búsqueda avanzada utilizando una variedad de recursos que te permiten restringir los resultados de acuerdo con lo que necesitas.

WikiMedia Commons es otra excelente opción para buscar imágenes. Aquí puedes encontrar una gran variedad de imágenes que en su mayoría puedes compartir mediante la atribución solamente. En los resultados podrás ver las licencias de cada una de las imágenes para saber como puede ser su uso. Además de imágenes puedes encontrar audios y videos.

360 Daily encontrarás una gran cantidad de videos en inglés. La plataforma todavía no lanza una versión en español, pero si el idioma no es ningún impedimento para ti. Aquí tienes una amplia variedad de opciones.

Buscadores de Menciones y Contenidos

Keyhole este motor de búsqueda te permite obtener un análisis de tu contenido sobre tu marca. Puedes hacer búsquedas de hashtags, palabras clave, URL o menciones de tu marca en Twitter o Instagram.

SocialMention con este buscador puedes hacer un monitoreo de tus redes sociales y descubrir las menciones que tu marca ha tenido.

BuzzSumo este buscador de contenidos te permite descubrir cuáles son tus contenidos con mejor actuación en las redes sociales. La versión gratuita es limitada. Puedes contratar la versión pagada a partir de 79 dólares mensuales.

Motores de Búsqueda de Datos y Estadísticas

SimilarWeb te permite obtener informaciones y estadísticas de cualquier sitio Web. No está disponible en español, pero te analiza cualquier URL independientemente del idioma.

BuiltWith este sorprendente motor de búsqueda permite que accedas a las informaciones sobre la tecnología que un sitio Web utiliza. Con él puedes conocer cuáles son los recursos que un sitio Web o blog utiliza en lo que a tecnología se refiere. Incluso puedes saber el gasto aproximado que la URL hace al utilizar esos recursos.

5.8 ¿Para qué Saber Sobre los Motores de Búsqueda?

Como personas interesadas en la búsqueda de información debemos tener un conocimiento pleno en las diversas áreas del saber.

Como dije en un comienzo, cuando trabajamos en internet debemos buscar la forma de tener la mejor posición en los buscadores Web y muchas veces nos quedamos trabajando solo para Google cuando:

- Existen otros buscadores que nuestro público está usando y estamos desperdiciando su potencial;
- Nuestro tema de negocio puede estar vinculado más hacia una modalidad específica de buscadores y no lo teníamos en cuenta.

¿Qué son las Tendencias de Búsqueda? Tendencia hace referencia a inclinación, disposición, propensión o preferencia; por ejemplo, cuando se habla de las últimas tendencias de moda, se alude a esa prenda, zapatos o color que la mayoría empezará a utilizar debido a que es lo más nuevo e innovador en cuanto a moda se refiere.

Desde esta perspectiva, se entiende como tendencias de búsquedas aquellas inclinaciones que muestran los usuarios al momento de buscar en internet y que se relaciona con algún evento, lugar, persona o cosa de la que todos hablan, o empiezan a hablar cada vez más, y quieren saber porque está ligado a algún tema que resulta interesante en la actualidad.

Tendencias de búsqueda se entiende como 'eso que está, o está empezando a ser, de moda' en determinado momento, pero que luego ya se va quedando atrás. Este concepto no debe confundirse con búsquedas populares. Para entenderlo con mayor claridad, podemos decir que la frase 'mundial Rusia

2018', porque era año de mundial, marcó tendencia en 2018, pero no significa que vuelva a marcar tendencia para 2019; mientras que el término 'Rusia' siempre ha sido popular entre las búsquedas y, aunque no se considere una tendencia, sigue siendo popular y, seguramente también lo será el próximo año.

Lo que intento decir es que las tendencias de búsqueda son precisamente 'eso que está de moda' en determinado momento, pero que luego ya se va quedando atrás; al igual que en la moda, una misma prenda o mismo color no está en tendencia año tras año.

Herramientas para Conocer las Búsquedas Populares y Tendencias
Ya debería estar claro que búsquedas populares y tendencias de búsqueda, aunque se relacionan, no son exactamente lo mismo, ¿cierto? Ahora bien, te explicaré qué herramientas hay disponibles para que conozcas cuáles son los temas, palabras claves y las tendencias que están 'dando de qué hablar' en Google.

Google Trends es la herramienta por excelencia para hacer análisis Web SEO y para analizar tendencias y saber cuáles son las búsquedas más populares en Google. Posee datos de búsqueda desde el año 2004, un plus del que puede hacer alarde.

Su mayor ventaja es que actualiza los datos de forma continua y permite conocer la evolución de una palabra clave, un tema o de términos relacionados con ellos. Puedes comparar los términos de búsqueda por fecha, ubicación o por otros términos, así como te permite filtrar las búsquedas y explorar las tendencias del día, o del último año. ¡Como lo prefieras!

Google Trends puede beneficiar tu negocio y ayudarte a establecer un plan de marketing online basado en la información que la herramienta te suministra. Con ella es posible observar qué tan popular y efectiva es una palabra clave, si está en desuso o desde dónde fue buscada; a la vez que te arroja datos acerca de cómo podría evolucionar dicha palabra o frase clave.

¡No es que tenga una pitonisa adentro! Pero sí es capaz de dar estimaciones concretas.

Google Autocompletado es una sencilla herramienta que todos ya estamos utilizando, pues está activada en el buscador Google y cuando se empieza a escribir la frase o término que se desea buscar, inmediatamente

Google te arroja una serie de opciones relacionadas con tu búsqueda, para simplificar el proceso.

¿Cómo sacar ventaja de Google Autocompletado? ¡Fácil! ¿Recuerdas lo de ser perspicaz?

Aquí deberás serlo, pues puedes aprovechar las sugerencias que Google te muestra para determinada palabra clave y a partir de ellas escoger las más idóneas para tu Web.

Déjame decirte que estas sugerencias no las adivina el buscador, sino que son producto de las búsquedas reales que recibe y de las tendencias que detecta para ciertos temas, por lo que es capaz de 'predecir' o relacionar (para no ser tan místicos) las palabras y así descifrar lo que el usuario quiere antes de que éste lo diga por completo. ¡Saca provecho de esta fantástica herramienta de Google!

Google Ads Keyword Planner es una herramienta perfecta para planificar las palabras clave. Con ella podrás evaluar el rendimiento de tus palabras clave y las de tu competencia, determinar las más buscadas y las más competidas, para que definas qué palabras o frases claves utilizar en tu página Web y su contenido, para conseguir el éxito que buscas en cuanto a la entrada de visitas a tu página Web.

¡Es una aplicación fantástica para planificar tu próxima campaña de marketing digital!

Google News es una especie de herramienta rastreadora donde puedes hallar las noticias más relevantes de todo el mundo.Es una herramienta fantástica a la hora buscar temáticas, sobre los que escribir en tu Blog o Web, sobre noticias relacionadas con tus palabras clave o temas de interés. Además, si eres una empresa de noticias, te interesa hacer SEO en Google Noticias para aparecer allí.

Tendencias en Redes Sociales ¡Las redes sociales mandan! De eso no hay duda. Lo que se vuelve viral en alguna plataforma social, como Twitter, Instagram o Facebook, es importante para toda Web y representa una venta de posibilidades que puede impulsar el éxito de cualquier negocio, así que hay que conocer de qué se habla en ellas.

Tendencias en Twitter conocer las tendencias en Twitter es muy fácil, pues esta red social las arroja por defecto, considerando tu ubicación y a las personas que sigues. Además, Twitter agregó una nueva función llamada 'Momentos', donde podrás hallar una lista con los principales tweets o temas del momento para que no te pierdas de nada.

Tendencias en Facebook Facebook también selecciona los temas más populares o que son tendencia según tus datos, gustos y actividades en su plataforma. Puedes acceder a esta información en News Feed, una sección que se va actualizando con tu información más nueva y la de tus amigos. La información se halla etiquetada por categorías: política, entretenimiento, deportes, etc.

Tendencias en YouTube acceder a las tendencias de YouTube es muy sencillo; al ingresar a la página principal de YouTube encontrarás un menú vertical hacia el lado izquierdo, en cuya parte superior está la sección 'tendencias'. Cuando entres a ella notarás que el contenido se organiza en 4 categorías: música, videojuegos, noticias y películas.

Snapchat es una herramienta de mensajería instantánea muy activa, y te asombrará saber que diariamente tiene más de 100 millones de usuarios conectados, de los cuales se estima que un 60% de ellos generan contenido en esta plataforma todos los días.

Ver las tendencias en Snapchat Discover es muy sencillo, y no te quitará sino unos pocos minutos. ¿Por qué descuidar esta plataforma en ascenso?

Pinterest cuenta con más de 250 millones de usuarios este año, por lo que es una de las redes sociales más importantes y relevantes para cualquier negocio en internet. Puedes ver las tendencias desde su página principal, aunque también existen herramientas externas con las que puedes acceder a ellas.

Google Play tiene más de 5 mil millones de descargas, así que no es una plataforma que se deba ignorar, pues al conocer sus tendencias es fácil conocer qué temas interesan a los usuarios, Desde 2016, Google Play incorporó una sección llamada 'tendencias', por lo que es muy fácil conocer cuáles apps son las más populares.

Buzzsumo se ha convertido en una herramienta amiga para reforzar el SEO de una Web y diseñar campañas de marketing digital; con ella se puede conocer qué tipo de contenido es el más compartido y buscado en las redes sociales, quiénes son los influencers tendencia del momento y qué clase de contenido existe en una página Web específica. ¡Buzzsumo podría darle un giro positivo al timón de tu negocio!

Preguntas en Quora es una herramienta de mucha utilidad para la investigación de palabras clave de cola larga. Recuerda, ya no sólo se trata de investigar palabras o frases clave básicas, sino que ahora Google valora las palabras clave de cola larga o long-tail keywords, algo favorable para tu contenido y tu SEO.

Answer the Public (Responde el público) es una sencilla herramienta que te muestra una variedad de preguntas que giran en torno a tus palabras clave, las cuales pueden dar respuesta a las posibles interrogantes que pudiera hacerse tu público objetivo. Lo mejor de ella, es que utiliza datos de Google y Bing, por lo que puede proveerte información valiosa, bien organizada y fácil de exportar o descargar para ser analizada posteriormente.

Bing Webmaster Tools: Herramienta de Investigación de Palabras Clave es la herramienta de investigación de palabras clave de Bing, por lo que cuenta con datos de búsqueda exclusivo de este buscador. Te será útil para conocer los términos más populares que se buscan en Bing y sus tendencias.

Bing Ads: Consulta de Tendencias y Amplia Concordancia posee una opción para consultar las tendencias y compararlas semana a semana; puedes definir el tipo de análisis filtrando según el público objetivo, el tema o nicho, entre otras variables.

5.9 Averiguar Todo Sobre Cualquier Persona en Internet

A veces no somos conscientes de que nuestro paso por internet va dejando un rastro que se puede analizar. Si estudiamos con detenimiento la información pública, en ocasiones podemos obtener una información valiosa. Una simple

dirección IP pública nos puede ofrecer una gran cantidad de información del usuario. De esta forma, con relativa exactitud obtenemos dónde se encuentra geográficamente, cuál es su proveedor de servicios de Internet y más. No obstante, la recopilación de información ha ido cambiando y han aparecido nuevas técnicas y herramientas como OSINT.

Lo primero que vamos a hacer es explicar qué es OSINT. Luego explicaremos como mediante el uso herramientas como SpiderFoot HX podemos averiguar mucha información sobre una persona en concreto.

¿Qué es OSINT y qué nos Puede Aportar su Utilización? OSINT viene de las siglas en inglés Open Source Intelligence que traducido significa Inteligencia de origen abierto. En este caso nos referimos a un conjunto de técnicas y herramientas que vamos a utilizar para recopilar información pública, analizar datos y luego los correlacionamos para convertirlos en un conocimiento muy provechoso. OSINT es un conjunto de técnicas que se usan como una herramienta muy versátil y que puede utilizarse en ámbitos de Marketing, financieros, policiales y más. Además, si pensamos utilizarla para entornos relacionados con la seguridad informática nos puede ser útil para:

- Para realizar la fase de reconocimiento en pruebas de penetración o Pentesting. Así, podemos averiguar los Hosts de una organización, sacar información del Whois y más.
- La aplicación de técnicas de ingeniería social para buscar información de un usuario en redes sociales y documentos.
- Prevención de ataques informáticos en la que podemos obtener información sobre una amenaza o el potencial ciberataque que pueda recibir nuestra empresa.

En definitiva, gracias a la utilización de OSINT podemos averiguarlo todo sobre un usuario o una organización.

¿Qué es y qué nos Ofrece SpiderFoot? SpiderFoot podemos definirlo como una herramienta de reconocimiento que consulta automáticamente más de 100 fuentes de datos públicas OSINT. Su objetivo es recopilar información sobre direcciones IP, nombres de dominio, correos electrónicos, nombres

y más. Su forma de funcionamiento es sencilla, especificamos un objetivo, escogemos los módulos que vamos a utilizar y continuación, SpiderFoot recopilará los datos y verá cómo se relacionan entre sí.

También tenemos SpiderFoot HX que se basa en la base del módulo de la versión de código abierto para ofrecer una funcionalidad mejorada. Esta versión es de pago y está destinada para los profesionales que desean automatizar OSINT, la inteligencia de amenazas, el descubrimiento de activos o para evaluaciones de seguridad. Entre sus características principales tenemos:

- No necesita instalación ya que está alojado y gestionado en la nube. Simplemente con registrarse ya estaremos listos para utilizarlo.
- Investigación de forma individual utilizando un sólo módulo o realizando el escaneo de múltiples objetivos de manera rápida.
- Monitoreo OSINT en el que podremos ejecutar escaneos automáticamente diariamente, semanalmente, mensualmente o programarlos a nuestro gusto.
- Notificaciones por correo electrónico y Slack cuando se produzcan cambios o finalice el análisis.
- Integración con TOR que nos proporciona que ninguna entidad escaneada sepa que somos nosotros quienes realizamos el escaneo.
- La autenticación de dos factores (2FA) significa que la seguridad de nuestra plataforma e investigaciones OSINT están seguras.
- Perfiles de escaneo personalizados.

En cuanto a esta herramienta, tiene una versión gratuita y otras de pago. La versión gratuita para conocer todo sobre un usuario, en este caso nos permite 3 escaneos por mes, el límite de duración del análisis es de 1 hora y tenemos 1 objetivo por escaneo.

Por qué Proteger tu Privacidad y qué Información Tuya se Puede Encontrar con una Búsqueda en Internet La privacidad de las personas es un derecho. Un derecho que con internet y el tratamiento de datos que hacen ya todos los negocios y servicios no es fácil proteger. O aceptas los términos y condiciones de los servicios en los que quieres crear una cuenta

o no permiten crearla, algunos servicios permiten cambiar algunas opciones para no recibir comunicaciones comerciales por correo electrónico pero el tratamiento de datos está implícito en la aceptación de las condiciones.

Un buscador como Google permite encontrar datos de una persona relacionada con ella incluso pasado algún lustro. A través de las redes sociales es posible conocer cualquier cosa que una persona comparta de forma pública.

Por qué Proteger tu Privacidad internet ofrece muchas posibilidades pero también tiene otros inconvenientes. Uno de ellos es que es muy fácil perder el control de la privacidad. Una vez algo se ha publicado a través de internet es difícil revocar la información y ejercer el derecho al olvido para que esa información que se desea eliminar lo sea. Una vez publicado algo en internet cualquier persona que tenga acceso lo puede copiar o republicar en cualquier otra ubicación en la que ya no se tiene el control. Lo que en un momento se considera como poco importante publicar más adelante puede desearse que no hubiese sido publicado, el problema es que publicar algo es muy fácil pero eliminar algo publicado es muy difícil.

Las redes sociales como Facebook, Instagram o Twitter permiten relacionarse con personas en cualquier parte del mundo, conocidas y desconocidas. En estas redes sociales se publica gran cantidad de información personal que alguien interesado en conocerla le permite un acceso sencillo. Con algo tan simple como publicar el nombre y apellidos de una persona ya es posible encontrar mucha otra información utilizando un buscador como Google, cualquier página pública a la que Google tenga acceso la indexa en su buscador, y a partir de aquí a más información en las redes sociales que esté como pública.

En los términos y condiciones que nadie lee al crear una cuenta en un servicio se detallan las autorizaciones que se otorga a la empresa por el hecho de usar su servicio, en esos términos con una jerga legal difícil de entender y larga para agotar en su lectura se incluyen apartados que suele detallar que el propietario del servicio procesa y compartirá los datos con terceras partes. A partir de compartir estos datos luego uno no sabe porque le llegan llamadas comerciales telefónicas cuando no se ha tenido relación ninguna con esa empresa de la que nos llama el comercial o mensajes SMS al teléfono móvil.

Algunos ataques informáticos son realizados de forma masiva pero otros son llevados a cabo de forma individualizada más laboriosa pero menos, que

alguien que está intentando perpetrar un ataque informático sepa cosas de la víctima le da mayor conocimiento para realizar un ataque de ingeniería social.

Busca Información Tuya a Través de Datos que te Identifican es fácil averiguar cuánta información personal de uno mismo hay publicada en internet con una simple búsqueda en Google cualquier dato personal que identifique a una persona. A partir de uno de estos datos personales es posible encontrar mucha información de lo que hace una persona, que piensa a través de lo que dice, fotos suyas, de sus familiares y amigos, su ciudad y lugar de residencia, de vacaciones, lugar de trabajo y profesión, aficiones, ...

Los datos personales a través de los que averiguar la información que haya en internet de una persona son: nombre y un apellido, nombre y dos apellidos, número de teléfono móvil, dirección de correo electrónico o si utiliza su seudónimo. Si se sabe algo de esa persona como su ciudad de residencia permite descartar coincidencias en caso de obtener varias.

Pon en Google tu nombre y apellidos, tu correo electrónico, tu número de teléfono móvil que son algunas cosas que te identifican y mira la información que encuentras de ti mismo. Cuando se dice que Google conoce más de ti que tú mismo es cierto en el aspecto de que muy posiblemente hay cosas que tu ya has olvidado pero Google es capaz de encontrar.

Y Google también es capaz de reconocer texto en las imágenes. Busca también en Google Imágenes ahí también se encuentran fotos a partir de los datos identificativos.

Quizá te asustes de lo que puede conocer alguien de ti con interés por información que tú mismo has publicado. En el momento de publicar la información quizá no se le da importancia pero en el futuro en alguna circunstancia quizá se desease eliminar esa información cosa que era muy difícil o quizá no sea posible.

Cada uno es libre de hacer con su privacidad lo que mejor le parezca. Nosotros recomendamos protegerla porque es fácil perderla pero muy difícil recuperarla. La privacidad no involucra solo la de uno mismo sino también la de los demás, si tú no le das importancia a tu privacidad otras personas si se la damos por ello si publicas datos o fotos no publiques ninguna que incluya a ninguna persona de la que no tengas su consentimiento.

La razón de utilizar un pseudónimo en internet es para utilizar un nombre que no sea el nombre y apellidos reales, esto permite separar en parte el alter

ego de la persona real y desecharlo en caso de desearlo.

6 Programación y Lenguajes de Programación

En la actualidad, la mayoría de nosotros utilizamos computadoras permanentemente: para mandar correos electrónicos, navegar por Internet, chatear, jugar, escribir textos. Las computadoras se usan para actividades tan disímiles como predecir las condiciones meteorológicas de la próxima semana, guardar historias clínicas, diseñar aviones, llevar la contabilidad de las empresas o controlar una fábrica. Y lo interesante aquí (y lo que hace apasionante a la programación) es que el mismo aparato sirve para realizar todas estas actividades: uno no cambia de computadora cuando se cansa de chatear y quiere jugar al solitario.

Muchos definen una computadora moderna como "una máquina que almacena y manipula información bajo el control de un programa que puede cambiar". Aparecen acá dos conceptos que son claves: por un lado se habla de una máquina que almacena información, y por el otro lado, esta máquina está controlada por un programa que puede cambiar.

Una calculadora sencilla, de esas que sólo tienen 10 teclas para los dígitos, una tecla para cada una de las 4 operaciones, un signo igual, encendido y limpiar, también es una máquina que almacena información y que está controlada por un programa. Pero lo que diferencia a esta calculadora de una computadora es que en la calculadora el programa no puede cambiar.

Un programa de computadora es un conjunto de instrucciones paso a paso que le indican a una computadora cómo realizar una tarea dada, y en cada momento uno puede elegir ejecutar un programa de acuerdo a la tarea que quiere realizar.

Las instrucciones se deben escribir en un lenguaje que nuestra computadora entienda. Los lenguajes de programación son lenguajes diseñados especialmente para dar órdenes a una computadora, de manera exacta y no ambigua. Sería muy agradable poder darle las órdenes a la computadora en castellano, pero el problema del castellano, y de las lenguas habladas en general, es su ambigüedad.

El Mito de la Máquina Todopoderosa Muchas veces la gente se imagina que con la computadora se puede hacer cualquier cosa, que no hay tareas imposibles de realizar. Más aún, se imaginan que si bien hubo cosas que eran imposibles de realizar hace 50 años, ya no lo son más, o no lo serán dentro de algunos años, cuando las computadoras crezcan en poder (memoria, velocidad, unidades de almacenamiento), y la computadora se

vuelva una máquina todopoderosa.

Sin embargo eso no es así: existen algunos problemas, llamados no computables que nunca podrán ser resueltos por una computadora digital, por más poderosa que ésta sea. La computabilidad es la rama de la computación que se ocupa de estudiar qué tareas son computables y qué tareas no lo son. De la mano del mito anterior, viene el mito del lenguaje todopoderoso: hay problemas que son no computables porque en realidad se utiliza algún lenguaje que no es el apropiado.

En realidad todas las computadoras pueden resolver los mismos problemas, y eso es independiente del lenguaje de programación que se use. Las soluciones a los problemas computables se pueden escribir en cualquier lenguaje de programación. Eso no significa que no haya lenguajes más adecuados que otros para la resolución de determinados problemas, pero la adecuación está relacionada con temas tales como la elegancia, la velocidad, la facilidad para describir un problema de manera simple, etc., nunca con la capacidad de resolución.

Los problemas no computables no son los únicos escollos que se le presentan a la computación. Hay otros problemas que si bien son computables demandan para su resolución un esfuerzo enorme en tiempo y en memoria. Estos problemas se llaman intratables. El análisis de algoritmos se ocupa de separar los problemas tratables de los intratables, encontrar la solución más barata para resolver un problema dado, y en el caso de los intratables, resolverlos de manera aproximada: no encontramos la verdadera solución porque no nos alcanzan los recursos para eso, pero encontramos una solución bastante buena y que nos insume muchos menos recursos (el orden de las respuestas de Google a una búsqueda es un buen ejemplo de una solución aproximada pero no necesariamente óptima).

En este trabajo veremos problemas no sólo computables sino también tratables. Y aprenderemos a medir los recursos que nos demanda una solución, y empezaremos a buscar la solución menos demandante en cada caso particular. Por ejemplo:

> Dado un número N se quiere calcular N 33. Una solución posible, por supuesto, es hacer el producto N×N×. . .×N , que involucra 32 multiplicaciones. Otra solución, mucho más eficiente es: Calcular N×N . Al resultado anterior multiplicarlo por sí mismo con lo cual ya disponemos de N^4. Al resultado anterior multiplicarlo por sí mismo con lo cual ya disponemos de N^8.

> Al resultado anterior multiplicarlo por sí mismo con lo cual ya disponemos de N^{16}. Al resultado anterior multiplicarlo por sí mismo con lo cual ya disponemos de N^{32}. Al resultado anterior multiplicarlo por N con lo cual conseguimos el resultado deseado con sólo 6 multiplicaciones. Cada una de estas soluciones representa un algoritmo, es decir un método de cálculo, diferente. Para un mismo problema puede haber algoritmos diferentes que lo resuelven, cada uno con un costo distinto en términos de recursos computacionales involucrados.

Hay muchas aplicaciones a las herramientas computacionales que se han programado y podemos programar, pero nos interesan especialmente aquellas que permitan resolver problemas concomitantes en Ciencia e Ingeniería. Muchas de estas aplicaciones caen en lo que comúnmente se llama cómputo científico. La computación científica es el campo de estudio relacionado con la construcción de modelos matemáticos, técnicas numéricas para resolver problemas científicos y de ingeniería; y su respectiva implementación computacional.

Este campo es distinto a las ciencias de la computación y el procesamiento de información, también es diferente a la teoría y experimentación, que son las formas tradicionales de la ciencia y la ingeniería. El enfoque de la computación científica es para ganar entendimiento, principalmente a través del análisis de modelos matemáticos implementados en computadoras.

Los programas de aplicación de la computación científica a menudo modelan cambios en las condiciones del mundo real, tales como el tiempo atmosférico, el flujo de aire alrededor de un avión, el movimiento de las estrellas en una galaxia, el comportamiento de un dispositivo explosivo, entre otros. Estos programas deberían crear una 'malla lógica' en la memoria de la computadora, donde cada ítem corresponda a un área en el espacio y contenga información acerca del espacio relevante para el modelo. Por ejemplo, en modelos para el tiempo atmosférico, cada ítem podría ser un kilómetro cuadrado, con la altitud del suelo, dirección actual del viento, humedad ambiental, temperatura, presión, etc. El programa debería calcular el siguiente estado probable basado en el estado actual, simulado en medidas de tiempo, resolviendo ecuaciones que describen cómo operan los sistemas mediante el uso de un algoritmo[49], y repetir el proceso para calcular el siguiente estado. Este código o programa se escribe en un lenguaje de programación que sigue

[49] Un algoritmo es un conjunto preescrito de instrucciones o reglas bien definidas, orde-

algún paradigma de programación[50], que posteriormente puede ser ejecutado por una unidad central de procesamiento -computadora-.

Así, una parte importante de la programación es el hecho de conocer y usar uno o más paradigmas de programación. Entonces iniciemos delineando lo que es un paradigma de programación. Los paradigmas difieren unos de otros, en los conceptos y la forma de abstraer los elementos involucrados en un problema, así como en los pasos que integran la solución del problema, en otras palabras, el cómputo. Tiene una estrecha relación con la formalización de determinados lenguajes al momento de definirlos -es el estilo de programación empleado-.

6.1 Paradigmas de Programación

Los Tipos más comunes de paradigmas de programación son[51]:

- Programación imperativa o por procedimientos: es el más usado en general, se basa en dar instrucciones a la computadora de como hacer las cosas en forma de algoritmos. La programación imperativa es la más usada y la más antigua, el ejemplo principal es el lenguaje de máquina. Ejemplos de lenguajes puros de este paradigma serían C, BASIC o Pascal.

- Programación orientada a objetos: esta basado en el imperativo, pero encapsula elementos denominados objetos que incluyen tanto variables como funciones. Esta representado por C++, C#, Java o Python entre otros, pero el más representativo sería el Smalltalk que esta completamente orientado a objetos.

- Programación dinámica: esta definido como el proceso de romper problemas en partes pequeñas para analizarlos y resolverlos de forma lo más cercana al óptimo, busca resolver problemas en $O(n)$ sin usar por tanto métodos recursivos. Este paradigma está más basado en el

nadas y finitas que permiten llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.

[50] Este representa un enfoque particular o filosofía para diseñar soluciones e implementarlas en algún lenguaje de programación.

[51] En general la mayoría son variantes de los dos tipos principales: imperativa y declarativa.

modo de realizar los algoritmos, por lo que se puede usar con cualquier lenguaje imperativo como C, C++, Java o Python.

- Programación dirigida por eventos: la programación dirigida por eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos diseñen, por ejemplo en las interfaces gráficas de usuarios o en la Web.

- Programación declarativa: esta basado en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones. Hay lenguajes para la programación funcional, la programación lógica, o la combinación lógico-funcional. Unos de los primeros lenguajes funcionales fueron Lisp y Prolog.

- Programación funcional: basada en la definición los predicados y es de corte más matemático, está representado por Scheme (una variante de Lisp) o Haskell. Python también representa este paradigma.

- Programación lógica: basado en la definición de relaciones lógicas, está representado por Prolog.

- Programación con restricciones: similar a la lógica usando ecuaciones. Casi todos los lenguajes son variantes del Prolog.

- Programación multiparadigma: es el uso de dos o más paradigmas dentro de un programa. El lenguaje Lisp se considera multiparadigma. Al igual que Python, que es orientado a objetos, reflexivo, imperativo y funcional.

- Programación reactiva: este paradigma se basa en la declaración de una serie de objetos emisores de eventos asíncronos y otra serie de objetos que se «suscriben» a los primeros -es decir, quedan a la escucha de la emisión de eventos de estos- y "reaccionan" a los valores que reciben. Es muy común usar la librería Rx de Microsoft (Acrónimo de Reactive Extensions), disponible para múltiples lenguajes de programación.

El paradigma de programación más utilizado en la actualidad, es el de «orientación a objetos». El núcleo central de este paradigma es la unión de

datos y procesamiento en una entidad llamada «objeto», relacionable a su vez con otras entidades «objeto».

Tradicionalmente, datos y procesamiento se han separado en diferentes áreas del diseño y la implementación de Software. Esto provocó que grandes desarrollos tuvieran problemas de fiabilidad, mantenimiento, adaptación a los cambios y escalabilidad. Con la orientación a objetos y características como el encapsulado, polimorfismo o la herencia, se permitió un avance significativo en el desarrollo de Software a cualquier escala de producción.

Otra parte importante de la programación tiene que ver con el algoritmo a implementar y las estructuras de datos necesarias para soportar los datos del problema, así como su eficiencia. Entonces entenderemos por eficiencia algorítmica para describir aquellas propiedades de los algoritmos que están relacionadas con la cantidad de recursos utilizados por el algoritmo. Un algoritmo debe ser analizado para determinar el uso de los recursos que realiza. La eficiencia algorítmica puede ser vista como análogo a la ingeniería de productividad de un proceso repetitivo o continuo. Con el objetivo de lograr una eficiencia máxima se quiere minimizar el uso de recursos. Sin embargo, varias medidas (e.g. complejidad temporal, complejidad espacial) no pueden ser comparadas directamente, luego, cual de los algoritmos es considerado más eficiente, depende de cual medida de eficiencia se está considerando como prioridad, e.g. la prioridad podría ser obtener la salida del algoritmo lo más rápido posible, o que minimice el uso de la memoria, o alguna otra medida particular.

Entonces, ¿cómo determinar si un algoritmo es mejor que otro?. Algunas pautas pueden ser:

- Facilidad de implementar
- Facilidad de entender
- Facilidad de modificar
- Usa menos memoria
- Menor tiempo de ejecución

Tomando en cuenta todo lo anterior, ¿cómo se hace un programa?

- Hay que tener claro el problema a resolver (*formalización*)

- Se debe planear cómo se quiere resolver o abordar el problema (*análisis y diseño*)
- Se elige uno o más lenguajes de programación y se escriben las instrucciones en ese lenguaje para llevar a cabo esa tarea (*codificación*)
- El texto se compila o interpreta para detectar errores sintácticos y semánticos (*depuración*)
- El programa revisado se prueba con los distintos datos de entrada (*ejecución*)
- Se evalúan los resultados, y de ser necesario se regresa a cualquiera de los pasos anteriores para completar el proceso de corrección (*validación*)

Así, para dar solución a algún problema de nuestro interés, debemos elegir el mejor algoritmo a nuestra disposición, seleccionar[52] el paradigma de programación que nos ofrezca ventajas según su eficiencia algorítmica y con ello usar el lenguaje que nos permita implementar dicho programa usando las técnicas de análisis y diseño que garanticen la calidad de un producto de Software[53].

6.2 Lenguaje de Programación

Entenderemos por un lenguaje de programación, a un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos. Los lenguajes de programación pueden usarse para crear programas que pongan en práctica algoritmos específicos que controlen el comportamiento físico y lógico de una computadora. Esta formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual

[52] Si bien puede seleccionarse la forma pura de estos paradigmas al momento de programar, en la práctica es habitual que se mezclen, dando lugar a la programación multiparadigma o lenguajes de programación multiparadigma.

[53] La ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de Software y el estudio de estos enfoques, es decir, el estudio de las aplicaciones de la ingeniería al Software. Integra matemáticas, ciencias de la computación y prácticas cuyos orígenes se encuentran en la ingeniería.

se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación.

Un programa permite especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos para realizar la construcción de un programa de forma colaborativa.

Variables son títulos asignados a espacios en memoria para almacenar datos específicos. Son contenedores de datos y por ello se diferencian según el tipo de dato que son capaces de almacenar. En la mayoría de lenguajes de programación se requiere especificar un tipo de variable concreto para guardar un dato específico. A continuación, un listado con los tipos de variables más comunes:

Tipo de dato	Breve descripción
Char	Contienen un único carácter
Int	Contienen un número entero
Float	Contienen un número decimal
String	Contienen cadenas de texto
Boolean	Solo pueden contener[54] verdadero o falso

Condicionales son estructuras de código que indican que, para que cierta parte del programa se ejecute, deben cumplirse ciertas premisas; por ejemplo: que dos valores sean iguales, que un valor exista, que un valor sea mayor que otro. Estos condicionantes por lo general solo se ejecutan una vez a lo largo del programa. Los condicionantes más conocidos y empleados en programación son:

- If: Indica una condición para que se ejecute una parte del programa.
- Else if: Siempre va precedido de un "If" e indica una condición para que se ejecute una parte del programa siempre que no cumpla la condición del if previo y sí se cumpla con la que el "else if" especifique.

[54] En el caso de variables booleanas, el cero es considerado para muchos lenguajes como el literal falso ("False"), mientras que el uno se considera verdadero ("True").

- Else: Siempre precedido de "If" y en ocasiones de "Else If". Indica que debe ejecutarse cuando no se cumplan las condiciones previas.

Bucles son parientes cercanos de los condicionantes, pero ejecutan constantemente un código mientras se cumpla una determinada condición. Los más frecuentes son:

- For: Ejecuta un código mientras una variable se encuentre entre 2 determinados parámetros.
- While: Ejecuta un código mientras que se cumpla la condición que solicita.

Hay que decir que a pesar de que existan distintos tipos de bucles, todos son capaces de realizar exactamente las mismas funciones. El empleo de uno u otro depende, por lo general, del gusto del programador.

Funciones estas se crearon para evitar tener que repetir constantemente fragmentos de código. Una función podría considerarse como una variable que encierra código dentro de sí. Por lo tanto cuando accedemos a dicha variable (la función), en realidad lo que estamos haciendo es ordenar al programa que ejecute un determinado código predefinido anteriormente.

Todos los lenguajes de programación tienen algunos elementos de formación primitivos para la descripción de los datos y de los procesos o transformaciones aplicadas a estos datos -tal como la suma de dos números o la selección de un elemento que forma parte de una colección-. Estos elementos primitivos son definidos por reglas sintácticas y semánticas que describen su estructura y significado respectivamente.

Implementación de un Lenguaje de Programación Es la que provee una manera de que se ejecute un programa para una determinada combinación de Software y Hardware. Existen básicamente dos maneras de implementar un lenguaje:

- Compilación: es el proceso que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz interpretar. Los programas traductores que pueden realizar esta operación se llaman

compiladores. Estos, como los programas ensambladores avanzados, pueden generar muchas líneas de código de máquina por cada proposición del programa fuente.

- Interpretación: es una asignación de significados a las fórmulas bien formadas de un lenguaje formal. Como los lenguajes formales[55] pueden definirse en términos puramente sintácticos, sus fórmulas bien formadas pueden no ser más que cadenas de símbolos sin ningún significado. Una interpretación otorga significado a esas fórmulas.

Para ayudar al programador es común el uso de editores de texto o ambientes integrados de desarrollo que permiten resaltar los elementos de la sintaxis con colores diferentes para facilitar su lectura. A la forma visible de un lenguaje de programación se le conoce como sintaxis. La mayoría de los lenguajes de programación son puramente textuales, es decir, utilizan secuencias de texto que incluyen palabras, números y puntuación, de manera similar a los lenguajes naturales escritos. Por otra parte, hay algunos lenguajes de programación que son más gráficos en su naturaleza, utilizando relaciones visuales entre símbolos para especificar un programa.

La sintaxis de un lenguaje de programación describe las combinaciones posibles de los símbolos que forman un programa sintácticamente correcto. El significado que se le da a una combinación de símbolos es manejado por su semántica -ya sea formal o como parte del código duro[56] de la referencia de implementación-.

[55] En matemáticas, lógica y ciencias de la computación, un lenguaje formal es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados. Al conjunto de los símbolos primitivos se le llama alfabeto (o vocabulario) del lenguaje, y al conjunto de las reglas se le llama gramática formal (o sintaxis). A una cadena de símbolos formada de acuerdo a la gramática se le llama fórmula bien formada (o palabra) del lenguaje. Estrictamente hablando, un lenguaje formal es idéntico al conjunto de todas sus fórmulas bien formadas. A diferencia de lo que ocurre con el alfabeto (que debe ser un conjunto finito) y con cada fórmula bien formada (que debe tener una longitud también finita), un lenguaje formal puede estar compuesto por un número infinito de fórmulas bien formadas.

[56] Hard-Code, término del mundo de la informática que hace referencia a una mala práctica en el desarrollo de Software que consiste en incrustar datos directamente (a fuego) en el código fuente del programa, en lugar de obtener esos datos de una fuente externa como un fichero de configuración o parámetros de la línea de comandos, o un archivo de recursos.

La sintaxis de los lenguajes de programación es definida generalmente utilizando una combinación de expresiones regulares -para la estructura léxica- y la Notación de Backus-Naur[57] para la estructura gramática.

Objetivos a Cumplir en el Código Generado Para escribir programas que proporcionen los mejores resultados, cabe tener en cuenta una serie de objetivos, entre los que destacan:

- Corrección: Un programa es correcto si hace lo que debe hacer tal y como se estableció en las fases previas a su desarrollo. Para determinar si un programa hace lo que debe, es muy importante especificar claramente qué debe hacer el programa antes de desarrollarlo y, una vez acabado, compararlo con lo que realmente hace.

- Claridad: Es muy importante que el programa sea lo más claro y legible posible, para facilitar así su desarrollo y posterior mantenimiento. Al elaborar un programa se debe intentar que su estructura sea sencilla y coherente, así como cuidar el estilo en la edición; de esta forma se ve facilitado el trabajo del programador, tanto en la fase de creación como en las fases posteriores de corrección de errores, ampliaciones, modificaciones, etc. Fases que pueden ser realizadas incluso por otro programador, con lo cual la claridad es aún más necesaria para que otros programadores puedan continuar el trabajo fácilmente.

- Eficiencia: Se trata de que el programa, además de realizar aquello para lo que fue creado -es decir, que sea correcto-, lo haga gestionando de la mejor forma posible los recursos que utiliza. Normalmente, al hablar de eficiencia de un programa, se suele hacer referencia al tiempo que tarda en realizar la tarea para la que ha sido creado y a la cantidad de memoria que necesita, pero hay otros recursos que también pueden ser de consideración al obtener la eficiencia de un programa, depen-diendo de su naturaleza -espacio en disco que utiliza, tráfico de red que genera, etc.-.

[57] La notación de Backus-Naur, también conocida por sus denominaciones inglesas Backus-Naur form (BNF), Backus-Naur formalism o Backus normal form, es un metalenguaje usado para expresar gramáticas libres de contexto: es decir, una manera formal de describir lenguajes formales.

- Portabilidad: Un programa es portable cuando tiene la capacidad de poder ejecutarse en una plataforma, ya sea Hardware o Software, diferente a aquella en la que se elaboró. La portabilidad es una característica muy deseable para un programa, ya que permite, por ejemplo, a un programa que se ha desarrollado para sistemas GNU/Linux ejecutarse también en la familia de sistemas operativos Windows. Esto permite que el programa pueda llegar a más usuarios más fácilmente.

- Integridad: Un programa tiene integridad si es posible controlar su uso, el grado con que se puede controlar el acceso al Software o a los datos a personas no autorizadas denotará su integridad.

- Facilidad de uso: Un programa tiene facilidad de uso si el esfuerzo requerido es mínimo o moderado para aprender el manejo de una aplicación, trabajar con ella, introducir datos y conseguir resultados.

- Facilidad de mantenimiento: Un programa tiene facilidad de mantenimiento si el esfuerzo requerido para localizar y reparar errores es moderado o mínimo.

- Flexibilidad: Un programa tiene flexibilidad si el esfuerzo requerido para modificar y/o añadir funcionalidades a una aplicación en funcionamiento es moderado o mínimo.

- Reusabilidad: Es el grado en que un programa o porción de este es reusable en otras aplicaciones.

- Interoperatibilidad: Un programa debe poder comunicarse con otras aplicaciones o sistemas informáticos, el esfuerzo necesario para ello medirá su interoperatibilidad.

Existe una gran variedad de lenguajes de programación y su grado de uso depende de diversos factores, entre los que destacan: C, C++, Java y Python.

6.3 Conceptos Transversales

En esta sección comentaremos algunos conceptos transversales a los distintos paradigmas de programación y a los lenguajes que los implementan.

Abstracción Entendiendo un sistema como una abstracción de la realidad, los datos son las entidades que representan cada uno de los aspectos de la realidad que son significativos para el funcionamiento del sistema. Para que tenga sentido como abstracción de la realidad, cada dato implica un determinado valor y requiere de una convención que permita representarlo sin ambigüedad y procesarlo de una manera confiable.

En la misma línea, la lógica del procesamiento de los datos es también una abstracción de los procesos que suceden en la realidad que conforma el dominio de la aplicación. El proceso de abstraerse progresivamente de los detalles y así manejar niveles de abstracción, es el que permite construir sistemas complejos.

Las unidades de Software que realizan la funcionalidad del sistema requieren también de convenciones y criterios de ordenamiento y articulación interna tanto para un funcionamiento confiable y eficiente del sistema, como para que el proceso de construcción y mantenimiento del Software sea de la forma más simple posible.

Modularización, Encapsulamiento y Delegación Una estrategia central de la programación es buscar la manera de organizar y distribuir la funcionalidad de un sistema complejo en unidades más pequeñas de Software con que se responsabilizan de tareas específicas y que interactúan entre ellas. Estas unidades reciben nombres diferentes según cada lenguaje de programación, como rutinas, funciones, procedimientos, métodos, predicados, subprogramas, bloques, entidades, siendo "módulos" una de las más frecuentes y que dan origen al término.

La clave de cada módulo es no conocer el funcionamiento interno de los demás módulos con los que interactúa, sino sólo su interfaz, es decir, la forma en que debe enviarle información adicional en forma de parámetros y cómo va a recibir las respuestas. Esta propiedad recibe diversos nombres, como el de encapsulamiento, ocultación de información o "caja negra". Ante la modificación de una funcionalidad en particular del sistema, en la medida que su implementación esté encapsulada en un módulo, el impacto que produce su cambio no afectará a los otros módulos que interactúan con el.

En concordancia con la distribución de responsabilidades entre las diferentes unidades de Software, la delegación consiste en la invocación que desde un módulo se efectúa a otro módulo, de manera que el que invoca indica de forma explicita qué es lo que pretende y el que es invocado se ocupa de todo lo

necesario para realizarlo. Puede realizarse de numerosas maneras, variando el criterio de distribución de responsabilidades, el modo de evaluación, la forma de paso de parámetros, los tipos de datos que utiliza, de acuerdo a las posibilidades y restricciones de cada lenguaje y paradigma.

Declaratividad La declaratividad, en términos generales, se basa en la separación del conocimiento sobre la definición del problema con la forma de buscar su solución, una separación entre la lógica y el control.

En un programa declarativo se especifican un conjunto de declaraciones, que pueden ser proposiciones, condiciones, restricciones, afirmaciones, o ecuaciones, que caracterizan al problema y describen su solución. A partir de esta información el sistema utiliza mecanismos internos de control, comúnmente llamado "motores", que evalúan y relacionan adecuadamente dichas especificaciones, la manera de obtener la solución. De esta forma, en vez de ser una secuencia de órdenes, un programa es un conjunto de definiciones sobre el dominio del problema.

Basándose en la noción de delegación, la declaratividad plantea como criterio para distribuir las responsabilidades, separar las relacionadas con modelar o definir el conocimiento del problema de aquellas de manipular ese conocimiento para alcanzar un objetivo concreto. En otras palabras, distinguir el "qué" del "cómo".

La declaratividad brinda la posibilidad de usar una misma descripción en múltiples contextos en forma independiente de los motores que se utilicen.

Permite focalizar por un lado en las cuestiones algorítmicas del motor, por ejemplo para trabajar en forma unificada sobre eficiencia, y por otro en la definición del dominio del problema y la funcionalidad de la aplicación en sí.

La noción opuesta, aunque en cierta medida complementaria, de la declaratividad, puede denominarse "proceduralidad". Los programas procedurales se construyen indicando explícitamente la secuencia de ejecución en la que se procesan los datos y obtienen los resultados. Para ello se detalla un conjunto de sentencias, ordenadas mediante estructuras de control como decisiones, iteraciones y secuencias, que conforman "algoritmos".

En un sistema complejo, no se puede hablar de declaratividad o proceduralidad como conceptos excluyentes o totalizantes, sino que coexisten y se relacionan en una permanente tensión. Dependiendo de los lenguajes y de las herramientas que se utilicen, y en particular del diseño del sistema, habrá

partes del sistema que por su sentido o ubicación dentro del sistema global serán más declarativas o procedurales que otras, logrando con ello aprovechar las ventajas respectivas.

Tipos de Datos Un tipo de dato, o como también es llamado, un tipo abstracto de dato, es un conjunto de valores y de operaciones asociadas a ellos. La utilización de diversos tipos de datos permite la agrupación o clasificación del gran volumen y variedad de valores que es necesario representar y operar en un sistema, según sus semejanzas y diferencias.

Tomando como criterio las similitudes en cuanto al contenido de lo que representan, una primera condición es la existencia de un conjunto de valores o entidades homogéneos en su representación. Otra condición es que los elementos del mencionado conjunto se comporten en forma uniforme respecto a una serie de operaciones.

Cada paradigma de programación, y en particular cada lenguaje, tiene su forma de determinar tanto la conformación de cada tipo de dato, con sus valores y operaciones, como la forma en que se relacionan entre sí conformando un sistema de tipo de datos.

En un lenguaje fuertemente tipado, toda variable y parámetro deben ser definidos de un tipo de dato en particular que se mantiene sin cambios durante la ejecución del programa, mientras que en uno débilmente tipado no, sino que pueden asumir valores y tipos de datos diferentes durante la ejecución del programa.

El tipo de dato al que pertenece una entidad determina la operatoria que se puede realizar con el. Una tarea que realizan muchos de los lenguajes de programación como forma de su mecanismo interno es el chequeo del tipo de dato de las entidades del programa. Esta acción se realiza de diferentes maneras, con mayor o menor flexibilidad, y en diferentes momentos, como la compilación o la ejecución del programa, y en otros no se realiza.

De todas maneras, el tipo de datos permite entender qué entidades tienen sentido en un contexto, independientemente de la forma de chequeo o si el tipado es débil o fuerte. Por ejemplo, a un bloque de Software que recibe como argumento una variable, conocer de qué tipo de dato es, le permite saber qué puede hacer con ella.

Estructuras de Datos Los valores atómicos, es decir, aquellos que no pueden ser descompuestos en otros valores, se representan mediante tipos de

datos simples.

En contrapartida, en un tipo de dato compuesto los valores están compuestos a su vez por otros valores, de manera que conforma una estructura de datos. Cada uno de los valores que forman la estructura de datos corresponde a algún tipo de dato que puede ser tanto simple como compuesto.

Su utilidad consiste en que se pueden procesar en su conjunto como una unidad o se pueden descomponer en sus partes y tratarlas en forma independiente. En otras palabras, las estructuras de datos son conjuntos de valores.

Polimorfismo y Software Genérico El polimorfismo es un concepto para el que se pueden encontrar numerosas y diferentes definiciones. En un sentido amplio, más allá de las especificidades de cada paradigma y del alcance que se le dé a la definición del concepto desde la perspectiva teórica desde la que se le aborde, el objetivo general del polimorfismo es construir piezas de Software genéricas que trabajen indistintamente con diferentes tipos de entidades, para otra entidad que requiere interactuar con ellas; en otras palabras, que dichas entidades puedan ser intercambiables. Mirando con mayor detenimiento los mecanismos que se activan y sus consecuencias para el desarrollo de sistemas, se pueden distinguir dos grandes situaciones.

Hay polimorfismo cuando, ante la existencia de dos o más bloques de Software con una misma interfaz, otro bloque de Software cualquiera puede trabajar indistintamente con ellos. Esta noción rescata la existencia de tantas implementaciones como diferentes tipos de entidades con que se interactúe.

Una entidad emisora puede interactuar con cualquiera de las otras entidades de acuerdo a las características de la interfaz común, y la entidad receptora realizará la tarea solicitada de acuerdo a la propia implementación que tenga definida, independientemente de las otras implementaciones que tengan las otras entidades. Consistentemente con la noción de delegación, la entidad emisora se desentiende de la forma en que las otras entidades implementaron sus respuestas, ya sea igual, parecida o totalmente diferente.

Analizando el concepto desde el punto de vista de las entidades que responden a la invocación, el proceso de desarrollo debe contemplar la variedad y especificidad de cada una para responder adecuadamente a lo que se les solicita. Desde el punto de vista de la entidad que invoca, el proceso es transparente, y es en definitiva esta, la que se ve beneficiada por el uso del concepto.

Otra forma de obtener un bloque de Software genérico es ante el caso en

que las entidades con las que el bloque quiere comunicarse, en vez de tener diferentes implementaciones, aún siendo de diferente tipo, sean lo suficientemente similares para compartir una misma y única implementación.

Desde el punto de vista de la entidad que invoca, el proceso continúa siendo transparente y sigue aprovechando los beneficios de haber delegado la tarea y haberse despreocupado de qué tipo de entidad es la receptora. En este caso, el concepto de polimorfismo se relaciona o se basa en la coexistencia de herencia, de variables de tipo de dato o en las formas débiles de declaración y chequeo de tipos de datos, dependiendo de las diferentes herramientas de cada paradigma.

Asignación y Unificación La asignación destructiva es una operación que consiste en cambiar la información representada por una variable, de forma tal que si se consulta su valor antes y después de dicha operación, se obtiene un resultado distinto. Estas asignaciones se realizan repetitivamente sobre la misma celda de memoria, remplazando los valores anteriores.

La asignación determina el estado de una variable, que consiste en el valor que contiene en un momento en particular. La asignación destructiva es la forma más usual de provocar efecto de lado, pero no la única, ya que, dependiendo de los lenguajes, hay otro tipo de instrucciones que también permiten modificar el estado de información de un sistema.

La unificación es un mecanismo por el cual una variable que no tiene valor, asume un valor. Una vez unificada, o "ligada", como también se le dice, una variable no cambia su valor, por lo que no existe la noción de estado. La duración de la unificación está condicionado por el alcance que tienen las variables en cada lenguaje en particular, pudiendo una variable unificarse con varios valores alternativos en diferentes momentos de la ejecución del bloque de Software en el que se encuentran. Se suele denominar como "indeterminada" a una variable sin ligar o no unificada.

Modo de Evaluación Los parámetros que se utilizan en la invocación de un bloque de Software cualquiera pueden ser evaluados en diferentes momentos de acuerdo al modo de evaluación que utilice el lenguaje de programación.

La evaluación ansiosa consiste en que los argumentos son evaluados antes de invocar al bloque de Software y es responsabilidad de la entidad que los invoca.

La evaluación diferida plantea que la evaluación de los argumentos es res-

ponsabilidad del bloque de Software invocado, quien decide el momento en que lo hará. Provoca que se difiera la evaluación de una expresión, permitiendo que en algunos casos, de acuerdo a cómo sea la implementación, no sea necesario evaluarla nunca, con el consiguiente beneficio en términos de eficiencia.

La evaluación diferida es utilizada en cierto tipo de situaciones que serían consideradas erróneas o imposibles de resolver con la evaluación ansiosa, como por ejemplo los bucles o las listas infinitas.

Orden Superior Asumiendo un esquema de un único orden, en un programa existen por un lado datos y por otro los procedimientos -y todo bloque de Software- que trabajan con ellos, de manera tal que los proce-dimientos reciben datos como parámetros y devuelven datos como resultados. La noción de orden superior plantea que los procedimientos son tratados como datos y en consecuencia pueden ser utilizados como parámetros, representados en variables, devueltas como resultados u operados en cálculos más complejos con otros datos. Se denomina de orden superior a los procedimientos que reciben como argumentos a otros procedimientos.

Recursividad La recursividad, entendida como iteración con asignación no destructiva, está relacionada con el principio de inducción. En general, un bloque de Software recursivo se define con al menos un término recursivo, en el que se vuelve a invocar el bloque que se está definiendo, y algún término no recursivo como caso base para detener la recursividad.

6.4 Algo de Programación

Tipos de datos Dependiendo del lenguaje (véase [7], [8], [9] y [10]), la implementación del mismo y la arquitectura en la que se compile/ejecute el programa, se tienen distintos tipos de datos, pero los básicos son[58]:

- char, 8 bits, rango de -128 a 127
- unsigned char, 8 bits, rago de 0 a 255
- int, 16 bits, rango de -32,768 a 32,767
- unsigned int, 16 bits, rango de 0 a 65,535
- long int, 32 bits, rango de -2,147,483,648 a 2,147,483,647
- long, 64 bits, rango de -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
- float, 32 bits, rango de -3.4x10^{38} a 3.4x10^{38}, con 14 dígitos de precisión
- double, 64 bits, rango de -1.7x10^{308} a 1.7x10^{308}, con 16 dígitos de precisión
- long double, 80 bits, 3.4x10^{-4932} a 3.4x10^{4932}, con 18 dígitos de precisión
- boolean, 1 bit, rango de True o False

La conversión entre los diferentes tipos de datos esta en función del lenguaje y existe una perdida de datos si se hace de un tipo de dato que no "quepa" en el rango del otro; estos y otros errores de programación han generado graves errores en el sistema automatizado de control que han desembocado por ejemplo en, misiones fallidas de cohetes espaciales y su carga en numerosas ocasiones[59], la quiebra de la empresa Knight Capital 2012, sobredosis de radioterapia en 1985 a 1987, la desactivación de servidores de Amazon en el 2012, el apagón en el noreste de EE.UU. en el 2003, fallo en el sistema de reserva de vuelos de American Airlines en el 2013, fallo en el sistema de misiles estadounidense Patriot en Dhahran en el 1991, etc.

[58] La precedencia de los operadores básicos son: $=$ el de mayor procedencia, siguen $-$ y $+$ unario, luego $*, /, \%$ y finalmente los operadores $+$ y $-$.

[59] Una de las pérdidas, fue el satélite mexicano Centenario en mayo del 2015.

Condicionales El flujo de programas a menudo tiene que dividirse. Es decir, si una condición se cumple, se hace algo, y si no, se hace otra cosa. Los enunciados condicionales permiten cambiar el comportamiento del programa de acuerdo a una condición dada, la estructura básica de un condicional es:

Pseudocódigo:

```
Si <condición> entonces
   <acción>
```

Ejemplo en C, C++ y Java[60]:

```
if (x > 0) x*= 10;
```

```
if (x > 0) {
   x *= 10;
}
```

Ejemplo en Python:

```
if x > 0:
   x = x * 10
```

La estructura completa del condicional es:

Pseudocódigo:

```
Si <condición> entonces
   <acción>
else
   <acción>
```

Ejemplo en C, C++ y Java:

```
if (x % 2 == 0) x = x * x;
else x = x * 2;
```

```
if (x % 2 == 0) {
   x = x * x;
} else {
   x = x * 2;
}
```

[60] En condicionales, tienen mayor precedencia $<, >, >=, <=$, luego están $==$ y $!=$, finalmente && (AND) y || (OR).

Ejemplo en Python:

```
if x % 2 == 0:
   x = x * x
else:
   x = x * 2
```

La estructura condicional encadenada es:
Pseudocódigo:

```
Si <condición> entonces
   <acción>
else if
   <acción>
else
   <acción>
```

Ejemplo en C, C++ y Java:

```
if (x < y) {
   z = x;
} else if (x > y) {
   z = y;
} else {
   z = 0;
}
```

Ejemplo en Python:

```
if x < y:
   z = x
elif[61] x > y:
   z = y
else:
   z = 0
```

[61] elif es una contracción de "else if", sirve para enlazar varios "else if", sin tener que aumentar las tabulaciones en cada nueva comparación.

Condicionales con operadores lógicos, tenemos dos operadores lógicos el AND (&&) y el OR (||), ejemplos:

Pseudocódigo:

```
Si <condición operador condición> entonces
   <acción>
else
   <acción>
```

Ejemplo en C, C++ y Java:

```
if (x % 2 == 0 || x > 0) x = x * x;
else x = x * 2;
```

```
if (x % 2 == 0 && x > 0) {
   x = x * x;
} else {
   x = x * 2;
}
```

Ejemplo en Python:

```
if x % 2 == 0 and x > 0:
   x = x * x
else:
   x = x * 2
```

Bucle for La implementación de bucles mediante el comando *for* permite ejecutar el código que se encuentre entre su cuerpo mientras una variable se encuentre entre 2 determinados parámetros, la estructura básica es:

Pseudocódigo:

```
for <inicio; mientras; incremento/decremento> entonces
   <acción>
```

Ejemplo en C, C++ y Java:

```
for (x = 0; x < 21; x++) {
   z = z + x;
}
```

Ejemplo en Python:

```
for x in range(0, 21):
  z = z + x
for i in range(1,3):
  if n == 3:
    break
else[62]
  print("no se encontro el numero 3")
```

Bucle for para navegar en por ejemplo Listas Si se tene una lista declarada, es posible hacer un recorrido en sus elementos usando *for*, por ejemplo:

Ejemplo en Java:

```
List<String> lista = new ArrayList<String>();
...
for (String elem : lista) {
    System.out.print(elem);
}
```

Ejemplo en Python:

```
lista = ["uno","dos","tres","cuatro"]
for elem in lista:
    print elem
```

Bucle while La implementación de bucles mediante el comando *while* permite ejecutar el código que se encuentre entre su cuerpo mientras una condición se cumple, la estructura básica es:

Pseudocódigo:

```
while <condición> entonces
  <acción>
```

Ejemplo en C, C++ y Java:

[62] La instrucción *else* sólo se ejecutará si concluye normalmente el ciclo *for*, es decir, sin la interrupción por *break*. En este caso nunca se ejecutara el *print*.

```
while (z < 20) {
  z = z + x;
}
```

Ejemplo en Python:

```
while z < 20:
  z = z + x
else[63]
  print("concluyo")
```

Bucle do-while La implementación de bucles mediante el comando *do-while* permite ejecutar el código que se encuentre entre su cuerpo y después revisa si se cumple la condición para continuar el ciclo, se puede usar *do-while* en C, C++ y Java, pero no esta presente en Python (pero siempre se puede usar un *while* para emularlo). La estructura básica es:

Pseudocódigo:

```
do
  <acción>
while <condición>
```

Ejemplo en C, C++ y Java:

```
do {
  z = z + x;
} while (z < 20);
```

Condicional switch Cuando se requiere hacer un condicional sobre una variable y que esta tenga varias alternativas, una opción para evitar una cadena de sentencias *if-else*, se puede usar *switch* en C, C++ y Java, pero no esta presente en Python (siempre se puede usar *if-elif* para emularlo). La estructura básica es:

Pseudocódigo:

[63] La instrucción *else* sólo se ejecutará si concluye normalmente el ciclo *while*, es decir, sin la interrupción por *break*. En este caso al terminar el *while* se ejecutara el *print*.

```
switch(expresion) {
   case constante:
      <acción>
      break;
   case constante:
      <acción>
      break;
   .
   .
   .
   default:
   <acción>
}
```

son opcionales los *break* y el *default*.

Ejemplo en C, C++ y Java:

```
switch(i) {
   case 1:
      x=23;
   case 2:
      x ++;
      break;
   default:
   x=0;
}
```

Primer ejemplo Como ya se ha hecho costumbre, el primer ejemplo de un programa es: *Hola Mundo*, así que iniciemos con ello creando el archivo correspondiente *hola.ext*[64] en cualquier editor de texto o en un IDE, entonces:

Ejemplo en C:

```
#include <stdio.h>
int main(void) {
   printf("Hello World\n");
   return 0;
}
```

para compilarlo usamos *gcc* en línea de comandos mediante:

```
$ gcc hola.c -o hola
```

y lo ejecutamos con:

```
$ ./hola
```

Ejemplo en C++:

```
#include <iostream>
int main() {
   std::cout << "Hello World!\n";
}
```

para compilarlo usamos *g++* en línea de comandos mediante:

```
$ g++ hola.cpp -o hola
```

y lo ejecutamos con:

```
$ ./hola
```

[64] La extensión depende del lenguaje de programación, para el lenguaje C la extensión es .c, para el lenguaje C++ la extensión es .cpp, para el lenguaje Java la extensión es .java, para el lenguaje Python la extensión es .py.

Ejemplo en Java:

```
class hola {
   public static void main(String[] args) {
      System.out.println("Hello world!");
   }
}
```

para compilarlo usamos *javac* en línea de comandos mediante:

```
$ javac hola.java
```

y lo ejecutamos con:

```
$ java hola
```

Ejemplo en Python 2:

```
print "Hello World\n"
```

para compilarlo y ejecutarlo usamos *python2* en línea de comandos mediante:

```
$ python2 hola.py
```

Ejemplo en Python 3:

```
print ("Hello World\n")
```

para compilarlo y ejecutarlo usamos *python3* en línea de comandos mediante:

```
$ python3 hola.py
```

Un Ejemplo de Cálculo de Primos El algoritmo más simple, para determinar si un número es primo o compuesto, es hacer una serie de divisiones sucesivas del número, con todos los números primos menores que él, si alguna división da como residuo 0 o es divisible con el número entonces es compuesto en caso contrario es primo.

- Iniciamos con el número 4
- Verificamos si es divisible con los primos almacenados (iniciamos con 2 y 3)
- Si es divisible con algún número primo entonces es compuesto, en caso contrario es primo y este se guarda como un nuevo primo
- se incrementa uno al número y se regresa a verificar si es divisible entre los números primos almacenados, hasta encontrar por ejemplo los primeros mil primos

Ejemplo en C y C++:

```
#include <stdio.h>
// NPB Numero de primos a buscar
#define NPB 1000
int main ()
{
  int n, i, np;
  int p[NPB];
  // Guarda los primeros 2 primos
  p[0] = 2;
  p[1] = 3;
  np = 2;
  // Empieza la busqueda de primos a partir del numero 4
  n = 4;
  // Ciclo para buscar los primeros NPB primos
  while (np < NPB)
  {
    for (i = 0; i < np; i++)
    {
      if((n % p[i]) == 0) break;
```

```
    }
    if(i == np)
    {
      p[i] = n;
      np++;
    }
    n++;
  }
  // Visualiza los primos encontrados
  printf("\nVisualiza los primeros %d primos\n", NPB);
  for (i = 0; i < NPB; i++)
    printf("%d\n", p[i]);
  return 0;
}
```

Ejemplo en Java:

```
public class CalPrimos {
  public static void main(String[] args) {
    // NPB Numero de primos a buscar
    int NPB = 1000;
    int n, i, np;
    int p[] = new int[NPB];
    // Guarda los primeros 2 primos
    p[0] = 2;
    p[1] = 3;
    np = 2;
    // Empieza la busqueda de primos a partir del numero 4
    n = 4;
    // Ciclo para buscar los primeros NPB primos
    while (np < NPB) {
      for (i = 0; i < np; i++) {
        if((n % p[i]) == 0) break;
      }
      if (i == np) {
        p[i] = n;
        np++;
      }
```

```
            n++;
          }
          // Visualiza los primos encontrados
          System.out.println("Visualiza los primeros " + NPB + "
primos");
          for (i = 0; i < NPB; i++) System.out.println(p[i]);
        }
      }
```

Nótese que el algoritmo para buscar primos de los lenguajes C, C++ y el de Java son muy similares salvo la declaración de la función inicial y el arreglo que contendrá a los primos. Esto deja patente la cercanía entre dichos lenguajes y por que en este trabajo los presentamos en forma conjunta.

Ejemplo en Python:

```
def criba_Eratostenes(N):
  p = [] # inicializa el arreglo de primos encontrados
  # Guarda los primeros 2 primos
  p.append(2)
  p.append(3)
  np = 2
  # Empieza la busqueda de primos a partir del numero 4
  n = 4
  #Ciclo para buscar los primeros N primos
  while np < N:
    xi = 0
    for i in p:[65]
      xi = xi + 1
      if (n % i) == 0:
        break
    if xi == np:
      p.append(n)
      np = np + 1
    n = n + 1
   # Visualiza los primos encontrados
```

[65] En este código, al usar el *for* sobre los elementos del arreglo de primos, es necesario usar un contador para saber si ya se recorrieron todos los primos existentes y así determinar si es un nuevo primo y agregarlo a la lista.

```
    print("Visualiza los primeros " + str(N) + " primos ")
    for i in range(np):
      print(p[i])
    return p
# Solicita el calculo de los primeros primos
P = criba_Eratostenes(1000)
print(P)
```

Otro ejemplo en Python:

```
def criba_Eratostenes(N):
    p = [] # inicializa el arreglo de primos encontrados
    # Guarda los primeros 2 primos
    p.append(2)
    p.append(3)
    np = 2
    # Empieza la busqueda de primos a partir del numero 4
    n = 4
    #Ciclo para buscar los primeros N primos
    while np < N:
      for i in range(np):[66]
        if (n % p[i]) == 0:
          break
      if i == np-1:
        p.append(n)
        np = np + 1
      n = n + 1
     # Visualiza los primos encontrados
    print("Visualiza los primeros " + str(N) + " primos ")
    for i in range(np):
      print(p[i])
    return p
# Solicita el calculo de los primeros primos
P = criba_Eratostenes(1000)
print(P)
```

[66] Aquí, se hace el recorrido sobre el arreglo de primos usando la indexación sobre sus elementos, usando el número de elementos que se tiene mediante el uso del *for* con un *range*.

6.5 Introducción a los Paradigmas de Programación

La construcción de programas de cómputo -Software- puede involucrar elementos de gran complejidad, que en muchos casos no son tan evidentes como los que se pueden ver en otras Ciencias e Ingenierías. Un avión, una mina, un edificio, una red de ferrocarriles son ejemplos de sistemas complejos de otras Ingenierías, pero el programador construye sistemas cuya complejidad puede parecer que permanece oculta. El usuario siempre supone que en informática todo es muy fácil -"apretar un botón y ya esta"-.

Cuando se inicia uno en la programación, es común el uso de pequeños ejemplos, generalmente se programa usando una estructura secuencial -una instrucción sigue a la otra- en programas cortos[67], cuando los ejemplos crecen se empieza a usar programación estructurada[68] -que usa funciones- para después proseguir con ejemplos más complejos haciendo uso de la programación orientada a objetos -uso de clases y objetos- o formulaciones híbridas de las anteriores[69].

A continuación delinearemos estos paradigmas de programación:

Programación Secuencial es un paradigma de programación en la que una instrucción del código sigue a otra en secuencia también conocido como código espagueti. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del programa.

[67] Los ejemplos no son complejos, suelen estar construidos y mantenidos por una o pocas personas, son códigos de cientos o miles de líneas y tienen un ciclo de vida corto. Además, se puede construir aplicaciones alternativas en un período razonable de tiempo y no se necesitan grandes esfuerzos en anális y diseño.

[68] Al crecer la complejidad del Software a desarrollar, es muy difícil o imposible que un desarrollador o un grupo pequeño de ellos pueda comprender todas las sutilidades de su diseño, para paliar los problemas que conlleva el desarrollo de grandes y complejos sistemas informáticos surge la programación orientada a objetos.

[69] El surgimiento de la programación orientada objetos trata de lidiar con una gran cantidad de requisitos que compiten entre sí, incluso contradiciéndose, tienen desacoplamientos de impedancias entre usuarios del sistema y desarrolladores y es común la modificación de los requisitos con el paso del tiempo pues los usuarios y desarrolladores comienzan a compenetrarse mejor. Así, la programación orientada a objetos permite dirigir un equipo grande de desarrolladores, manejar una gran cantidad de código, usar estándares de desarrollo —al igual que en otras ingenierías— y verificar la fiabilidad de los estándares existentes en el mercado.

Programación Estructurada también llamada Procedimental, es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: secuencia, selección (if y switch) e iteración (bucles for y while); así mismo, se considera innecesario y contraproducente el uso de la instrucción de transferencia incondicional, que podría conducir a código espagueti, mucho más difícil de seguir y de mantener, y fuente de numerosos errores de programación.

Entre las ventajas de la programación estructurada sobre el modelo anterior -hoy llamado despectivamente código espagueti-, cabe citar las siguientes:

- Los programas son más fáciles de entender, pueden ser leídos de forma secuencial y no hay necesidad de tener que rastrear saltos de líneas dentro de los bloques de código para intentar entender la lógica interna.
- La estructura de los programas es clara, puesto que las instrucciones están más ligadas o relacionadas entre sí.
- Se optimiza el esfuerzo en las fases de pruebas y depuración. El seguimiento de los fallos o errores del programa (debugging), y con él su detección y corrección se facilita enormemente.
- Se reducen los costos de mantenimiento. Análogamente a la depuración, durante la fase de mantenimiento, modificar o extender los programas resulta más fácil.
- Los programas son más sencillos y más rápidos de confeccionar.
- Se incrementa el rendimiento de los programadores.

Programación Orientada a Objetos (POO, u OOP según sus siglas en inglés) es un paradigma de programación que viene a innovar la forma de obtener resultados. El surgimiento de la programación orientada objetos trata de lidiar con una gran cantidad de requisitos que compiten entre sí, incluso contradiciéndose, tienen desacoplamientos de impedancias entre usuarios del sistema y desarrolladores y es común la modificación de los requisitos con el paso del tiempo pues los usuarios y desarrolladores comienzan a compenetrarse mejor. Así, la programación orientada a objetos permite

dirigir un equipo grande de desarrolladores, manejar una gran cantidad de código, usar estándares de desarrollo -al igual que en otras ingenierías- y verificar la fiabilidad de los estándares existentes en el mercado.

Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial. Los objetos son entidades que tienen un determinado «estado», «comportamiento (método)» e «identidad»:

- La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador -concepto análogo al de identificador de una variable o una constante-.

- Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

- Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una «programación estructurada camuflada» en un lenguaje de POO.

La programación orientada a objetos difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada solo se escriben funciones que procesan

datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

Conceptos fundamentales La programación orientada a objetos es una forma de programar que trata de encontrar solución a los problemas que genera el desarrollo de proyectos de tamaño mediano o grande y/o complejos. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- Clase: se puede definir de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella.

- Herencia: Por ejemplo, la herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables registrados como «públicos (public)» en C. Los componentes registrados como «privados (private)» también se heredan pero se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Para poder acceder a un atributo u operación de una clase en cualquiera de sus subclases pero mantenerla oculta para otras clases es necesario registrar los componentes como «protegidos (protected)», de esta manera serán visibles en C y en D pero no en otras clases.

- Objeto: La instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

- Método: Es un algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un «mensaje». Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un «evento» con un nuevo mensaje para otro objeto del sistema.

- Evento: Es un suceso en el sistema -tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto-. El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

- Atributos: Características que tiene la clase.

- Mensaje: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- Propiedad o atributo: Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

- Estado interno: Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

- Componentes de un objeto: Atributos, identidad, relaciones y métodos.

- Identificación de un objeto: Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

Características de la POO Existe un acuerdo acerca de qué características contempla la «orientación a objetos». Las características siguientes son las más importantes:

- Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un «agente» abstracto que puede realizar trabajo, informar y cambiar su estado, y «comunicarse» con otros objetos en el sistema sin revelar «cómo» se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos,

y, cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

- Encapsulamiento: Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión (diseño estructurado) de los componentes del sistema.

- Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en «tiempo de ejecución» esta última característica se llama asignación tardía o asignación dinámica.

- Herencia: Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple; siendo de alta complejidad técnica por lo cual suele recurrirse a la herencia virtual para evitar la duplicación de datos.

- Modularidad: Se denomina «modularidad» a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos),

cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las partes restantes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.

- Principio de ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una «interfaz» a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas.

- Recolección de basura: La recolección de basura (garbage collection) es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando.

Muchos de los objetos prediseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas. Está basada en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

Recursión o Recursividad Un concepto que siempre le cuesta bastante a los programadores que están empezando es el de recursión o recursividad (se puede decir de las dos maneras). La recursividad consiste en funciones que se llaman a sí mismas, evitando el uso de bucles y otros iteradores.

Uno ejemplo fácil de ver y que se usa a menudo es el cálculo del factorial de un número entero. El factorial de un número se define como ese número multiplicado por el anterior, éste por el anterior, y así sucesivamente hasta

llegar a 1. Así, por ejemplo, el factorial del número 5 sería: 5x4x3x2x1 = 120.

Tomando el factorial como base para un ejemplo, ¿cómo podemos crear una función que calcule el factorial de un número? Bueno, existen multitud de formas. La más obvia quizá sería simplemente usar un bucle determinado para hacerlo, algo así en C:

```
long factorial(int n){
   long res = 1;
   for(int i=n; i>=1; i- -) res = res * i;
   return res;
}
```

Sin embargo hay otra forma de hacerlo sin necesidad de usar ninguna estructura de bucle que es mediante recursividad. Esta versión de la función hace exactamente lo mismo, pero es más corta, más simple y más elegante:

```
long factorial(int n)
{
    long fact;
    if (n <= 1) return 1;
    return n*factorial(n-1);
}
```

Aquí lo que se hace es que la función se llama a sí misma (eso es recursividad), y deja de llamarse cuando se cumple la condición de parada (en este caso que el argumento sea menor o igual que 1 que es lo que hay en el condicional).

Ventajas e inconvenientes ¿Ganamos algo al utilizar recursión en lugar de bucles/iteradores para casos como este?

En este caso concreto del cálculo factorial no, y de hecho es una forma más lenta de hacerlo y ocupa más memoria. Esto no es preocupante en la realidad, pero conviene saberlo. Lo del factorial es solo una forma de explicarlo con un ejemplo sencillo y que sea fácil de entender.

Pero entonces, si no ganamos nada en este caso ¿para qué sirve realmente la recursividad?

Pues para resolver ciertos problemas de manera elegante y eficiente. El ejemplo típico sería el recorrer un árbol de elementos para hacer algo con

todos ellos. Imagínate un sistema de archivos con carpetas y subcarpetas y archivos dentro de estas carpetas, o el árbol de elementos de una página Web donde unos elementos incluyen a su vez otros y no sabes cuántos hay en cada uno. En este tipo de situaciones la manera más eficiente de hacer una función que recorra todos los elementos es mediante recursión. Es decir, se crea una función que recorra todos los elementos hijo del nodo que se le pase y que se llame a sí misma para hacer lo mismo con los subnodos que haya. En el caso del sistema de archivos se le pasaría una carpeta y se llamaría a sí misma por cada subcarpeta que hubiese, y así sucesivamente con todas las demás. Con una sola llamada inicial recorrerá automáticamente toda la estructura del sistema de archivos.

Con eso, y sin necesidad de complicarse, de repente se tiene una función muy poderosa capaz de enumerar cualquier estructura arbitraria por compleja que sea. Ahí es donde se ve el verdadero poder de la recursividad, aunque hay aplicaciones más potentes y más complejas todavía.

Detalles a Tener en Cuenta Otra cosa importante a tener en cuenta es que, cada vez que se hace una llamada a una función desde otra función (aunque sea a sí misma), se crea una nueva entrada en la pila de llamadas del intérprete. Esta tiene un espacio limitado por lo que puede llegar un punto en el que si se hacen demasiadas se sature y se produzca un error. A este error se le denomina "Desbordamiento de pila" o "Stack Overflow". Ahora ya sabemos de donde viene el nombre del famoso sitio para dudas de programadores sin el que la programación moderna no sería posible.

Además, hay que tener mucho cuidado con la condición de parada. Esta se refiere a la condición que se debe comprobar para determinar que ya no se harán más llamadas a la función. Es en ese momento en el que empiezan a devolverse los valores hacia "arriba", retornando a la llamada original.

Si no tienes la condición de parada controlada pueden pasar varias cosas (todas malas), como por ejemplo:

- Que se sature la pila y se produzca un desbordamiento
- Que se ocupe cada vez más memoria
- Que se produzcan desbordamientos de variables al ir acumulando resultados.

6.6 Errores de Redondeo y de Aritmética

La aritmética que realiza una computadora es distinta de la aritmética de nuestros cursos de álgebra o cálculo. En nuestro mundo matemático tradicional consideramos la existencia de números con una cantidad infinita de cifras, en la computadora cada número representable tienen sólo un número finito, fijo de cifras (véase 6.4), los cuales en la mayoría de los casos es satisfactoria y se aprueba sin más, aunque a veces esta discrepancia puede generar problemas.

Un ejemplo de este hecho lo tenemos en el cálculo de raíces de:

$$ax^2 + bx + c = 0$$

cuando $a \neq 0$, donde las raíces se calculan comúnmente con el algoritmo:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ y } x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

o de forma alternativa con el algoritmo que se obtiene mediante la racionalización del numerador:

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}} \text{ y } x_2 = \frac{-2c}{b - \sqrt{b^2 - 4ac}}$$

Otro algoritmo que implementaremos es el método de Newton-Raphson[70] en su forma iterativa:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

en el cual se usa x_0 como una primera aproximación a la raíz buscada y x_n es la aproximación a la raíz después de n iteraciones (sí se converge a ella), donde $f(x) = ax^2 + bx + c$ y $f'(x_i) = 2ax + b$.

Salida del Cálculo de Raíces La implementación computacional se muestra en los programas desarrollados en distintos lenguajes (C, C++, Java y Python) usando diferentes paradigmas de programación (secuencial, procedimental y orientada a objetos), todos ellos generan la siguiente salida:

[70] También podemos usar otros métodos, como el de Newton Raphson Modificado para acelerar la convergencia

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

que involucra la función $f(x)$, la primera derivada $f'(x)$ y a la segunda derivada $f''(x)$.

Polinomio (1.000000) X^2 + (4.000000)X + (1.000000) = 0

Chicharronera 1
Raiz (-0.2679491924311228), evaluacion raiz: -4.4408920985006262e-16
Raiz (-3.7320508075688772), evaluacion raiz: 0.0000000000000000e+00

Chicharronera 2
Raiz (-0.2679491924311227), evaluacion raiz: 0.0000000000000000e+00
Raiz (-3.7320508075688759), evaluacion raiz: -5.3290705182007514e-15

Metodo Newton-Raphson
Valor inicial aproximado de X1 = -1.2679491924311228
Raiz (-0.2679491924311227), evaluacion raiz: 0.0000000000000000e+00
Valor inicial aproximado de X2 = -4.7320508075688759
Raiz (-3.7320508075688772), evaluacion raiz: 0.0000000000000000e+00

En esta salida se muestra la raíz calculada y su evaluación en la ecuación cuadrática, la cual debería de ser cero al ser una raíz, pero esto no ocurre en general por los errores de redondeo. Además, nótese el impacto de seleccionar el algoritmo numérico adecuado a los objetivos que persigamos en la solución del problema planteado.

En cuanto a la implementación computacional, el paradigma de programación seleccionado depende la complejidad del algoritmo a implementar y si necesitamos reusar el código generado o no.

Si lo que necesitamos implementar computacionalmente es una fórmula o conjunto de ellas que generen un código de decenas de líneas, la implementación secuencial es suficiente, si es menor a una centena de líneas puede ser mejor opción la implementación procedimental y si el proyecto es grande o complejo, seguramente se optará por la programación orientada a objetos o formulaciones híbridas de las anteriores.

En última instancia, lo que se persigue en la programación es generar un código: correcto, claro, eficiente, de fácil uso y mantenimiento, que sea flexible, reusable y en su caso portable.

Ejemplo en C Usando Programación Estructurada: Escribimos el código siguiente en cualquier editor y lo llamamos *cuadratica.c:*

```
#include <stdio.h>
#include <math.h>

// Funcion cuadratica
double f(double x, double a, double b, double c) {
  return (x * x * a + x * b + c);
}

// Derivada de la funcion cuadratica
double df(double x, double a, double b) {
  return (2.0 * x * a + b);
}

// Evalua el valor X en la función cuadratica
void evalua(double x, double a, double b, double c) {
  printf("\nRaiz (%1.16f), evaluacion raiz: %1.16e", x, (a * x * x
+ b * x + c));
}

// Metodo Newton-Raphson
// x = x - f(x)/f'(x)
double metodoNewtonRapson(double x, int ni, double a, double
b, double c) {
   int i;
  for (i = 0; i < ni; i++) {
    x = x - (f(x, a, b, c) / df(x, a, b));
  }
  return x;
}

// Funcion Principal ....
int main() {
   // Coeficientes del polinomio
   double A = 1.0, B = 4.0, C = 1.0;
   // Raices del polinomio
```

```
        double X1, X2, x;
        // Calculo del discriminante
        double d = B * B - 4.0 * A * C;
        // Raices reales
        if (d >= 0.0) {
          printf("\nPolinomio (%f) X^2 + (%f )X + (%f) = 0\n", A,
B, C);
          printf("\nChicharronera 1");
          X1 = (-B + sqrt(d)) / (2.0 * A);
          X2 = (-B - sqrt(d)) / (2.0 * A);
          evalua(X1, A, B, C);
          evalua(X2, A, B, C);
          printf("\n\nChicharronera 2");
          X1 = (-2.0 * C) / (B + sqrt(d));
          X2 = (-2.0 * C) / (B - sqrt(d));;
          evalua(X1, A, B, C);
          evalua(X2, A, B, C);
          // Metodo Newton-Raphson
          printf("\n\nMetodo Newton-Raphson");
          x = X1 - 1.0;
          printf("\nValor inicial aproximado de X1 = %f", x);
          x = metodoNewtonRapson(x, 6, A, B, C);
          evalua(x, A, B, C);
          x = X2 - 1.0;
          printf("\nValor inicial aproximado de X2 = %f", x);
          x = metodoNewtonRapson(x, 6, A, B, C);
          evalua(x, A, B, C);
          printf("\n");
        } else {
          // Raices complejas
         printf("Raices Complejas ...");
        }
        return 0;
    }
```

Ejemplo en C++ Usando Programación Orientada a Objetos Escribimos el código siguiente en cualquier editor y lo llamamos *cuadratica.cpp:*

```
#include <stdio.h>
#include <math.h>
#include<iostream>
using namespace std;
class cuadratica {
 private:
    // Coeficientes del polinomio
    double A, B, C;

  // Funcion cuadratica
  double f(double x, double a, double b, double c) {
    return (x * x * a + x * b + c);
  }

  // Derivada de la funcion cuadratica
  double df(double x, double a, double b) {
    return (2.0 * x * a + b);
  }

  // Evalua el valor X en la función cuadratica
  void evalua(double x, double a, double b, double c) {
    printf("\nRaiz (%1.16f), evaluacion raiz: %1.16e", x, (a * x
* x + b * x + c));
  }

  // Metodo Newton-Raphson
  // x = x - f(x)/f'(x)
  double metodoNewtonRapson(double x, int ni, double a, double
b, double c) {
    int i;
    for (i = 0; i < ni; i++) {
      x = x - (f(x, a, b, c) / df(x, a, b));
    }
    return x;
  }
```

```
public:
  // Constructor de la clase
  cuadratica(double a, double b, double c) {
    A = a;
    B = b;
    C = c;
  }
  // calculo de raices
  void raices() {
    // Raices del polinomio
    double X1, X2, x;
    // Calculo del discriminante
    double d = B * B - 4.0 * A * C;

    // Raices reales
    if (d >= 0.0) {
      printf("\nPolinomio (%f) X^2 + (%f )X + (%f) = 0\n",
A, B, C);
      printf("\nChicharronera 1");
      X1 = (-B + sqrt(d)) / (2.0 * A);
      X2 = (-B - sqrt(d)) / (2.0 * A);
      evalua(X1, A, B, C);
      evalua(X2, A, B, C);

      printf("\n\nChicharronera 2");
      X1 = (-2.0 * C) / (B + sqrt(d));
      X2 = (-2.0 * C) / (B - sqrt(d));;
      evalua(X1, A, B, C);
      evalua(X2, A, B, C);
      // Metodo Newton-Raphson
      printf("\n\nMetodo Newton-Raphson");
      x = X1 - 1.0;
      printf("\nValor inicial aproximado de X1 = %f", x);
      x = metodoNewtonRapson(x, 6, A, B, C);
      evalua(x, A, B, C);
      x = X2 - 1.0;
      printf("\nValor inicial aproximado de X2 = %f", x);
```

```
            x = metodoNewtonRapson(x, 6, A, B, C);
            evalua(x, A, B, C);
            printf("\n");
          } else {
            // Raices complejas
            printf("Raices Complejas ...");
          }
        }
};

int main(void)
{
    cuadratica cu1 = cuadratica(1.0, 4.0, 1.0);
    cu1.raices();
    return 0;
}
```

Ejemplo en Java Usando Programación Orientada a Objetos Escribimos el código siguiente en cualquier editor y lo llamamos *cuadratica.java:*

```
import java.lang.Math;
public class cuadratica {
    // Coeficientes del polinomio
    double A, B, C, d;

    // Constructor de la clase
    public cuadratica(double a, double b, double c) {
      A = a;
      B = b;
      C = c;
    }

    // Funcion cuadratica
    public double f(double x) {
      return (x * x * A + x * B + C);
    }
```

```
        // Evalua el valor X en la función cuadratica
        public void evalua(double x) {
          System.out.println("Raiz (" + x + ") , evaluacion raiz:" +
(A * x * x + B * x + C));
        }

        // Derivada de la funcion cuadratica
        public double df(double x) {
          return (2.0 * x * A + B);
        }

        // Metodo Newton-Raphson
        // x = x - f(x)/f'(x)
        public double metodoNewtonRapson(double x, int ni) {
          for (int i = 0; i < ni; i++) {
            x = x - (f(x) / df(x));
          }
          return x;
        }

        public void raices() {
          // Raices del polinomio
          double X1, X2, x;
          // Calculo del discriminante
          d = B * B - 4.0 * A * C;
          // Raices reales
          if (d >= 0.0) {
            System.out.println("");
            System.out.println("Raices Reales (" + A + ")X^2 + ("
+ B + ")X + (" + C + ") = 0");
            System.out.println("Chicharronera 1");
            X1 = (-B + Math.sqrt(d)) / (2.0 * A);
            X2 = (-B - Math.sqrt(d)) / (2.0 * A);
            evalua(X1);
            evalua(X2);
            System.out.println("");
            System.out.println("Chicharronera 2");
            X1 = (-2.0 * C) / (B + Math.sqrt(d));
```

```
                X2 = (-2.0 * C) / (B - Math.sqrt(d));;
                evalua(X1);
                evalua(X2);
                System.out.println("");
                  // Metodo Newton-Raphson
                System.out.println("Newton-Rapson");
                X1 = Math.round(X1);
                System.out.print("Valor inicial aproximado de X1 = " +
X1);
                x = metodoNewtonRapson(X1, 8);
                evalua(x);
                X2 = Math.round(X2);
                System.out.print("Valor inicial aproximado de X2 = " +
X2);
                x = metodoNewtonRapson(X2, 8);
                evalua(x);
              } else {
                // Raices complejas
                System.out.println("Raices Complejas ...");
              }
            }

            // Funcion Principal ....
              public static void main(String[] args) {
              cuadratica cu1 = new cuadratica(1.0, 4.0, 1.0);
              cu1.raices();
            }
        }
```

Ejemplo en Python Usando Programación Orientada a Objetos
Escribimos el código siguiente en cualquier editor y lo llamamos *cuadratica.py:*

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import math
class Ejemplo:
  # Constuctor
  def __init__(self, a, b ,c):
    self.A = a;
    self.B = b;
    self.C = c;
  # Funcion cuadratica
  def f(self, x, a, b, c):
    return (x * x * a + x * b + c);
  # Derivada de la funcion cuadratica
  def df(self, x, a, b):
    return (2.0 * x * a + b);
  # Evalua el valor X en la funcion cuadratica
  def evalua(self, x, a, b, c):
    print('Raiz (%1.16f), evaluacion raiz: %1.16e' % (x, (a * x *
x + b * x + c)))
  # Metodo Newton-Raphson
  # x = x - f(x)/f'(x)
  def metodo_Newton_Rapson(self, x, ni, a, b, c):

    for i in range(ni):
      x = x - (self.f(x, a, b, c) / self.df(x, a, b))
    return x
  # Calculo de raices
  def raices(self):
    # Calculo del discriminante
    d = self.B * self.B - 4.0 * self.A * self.C

    # Raices reales
    if d >= 0.0:
```

```
            print('\nPolinomio (%f) X^2 + (%f )X + (%f) = 0\n' %
(self.A, self.B, self.C))
            print('\nChicharronera 1')
            X1 = (-self.B + math.sqrt(d)) / (2.0 * self.A)
            X2 = (-self.B - math.sqrt(d)) / (2.0 * self.A)
            self.evalua(X1, self.A, self.B, self.C)
            self.evalua(X2, self.A, self.B, self.C)
            print('\nChicharronera 2')
            X1 = (-2.0 * self.C) / (self.B + math.sqrt(d))
            X2 = (-2.0 * self.C) / (self.B - math.sqrt(d))
            self.evalua(X1, self.A, self.B, self.C)
            self.evalua(X2, self.A, self.B, self.C)
            # Metodo Newton-Raphson
            print("\n\nMetodo Newton-Raphson")
            x = X1 - 1.0;
            print("\nValor inicial aproximado de X1 = %1.16f" % x)
            x = self.metodo_Newton_Rapson(x, 6, self.A, self.B, self.C)
            self.evalua(x, self.A, self.B, self.C);
            x = X2 - 1.0;
            print("\nValor inicial aproximado de X2 = %1.16f" % x);
            x = self.metodo_Newton_Rapson(x, 6, self.A, self.B, self.C)
            self.evalua(x, self.A, self.B, self.C)
            print("\n")
        else:
            # Raices complejas
            print("Raices Complejas ...")

if __name__ == '__main__':
    ej = Ejemplo(1.0, 4.0, 1.0)
    ej.raices()
```

6.7 Documentación del Código Fuente

La documentación dentro del código fuente (véase 8.7.1) tiene como objetivo que los que lo lean, mantengan y reparen el código, lo entiendan. Así que la documentación debe tener los siguientes propósitos:

- Explicar el propósito de cada clase y comportamiento, aunque pueda parecer obvia para tí, puede no ser tan obvia para otras personas.
- Describir los parámetros que espera cada comportamiento, los valores de retorno, y las excepciones que puede lanzar.
- Si la clase o los métodos están fuertemente acoplados con otra clase o forma de llamado, es necesario mencionarlo.
- Los comentarios deben indicar: El **por qué** (razones) y **cómo funciona el código**.
- Las cadenas de documentación deben indicar: **Cómo usar** el código.

Tener cadenas de documentación o comentarios incorrectos es mucho peor que no tenerlos en absoluto. Por ello es nuestro menester mantenerlos actualizados cando se hagan cambios, asegurándonos de que los comentarios y las cadenas de documentación continúan siendo consistentes con el código, y no lo contradicen. Algunas reglas son:

1. Los comentarios no deben duplicar el código.
2. Los buenos comentarios no excusan el código poco claro.
3. Si no puede escribir un comentario claro, es posible que haya un problema con el código.
4. Los comentarios deben disipar la confusión, no causarla.
5. Explique el código unidiomático en los comentarios.
6. Proporcione enlaces a la fuente original del código copiado.
7. Incluya enlaces a referencias externas donde sean más útiles.
8. Agregue comentarios al corregir errores.

9. Utilice comentarios para marcar implementaciones incompletas.

La documentación básica en C, C++ y Java se realiza usando:

- // para comentar dentro de una línea de código:

```
for (int i = 0; i < 10; i ++) // Este ciclo se realiza 10 veces
```

- /* y */ para comentar una o más líneas:

```
/*
  Este ciclo se realiza 10 veces
*/
for (int i = 0; i < 10; i ++) xp += 10 + i;
```

La documentación básica en Python se realiza usando:

- # para comentar dentro de una línea de código:

```
for i in Array: # Este ciclo se realiza tantas veces como elementos en Array
```

- """ y """ para comentar una o más líneas:

```
"""
  Este ciclo se realiza tantas veces como elementos en Array
"""
for i in Array:
  xp = xp + i
```

Además, si se realiza la documentación con cierta estructura, esta se puede utilizar para generar un manual de referencia del código en formatos: *HTML, PDF, PS, o XML* a partir de los fuentes con unos cuantos comandos de texto en unos segundos, pues qué mejor.

Existen varias herramientas para ello, una de ellas es DOXYGEN para códigos de Java, Fortran, C y C++, en Python se puede usar *Docstring* o una cadena de documentación como se verá en la siguientes secciones.

Instalación de Doxygen Para instalar DOXYGEN usar:

```
# apt install doxygen graphviz
```

una vez instalada, hay que generar el archivo de configuración de DOXYGEN, para ello usar:

```
$ doxygen -g
```

de aquí podemos editar el archivo Doxyfile generado según las necesidades de la documentación, un ejemplo de dicha configuración para generar la salida en *HTML, LaTeX y XML* esta en:
Jerarquia de Clases DOXYGEN
Para generar la documentación de los fuentes en la carpeta donde este el archivo de configuración y los archivos fuentes, usar:

```
$ doxygen
```

La documentación generada con DOXYGEN se mostrará en carpetas separadas para cada una de las salidas seleccionadas por ejemplo: *HTML, LaTeX, XML*, etc.
Para ver la documentación generada, usar en la consola:

```
$ cd html
$ xpdf index.html
```

Para generar la documentación en formato *PDF* a partir de la salida de *LaTeX* usar:

```
$ cd latex
$ make pdf
$ xpdf refman.pdf
```

en este caso se supone que se tiene instalado *LaTeX* en la máquina, en caso contrario podemos instalar lo básico usando:

```
# apt install science-typesetting texlive-science
```

y adicionalmente, si se requieren otras opciones instalamos:

```
# apt install texmaker texmacs texmacs-extra-fonts texlive-
  latex-base texlive-latex-recommended myspell-en-us myspell-es
```

6.7.1 Documentar en C, C++ y Java

Hay varios estilos de documentación (véase 8.7.1), aquí ponemos una que es fácil de usar para códigos en C++, pero es lógicamente extensible a lenguajes como Java.

```
#ifndef __test__
#define __test__
/// Descripcion breve de la clase.
/**
 * Descripcion detallada de la clase ...
 *
 * @author Antonio Carrillo
 * @date Winter 2010
 * @version 0.0.1
 * @bug No errors detected
 * @warning No warnings detected
 * @todo Exception handling
 */
class test
{
 private:

 /// Descripcion breve.
 const char *nmClass;
 /**
 * Descripcion corta.
 *
 * Descripcion larga ...
 *
 * 0 = Dirichlet, 1 = Neumann (or Robin)
 */
 int bdType;



 public:
 /**
```

```
* Descipcion breve.
*
* Descripcion detallada ...
*
* Algo de LaTeX ...
*
* \f[
* |I_2|=\left| \int_{0}^T \psi(t)
* \left\{
* u(a,t)-
* \int_{\gamma(t)}^a
* \frac{d\theta}{k(\theta,t)}
* \int_{a}^\theta c(\xi)u_t(\xi,t)\,d\xi
* \right\} dt
* \right|
* \f]
*
*
* @param[out] clas Descripcion del parametro de salida
* @param[in] fun Descripcion del parametro de entrada
*/
test(const char *clas, const char *fun)
{
nameClassFunct(clas, fun);
}

/**
* Descripcion breve.
*
* Descripcion detallada
*
* @param nVert Descripcion del parametro
* @param[in] g Descripcion del parametro
* @param[in] me Descripcion del parametro
* @param[out] values Descripcion del parametro
* @param z Descripcion del parametro
* @return Descripcion de lo que regresa
*/
```

```
    int eval(int nVert, double **g, StdElem *me, double ***values,
double *z);
    };
    /**
     * Descripcion breve de la clase.
     *
     * Descripcion detallada de la clase
     *
     * Otro parrafo de la descripcion ...
     *
     * Algo de formulas con LaTeX
     *
     * \f{eqnarray*}{
     * g &=& \frac{Gm_2}{r^2} \\
     * &=& \frac{(6.673 \times 10^{-11}\,\mbox{m}^3\,\mbox{kg}^{-
1}\,
     * \mbox{s}^{-2})(5.9736 \times 10^{24}\,\mbox{kg})}{(6371.01\,\mbox{km})^2}
\\
     * &=& 9.82066032\,\mbox{m/s}^2
     * \f}
     *
     *
     * Documentacion sobre la cual se basa la clase o archivo(s) que
hagan una descripcion de la
     * misma: Archivo.doc
     *
     * Descripcion breve del ejemplo de uso de esta clase (este archivo
se supone que estara en
     * una carpeta de nombre ./Examples en la posicion actual del
código)
     *
     * Algo de LaTeX
     *
     * La distancia entre \f$(x_1,y_1)\f$ and \f$(x_2,y_2)\f$ is
\f$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}\f$.
     *
     * @example ExampleText.cpp
     */
```

```
#endif
```

Adicionalmente es deseable que algunos comportamientos o clases tengan información adicional como son: propósito, entradas, salidas, estructuras de datos usadas en entradas y salidas, dependencia de datos o ejecución, restricciones, etc., usando una estructura como la siguiente:

```
/**
* Proposito y Metodo:
* Entradas:
* Salidas:
* Entradas TDS:
* Salidas TDS:
* Dependencias:
* Restricciones y advertencias"
*/
```

Para controlar las versiones se podría usar algo como lo siguiente:

```
/**
* @file release.notes
* @brief Package TkrRecon
* @verbatim
* Coordinator: Leon Rochester
*
* v4r4p8 09-Mar-2002 LSR Remove GFxxxxx and SiRecObjs,
no longer used
* v4r4p7 07-Mar-2002 TU Mainly, add a combo vertexing to
the TkrRecon sequence
* @endverbatim
*/
```

Un ejemplo completo puede ser el siguiente:

```
#ifndef __ErrorControl__
#define __ErrorControl__
#include <new>
using namespace std;
#include <stdlib.h>
```

```
#include "Printf.hpp"
#ifdef USE_HYPRE
#include <mpi.h>
#endif
/// Error Control, this class handles errors for the system RESSIM
/**
* @author Antonio Carrillo and Gerardo Cisneros
* @date Winter 2010
* @version 0.0.2
* @verbatim
Coordinator: Robert Yates
v0.0.1 Junuary 2011 Antonio Carrillo generates the first version of the class
v0.0.2 March 2011 Gerardo Cisneros add HYPRE errors control
Inputs: Name of class and function
Outputs: Exit of program
TDS Inputs: none
TDS Outputs: none
Dependencies: #ifdef USE_HYPRE, MPI package
Restrictions and Caveats: Non exception handling still
@endverbatim
* @bug No errors detected
* @warning No warnings detected
* @todo Exception handling
*/
class ErrorControl {
private:
/// Name of class
const char *nmClass;
/// Name of function generating the error
const char *nmFunction;
public:
/**
* Class Constructor
*/
ErrorControl(void) {
```

```
    nameClassFunct(" ", " ");
    }
    /**
    * Class Constructor
    * @param clas Class name
    */
    ErrorControl(const char *clas) {
    nameClassFunct(clas, " ");
    }
    /**
    * Class Constructor
    * @param clas Class name
    * @param fun Name of function generating the error
    */
    ErrorControl(const char *clas, const char *fun) {
    nameClassFunct(clas, fun);
    }
    /**
    * Name of class and function
    * @param clas Class name
    * @param func Name of function generating the error
    */
    void nameClassFunct(const char * clas, const char *func) {
    nameClass(clas);
    nameFunct(func);
    }
    /**
    * No memory for this request
    * @param var Var name
    */
    void memoryError(const char * var) {
    Afprintf(stderr, "\n\nNo memory for %s request in %s of class
  %s\n\n", var, nmFunction, nmClass);
    fatalError(1);
    }
    /**
    * No memory for this request
    * @param var Var name
```

```
    * @param i Index number
    */
    void memoryError(const char * var, int i) {
    Afprintf(stderr, "\n\nNo memory for %s request %d in %s of
  class %s\n\n", var, i, nmFunction, nmClass);
    fatalError(1);
    }
    /**
    * No memory for this request
    * @param var Var name
    * @param func Name of function generating the error
    */
    void memoryError(const char * var, const char *func) {
    Afprintf(stderr, "\n\nNo memory for %s request in %s of class
  %s\n\n", var, func, nmClass);
    fatalError(1);
    }
    /**
    * Fatal error.
    * @param cod Error code
    */
    void fatalError(int cod) {
    Afprintf(stderr, "\nFatal Error\nEnd program\n");
    #ifdef USE_HYPRE
    MPI_Abort(MPI_COMM_WORLD, cod);
    #else
    exit(cod);
    #endif
    }
    /**
    * Fatal error.
    * @param cod Error code
    */
    void fatalError(int cod, const char *txt) {
    Afprintf(stderr, txt);
    Afprintf(stderr, "\nFatal Error\nEnd program\n");
    #ifdef USE_HYPRE
    MPI_Abort(MPI_COMM_WORLD, cod);
```

```
#else
exit(cod);
#endif
}
/**
* Set name of class
* @param clas Class name
*/
void nameClass(const char *clas) {
nmClass = clas;
}
/**
* Set name of function
* @param func Function name
*/
void nameFunct(const char *func) {
nmFunction = func;
}
};
/**
* Error Control, this class handles errors for the system RESSIM
*
* Use of the class ErrorControl for error handling within the
system RESSIM,
* for example in the error control of memory request
*
* @example ExampleErrorControl.cpp
*/
#endif
```

Más detalles sobre los parámetros en la documentación del código fuente para ser usada por DOXYGEN se pueden ver en:

http://www.stack.nl/~dimitri/doxygen/commands.html#cmdparam

6.7.2 Documentar en Python

En Python el uso de acentos y caracteres extendidos esta soportado por la codificación UTF-8 (véase 8.7.1), para ello en las primeras líneas de código es necesario usar:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

siempre y cuando se use un editor o IDE que soporte dicha codificación, en caso contrario los caracteres se perderán.

Para documentar en Python se usa un *Docstring* o cadena de documentación, esta es una cadena de caracteres que se coloca como primer enunciado de un módulo, clase, método o función, y cuyo propósito es explicar su intención. Un ejemplo sencillo (en Python 3), es:

```
def promedio(a, b):
  """Calcula el promedio de dos numeros."""
  return (a + b) / 2
```

Un ejemplo más completo:

```
def formula_cuadratica(a, b, c):
  """Resuelve una ecuación cuadratica.
  Devuelve en una tupla las dos raices que resuelven la
  ecuacion cuadratica:

  ax^2 + bx + c = 0.
  Utiliza la formula general (tambien conocida
  coloquialmente como el "chicharronero").
  Parametros:
  a – coeficiente cuadratico (debe ser distinto de 0)
  b – coeficiente lineal
  c – termino independiente
  Excepciones:
  ValueError – Si (a == 0)
  """
if a == 0:
  raise ValueError(
    'Coeficiente cuadratico no debe ser 0.')
from cmath import sqrt
discriminante = b ** 2 - 4 * a * c
x1 = (-b + sqrt(discriminante)) / (2 * a)
x2 = (-b - sqrt(discriminante)) / (2 * a)
return (x1, x2)
```

La cadena de documentación en el segundo ejemplo es una cadena multilíneas, la cual comienza y termina con triples comillas (""""). Aquí se puede observar el uso de las convenciones establecidas en el PEP 257 (Python Enhancement Proposals):

- La primera línea de la cadena de documentación debe ser una línea de resumen terminada con un punto. Debe ser una breve descripción de la función que indica los efectos de esta como comando. La línea de resumen puede ser utilizada por herramientas automáticas de indexación; es importante que quepa en una sola línea y que este separada del resto del *docstring* por una línea en blanco.

- El resto de la cadena de documentación debe describir el comportamiento de la función, los valores que devuelve, las excepciones que arroja y cualquier otro detalle que consideremos relevante.

- Se recomienda dejar una línea en blanco antes de las triples comillas que cierran la cadena de documentación.

Todos los objetos documentables (módulos, clases, métodos y funciones) cuentan con un atributo *__doc__* el cual contiene su respectivo comentario de documentación. A partir de los ejemplos anteriores podemos inspeccionar la documentación de las funciones *promedio* y *formula_cuadratica* desde el *shell* de Python:

```
>>> promedio.__doc__
'Calcula el promedio de dos numeros.'
>>> formula_cuadratica.__doc__
'Resuelve una ecuación cuadratica.\n\n Devuelve en una
tupla las dos raices que resuelven la\n ecuacion
cuadratica:\n \n ax^2 + bx + c = 0.\n\n
Utiliza la formula general (tambien conocida\n
coloquialmente como el "chicharronero").\n\n
Parametros:\n a – coeficiente cuadratico (debe ser
distinto de 0)\n b – coeficiente lineal\n
c – termino independiente\n\n Excepciones:\n
ValueError – Si (a == 0)\n \n '
```

Sin embargo, si se esta usando el *shell* de Python es mejor usar la función *help()*, dado que la salida producida queda formateada de manera más clara y conveniente:

```
>>> help(promedio)
Help on function promedio in module __main__:
promedio(a, b)
Calcula el promedio de dos numeros.
>>> help(formula_cuadratica)
Help on function formula_cuadratica in module __main__:
formula_cuadratica(a, b, c)
Resuelve una ecuacion cuadratica.
Devuelve en una tupla las dos raices que resuelven la
ecuacion cuadratica:
ax^2 + bx + c = 0.
Utiliza la formula general (tambien conocida
coloquialmente como el "chicharronero").
Parametros:
a – coeficiente cuadratico (debe ser distinto de 0)
b – coeficiente lineal
c – termino independiente
Excepciones:
ValueError – Si (a == 0)
```

Ciertas herramientas, por ejemplo *shells* o editores de código, pueden ayudar a visualizar de manera automática la información contenida en los comentarios de documentación. De esta manera un usuario puede tener a su alcance de manera sencilla toda la información que necesita para poder usar nuestras funciones.

Generando documentación en páginas de HTML Los *Docstrings* se pueden usar también para producir documentación en páginas de HTML que pueden ser consultadas usando un navegador de Web. Para ello se usa el comando *pydoc* desde una terminal. Por ejemplo, si las dos funciones anteriores (*promedio* y *formula_cuadratica*) se encuentran en un archivo fuente llamado *ejemplos.py*, podemos ejecutar el siguiente comando en una terminal dentro del mismo directorio donde esta el archivo fuente:

```
pydoc -w ejemplos
```

La salida queda en el archivo *ejemplos.html*, y así se visualiza desde un navegador. La documentación de *pydoc* explica otros artilugios que puede hacer esta utilería de Python.

Docstrings vs. Comentarios Un comentario en Python inicia con el símbolo de número (#) y se extiende hasta el final de la línea. En principio los *Docstrings* pudieran parecer similares a los comentarios, pero hay una diferencia pragmática importante: los comentarios son ignorados por el ambiente de ejecución de Python y por herramientas como *pydoc*; esto no es así en el caso de los *Docstrings*.

A un nivel más fundamental hay otra diferencia aún más grande entre los *Docstrings* y los comentarios, y esta tiene que ver con la intención:

- Los *Docstrings* son documentación, y sirven para entender qué hace el código.
- Los comentarios sirven para explicar cómo lo hace.

La documentación es para la gente que usa tu código. Los comentarios son para la gente que necesita entender cómo funciona tu código, posiblemente para extenderlo o darle mantenimiento.

El uso de *Docstrings* en Python facilita la escritura de la documentación técnica de un programa. Escribir una buena documentación requiere de disciplina y tiempo, pero sus beneficios se cosechan cuando alguien -quizás mi futuro yo dentro de seis meses- necesita entender qué hacen nuestros programas. Los *Docstrings* no sustituyen otras buenas prácticas de programación, como son el uso apropiado de comentarios o el empleo de nombres descriptivos para variables y funciones.

6.8 Los Diagramas UML para Documentar

Todo el conocimiento está en el código fuente pero este no tiene una forma de fácil y rápida comprensión a alto nivel, para comprender un sistema es necesario una forma de documentación que muestre los detalles importantes de los que se compone el sistema. El lenguaje unificado de desarrollo o UML permite describir un sistema utilizando diferentes diagramas específicos para

mostrar diferentes aspectos del sistema. Hay muchas aplicaciones de Software libre disponible para diferentes sistemas operativos que permiten crear y exportar a imágenes los diferentes diagramas.

El trabajo de desarrollo de Software en mayor medida un trabajo colaborativo entre un grupo de personas, por ello la comunicación y transmisión de información es fundamental. Por otro lado, los sistemas informáticos de cierto tamaño son complejos de los cuales ninguna persona tiene el conocimiento completo y detallado de todas las partes que lo forman. Esto hace que la documentación y la transferencia de conocimiento sea importante en un grado similar al desarrollo de nuevas funcionalidades.

El problema de la documentación es que se quede obsoleta ni refleje el estado actual del sistema por los continuos cambios realizados en el código, para evitarlo la documentación ha de ser a alto nivel que describan los conceptos fundamentales del sistema que no serán propensos a cambiar. Otra forma de evitar esta obsolescencia en algunos casos es generar la documentación a partir de la información embebida en el código fuente como la documentación Javadoc de las clases y API de Java o la documentación de una API REST con Swagger. Sin embargo, no toda la documentación que aporta valor es posible generarla de forma automatizada.

El código fuente de una aplicación contiene el conocimiento más detallado y fiel de una aplicación , sin embargo, tratar de comprender un sistema complejo compuesto por unas cuantas decenas de miles de líneas de código aún usando un lenguaje de alto nivel ya sea Java , Python o C# es una tarea complicada o al menos que requiere mucho tiempo de investigación y esfuerzo de comprensión. Una breve descripción en prosa y un esquema del sistema permite obtener la misma información que analizando el código de forma mucho más rápida.

Si en una organización cuando una persona la abandona deja un vacío importante de conocimiento en las reglas de negocio implementadas o en la incorporación de una nueva persona a un equipo toma demasiado tiempo y dedicación adquirir el mismo conocimiento que el resto de miembros del equipo un problema de la organización quizá sea la falta de documentación. Con el paso del tiempo a medida que una aplicación es más compleja y dado que las personas en una organización se van con el conocimiento que posean y otras vienen pero necesitan adquirir conocimiento, la documentación es una forma de transferencia de conocimiento que está disponible de forma permanente y en forma de autoservicio.

Una buena documentación ha de ser concisa y transmitir su información

de forma rápida, una forma es utilizar diagramas que esquematizan el sujeto que se trata de describir complementados descripciones en texto. El lenguaje unificado de modelado o UML describe una notación estándar para los diagramas y varios tipos de diagramas.

El lenguaje unificado de modelado o UML estandariza las convenciones y elementos utilizados en los diagramas además de identificar y describir varios tipos de diagramas para describir un sistema desde varios puntos de vista. Cada uno de estos diagramas describen un sistema a alto nivel de forma esquematizada los conceptos fundamentales.

Los diagramas UML son de dos tipos estructurales que describen la parte estática del sistema y de comportamiento que describen la parte dinámica.

Diagramas UML estructurales:

- Diagrama de clases: en un lenguaje de programación orientado a objetos muestra las clases e interfaces por las que está compuesto el sistema y sus relaciones de herencia, composición e implementación con diferentes tipos de flechas. También muestra los atributos y métodos que poseen las clases.

- Diagrama de paquetes: los paquetes son agrupaciones lógicas de varias clases, en este diagrama se muestran los paquetes fundamentales asi como su relación y dependencia con otros.

- Diagrama de objetos: utilizan la misma notación que los diagramas de clases, se diferencian de los diagramas de clases en que muestran las instancias en un momento determinado del sistema.

- Diagrama de componentes: muestran los componentes fundamentales en los que se divide un sistema complejo y sus relaciones.

- Diagrama de estructura compuesta: muestra la estructura interna de una clase.

- Diagrama de despliegue: muestra en que elementos de computación se despliega el software asi como sus relaciones. Son especialmente útiles en los sistemas distribuidos o basados en microservicios.

Diagramas UML de comportamiento:

- Diagrama de flujo o actividad: muestra la secuencia de acciones y decisiones implementadas en un algoritmo o proceso.
- Diagrama de secuencia: muestran las interacciones de los objetos entre si y el orden en el que se producen esas interacciones. El orden de las acciones se muestran de forma vertical y las interacciones como flechas.
- Diagrama de caso de uso: proporciona una visión general de los actores involucrados en un sistema, las diferentes funciones que usa esos actores y cómo interactúan estas diferentes funciones. Proporciona una descripción global del sistema inicial.
- Diagrama de estado: algunos objetos o el sistema se comportan de forma diferente en función del estado, estos diagramas muestran los diferentes estados de los que se compone y cual es su comportamiento en función del estado y condiciones.
- Diagrama de comunicación
- Diagrama de interacción: muestra un proceso al igual que un diagrama de actividad pero empleando una colección de diagramas de secuencia con su misma notación.
- Diagrama de tiempos

Aplicación para crear diagramas de UML para crear los diagramas de UML es necesario utilizar una aplicación, una orientada y específica para crear diagramas UML. El Software libre dispone de muchas, algunas en la red y otras están disponibles como aplicación Flatpak como forma de empaquetar una aplicación para cualquier distribución GNU/Linux, para Windows y macOS.

Hay muchas herramientas de este tipo de modelado que son multiplataforma, pero no tantas si queremos que sean libres, sencillamente gratuitas y que se puedan usar en Linux. Puede que para muchos no sean las mejores, pero yo empezaría con dos que crean los desarrolladores de dos de los escritorios Linux más usados, o gente relacionada a ellos:

- Umbrello
- Gaphor

- PlantUML
- diagrams.net (https://www.diagrams.net)

7 Programación Orientada a Objetos

Comencemos este camino interminable de aprendizaje acerca de la programación orientada a objetos (POO) con los conceptos básicos de este tipo de programación -trataremos que sin pérdida de generalidad mostraremos los ejemplos en el lenguaje Java-, para ello iniciemos con lo que entendemos por un objeto.

¿Qué es un objeto? La definición más básica dice: "Es un ente computacional que puede contener datos y comportamientos".

> Ejemplo: un polinomio, una matriz o un número complejo. Cada uno de estos objetos tiene un comportamiento propio.
>
> Ejemplo: un polinomio puede sumar/integrar/derivar/evaluar.

Por lo tanto, un objeto lo podemos definir formalmente como: ente computacional que puede contener datos y exhibe comportamientos.

Mensaje Todos los objetos de los cuales hablamos tienen comportamientos y se relacionan con otros objetos (se piden cosas entre ellos). Por lo cual podemos definir a un mensaje como: Interacción entre dos objetos: un emisor E y un receptor R. Un emisor le envía un mensaje a un receptor. El emisor puede obtener o no una respuesta.

> Ejemplo: evaluar/grado/integrar.

Ciclo de vida de un objeto Todos los objetos tienen una vida, se puede decir que nacen cuando son instanciados y mueren cuando se elimina de la memoria. No obstante, existen dos formas de eliminar de memoria a un objeto dependiendo del lenguaje de programación. Los mecanismos son: Recolector de Basura (*Garbage Collector)* y *Destructores*.

Recolector de Basura: Es un mecanismo que se encarga de borrar de la memoria las referencias a objetos y entidades que ya no se usan más, de manera que se pueda maximizar el uso del espacio en memoria.

Destructores: Son métodos que se definen para cada objeto y cuyo principal objetivo es liberar los recursos que fueron adquiridos por el objeto a lo largo de su ciclo de vida y romper vínculos con otras entidades que puedan tener referencias a él.

Métodos ¿Que es un método? ¿En qué difiere de un mensaje?

Un método es la sección de código que se ejecuta al enviar un mensaje. Se identifica con una firma -el nombre y los parámetros del método-, que es la misma firma del mensaje enviado. Entonces, cuando un objeto recibe un mensaje, se ejecuta un método cuya firma es la misma que la del mensaje.

Método de la clase Polinomio:

```
// Evalua el polinomio en el valor pasado como parametro
public double evalua(double x) {
   int i;
   double p = 1.0, r = 0.0;
   for (i = 0; i < Dim; i++) {
      r += coeficiente(i) * p;
      p *= x;
   }
   return r;
}
```

La firma de un objeto se define con tres componentes:

1. El nombre del método.
2. Los parámetros que recibe el método.
3. Lo que devuelve el método (que puede ser nada u otro objeto).

Clases Es común que empezemos con la definición de "Clase" en los textos. Esto se debe a que las clases solo son una forma de implementar objetos, pero no son la única manera de hacerlo (como veremos más adelante), por lo que es fundamental que no pensemos automáticamente en clases cuando hablamos de objetos. Sin embargo, la mayor parte de los lenguajes de programación que usamos -laboral y académicamente- usan clases.

¿Qué es una clase? Podemos definir a una clase utilizando dos definiciones complementarias: una clase es un molde, a partir del cual se crean los objetos. Cuando instanciamos un objeto, el ambiente le pregunta a dicha clase que características y métodos debe tener el objeto. La otra definición es: una clase es un ente que determina el comportamiento y el tipo al que pertenecen sus instancias.

Clase Persona:

```
public class Persona {
   private String nombre;
   private String apellidos;
   private int edad;
   //Constructor
   public Persona(String nombre, String apellidos, int edad) {
      this.nombre = nombre;
      this.apellidos = apellidos;
      this.edad = edad;
   }
   public String getNombre () {
      return nombre;
   }
   public String getApellidos () {
      return apellidos;
   }
   public int getEdad () {
      return edad;
    }
   public void visualiza() {
      System.out.println ("Nombre: " + getNombre() + " " +
       getApellidos() + " edad: " + getEdad());
   }
   public static void main (String [ ] Args) {
      Persona p = new Persona ("Mauro", "Ruiz perez", 36);
      p.visualiza();
   }
}
```

Características de la Programación Orientada a Objetos Los siguientes conceptos que vamos a ver, son características que tiene un buen diseño[71] orientado a objetos, si bien este paradigma no garantiza dichas características, hacen mucho más fácil poder lograrlas y en cierta manera nos obliga a usarlas.

Abstracción La abstracción es la propiedad de los objetos que consiste en tener en cuenta sólo los aspectos más importantes desde un punto de vista determinado y no tener en cuenta otros aspectos. Durante el proceso de abstracción es cuando se decide qué características y comportamientos debe tener el modelo para así reducir su complejidad. De este modo, las características complejas se hacen más manejables.

> Ejemplo: En POO, podemos considerar una *Persona* como un objeto que tiene propiedades (como nombre, altura, peso, color

[71] El análisis orientado a objetos (Object-oriented Analysis OOA) es el proceso de analizar un problema, un sistema o una tarea —que alguien quiere convertir en una aplicación— e identificar los objetos y las interacciones entre esos objetos. La etapa de análisis tiene que ver con lo que se necesita hacer. La salida de la etapa de análisis es un conjunto de requisitos. Si tuviéramos que completar el análisis paso a paso, habríamos convertido una tarea, en un conjunto de requisitos. En cierto modo, el análisis es un nombre inapropiado. En su lugar, explora su entorno, manipula formas y ve dónde se podrían contener. Una mejor interpretación de la frase podría ser la exploración orientada a objetos. En desarrollo de Software, las etapas iniciales de análisis incluyen entrevistar a los clientes, estudiando sus procesos, y eliminando posibilidades.

El diseño orientado a objetos (Object-oriented Design OOD) es el proceso de convertir tales requisitos en una especificación de implementación. El diseñador debe nombrar los objetos, definir los comportamientos, y especificar formalmente qué objetos pueden activar comportamientos específicos en otros objetos. La etapa de diseño tiene que ver con cómo se deben hacer las cosas. La salida de la etapa de diseño es una especificación de implementación. Si fuéramos a completar la etapa de diseño en un solo paso, habríamos cumplido los requisitos definidos durante análisis orientado a objetos en un conjunto de clases e interfaces que podrían implementarse en (idealmente) cualquier lenguaje de programación orientado a objetos.

La programación orientada a objetos (Object-oriented Programming OOP) es el proceso de conversión del diseño en un programa de trabajo que haga exactamente lo que el solicitante del mismo originalmente solicitó. ¡Sí claro!, sería encantador si el mundo se encontrara con este ideal y pudiéramos seguir estas etapas una por una, en perfecto orden, como todos los antiguos libros de texto nos decían. Como de costumbre, el mundo real es mucho mas turbio, no importa cuánto intentemos separar estas etapas, siempre encontraremos cosas que necesitan más análisis mientras estamos diseñando. Cuando estamos programando, encontramos características que necesitan aclaración en el diseño.

de pelo, color de ojos, etcétera) y métodos (como hablar, mirar, andar, correr, parar, etcétera). Gracias a la abstracción, el objeto *Tren* puede manipular objetos del tipo *Persona* sin tener en cuenta sus propiedades ni métodos ya que sólo le interesa, por ejemplo, calcular la cantidad de personas que están viajando en él en ese momento, sin tener en cuenta ninguna otra información relacionada con dichas personas, tales como la altura, el nombre, el color de ojos, etcétera. Nuestro objeto *Tren* se abstrae del objeto del tipo *Persona*.

Ocultamiento de la información Este concepto hace referencia a que los componentes se deben utilizar como si sólo se conociera su interfaz y no se tuviera conocimiento de su implementación. En otras palabras, un objeto sabe que otro objeto entiende un determinado mensaje, el cual recibe ciertos parámetros y devuelve algo (o no).

Ejemplo: Yo como objeto *Médico* quiero saber cuantas cirugías tuvo un paciente, y entonces le envía un mensaje para que el *Paciente* me devuelva el número de cirugías que tuvo. El objeto *Médico* no sabe como devolvió ese número el *Paciente*, solo sabe que tiene que enviarle el mensaje y recibir un número.

Encapsulamiento Encapsulamiento es la capacidad de diferenciar qué partes de un objeto son parte de la interfaz y cuales permanecerán inaccesibles por el usuario. Son los lenguajes de programación los cuales, por medio de los modificadores de acceso, permiten indicar el modo de accesibilidad de un componente.

En POO, a la conjunción de abstracción y ocultamiento de implementación se le llama encapsulamiento.

Según Booch, encapsulamiento "... es el proceso de almacenar en un mismo comportamiento los elementos de una abstracción que constituyen su estructura y su comportamiento; sirve para separar la interfaz contractual de una abstracción y su implementación".

Veamos a través de un ejemplo:

```
public class TestPersona {
   public static void main (String [ ] Args) {
      Persona p = new Persona ("Mauro", "Ruiz perez", 36);
      System.out.println (p.edad);
   }
}
```

En este caso, podemos acceder al valor de la *edad* del objeto p de la clase *Persona*, pero lo recomendable es mandar el mensaje que solicite dicho valor mediante *getEdad*().

Viéndolo desde el punto de vista de la escalabilidad y de un lenguaje de programación, si el día de mañana tenemos diferentes objetos que tienen que modificar dicha propiedad, nuestro código se puede volver difícil de cambiar debido a que en varias partes se usa la propiedad *edad*, si tenemos encapsulado ese comportamiento, solo vamos a tener que hacer un cambio en un solo lugar.

Cohesión y Acoplamiento La cohesión mide la relación entre el concepto que deseamos modelar y las responsabilidades del componente que lo representan. El acoplamiento mide qué tan relacionados están los componentes del sistema entre sí y cómo esa dependencia provoca un diseño. Siempre buscaremos tener un bajo acoplamiento y que los objetos sean lo más cohesivos posible.

Claramente se puede ver que ambos conceptos están inversamente relacionados, nuestros diseños deben tener una alta cohesión y un bajo acoplamiento. El paradigma de objetos se basa en esta regla para generar diseños más sencillos, y al mismo tiempo fáciles de programar, probar y mantener.

Declaratividad y Expresividad Estos conceptos no son particulares de POO, pero aun así se nos olvidan, por eso les vamos a hacer un repaso.

La expresividad tiene que ver con qué tan claro es el código, qué tanto expresa lo que tenía en en mente el programador cuando lo escribió, es decir, que tan bien expresadas están las ideas del programador. Usar variables y métodos con nombres que representan lo que son o hacen, es una forma fácil de aplicar la expresividad, que no tiene nada que ver con el paradigma sino con el programador.

Por lo general la declaratividad ayuda a que el código sea más expresivo, porque no se mezcla el algoritmo con lo que se quiere que haga más a alto nivel. Cuando tenemos un código bien declarativo, podemos entender que

es lo que hace sin ver en detalle el algoritmo implementado. Es importante recalcar que la declaratividad es contraria a la imperatividad (o sea, detallar línea a línea los pasos que hace el algoritmo).

Entonces, la declaratividad indica qué se hace y la expresividad muestra la intención de lo que va a hacer el código.

Recordemos una frase de Martin Fowler que vale la pena comentar:

> "Cuando sientas la necesidad de escribir un comentario, intenta primero refactorizar el código de manera que cualquier comentario se convierte en innecesario."

La expresividad y la declaratividad son muy importantes para poder generar un código autodocumentado, ya que nunca sabemos en el futuro quienes van a tener que revisar nuestro código. Si un compañero que tiene que modificar nuestro código y no logra comprender fácilmente que quisimos hacer, tenemos un problema. Si nosotros volvemos a un código que escribimos y no lo entendemos, tenemos un grave problema.

7.1 Herencia y Composición

Sobrecarga de métodos La sobrecarga es la capacidad de un lenguaje de programación, que permite nombrar con el mismo identificador diferentes variables u operaciones.

La sobrecarga de métodos se refiere a la posibilidad de tener dos o más métodos con el mismo nombre pero diferentes parámetros. El compilador usará una u otra dependiendo de los parámetros usados.

El mismo método dentro de una clase permite hacer cosas distintas en función de los parámetros.

Clase Vector:

```
public class Vector {
    // Coeficientes del vector
    private double []C;
    // Guarda la dimension del arreglo de coeficientes
    private int Dim;
    // Constructor de la clase
    public Vector() {
       Dim = 0;
```

```
}
// Asigna coeficientes
public Vector(double []coef) {
   int i, n = coef.length;
   C = new double[n];
   Dim = n;
   for (i = 0; i < n; i++) C[i] = coef[i];
}
// Retorna los coeficientes
double coeficiente(int i) {
   if (i >= Dim) {
        System.out.print("Error");
        return 0;
  }
   return C[i];
}
// Retorna la dimension del arreglo de coeficientes
int dimension() {
   return Dim;
}
// Visualiza el vector
public void visualiza() {
   int i;
   System.out.print("(");
   for (i = 0; i < Dim; i++) {
      System.out.print(C[i]);
      if(i < Dim-1) System.out.print(", ");
   }
   System.out.print(")");
}
// Visualiza el vector
public void visualizaLN() {
   visualiza();
   System.out.println("");
}
// Suma con dos operandos
public void suma(Vector a, Vector b) {
   int i;
```

```
            Dim = 0;
            // Revisa el tamano de los vectores a sumar
            if(a.dimension() != b.dimension()) {
              System.out.println("Error, las dimensiones no son iguales");
            } else {
               // Solicita memoria para los coeficientes del vector
resultante
              Dim = a.dimension();
              C = new double[Dim];
              // Realiza la suma entre los coeficientes comunes
              for (i = 0; i < Dim; i++) C[i] = a.coeficiente(i) +
b.coeficiente(i);
            }
          }
          // Suma con un operando
          public void suma(Vector a) {
            int i;
            // Revisa el tamano de los vectores a sumar
            if(dimension() != a.dimension()) {
              System.out.println("Error, las dimensiones no son iguales");
            } else {
              // Realiza la suma entre los coeficientes comunes
              for (i = 0; i < Dim; i++) C[i] += a.coeficiente(i);
            }
          }
          // Funcion Principal ....
          public static void main(String[] args) {
            double a[] = {3.0, -4.0, 1.5};
            double b[] = {1.0, 1.0, 2.0};
            double c[] = {1.0, 3.0, -3.0};
            Vector A = new Vector(a);
            Vector B = new Vector(b);
            Vector C = new Vector(c);
            System.out.println("Suma con un operando:");
            C.visualizaLN();
            A.visualizaLN();
            C.suma(A);
            C.visualizaLN();
```

```
        System.out.println("Suma con dos operandos:");
        A.visualizaLN();
        B.visualizaLN();
        C.suma(A, B);
        C.visualizaLN();
    }
}
```

En este ejemplo que vemos, uno podría invocar al método "constructor" y "Suma" y según el número y tipo de parámetros o sin ellos de como se invoque en nuestro código, el compilador decidirá a cual llamar según corresponda.

En tiempo de compilación, se buscan todas las llamadas a este método y según el tipo de los parámetros con los que se esté invocando y el objeto que puede o no devolver (necesariamente son lenguajes de chequeo estático), se determina a qué implementación llamará. Si la combinación de parámetros no coincide, el compilador mandará mensaje de error.

Los multimétodos, a diferencia de la sobrecarga, son un conjunto de métodos con la misma firma -el nombre y los parámetros del método-, pero que se pueden solapar, y se decide cuál ejecutar en tiempo de ejecución.

Composición La composición se refiere a la combinación de objetos simples para hacer objetos más complejos.

Los objetos a menudo pueden dividirse en tipos compuestos y componentes, y la composición puede considerarse como una relación entre estos tipos: un objeto de un tipo compuesto (ej.: auto) "tiene un" objeto de un tipo simple (ej.: rueda).

Considere la relación de un automóvil con sus partes: el automóvil tiene o se compone de objetos como el volante, asientos, caja de cambios y el motor. Esta relación podría definirse como una relación de composición.

Los objetos compuestos generalmente se expresan por medio de referencias de un objeto a otro. Tales referencias pueden ser conocidas como atributos, campos, miembros o propiedades y la composición resultante como tipo compuesto. Sin embargo, tener esas referencias no necesariamente significa que un objeto es compuesto. Sólo se llama compuesto, si los objetos de los que se compone son realmente sus partes, es decir, no tienen existencia independiente.

Herencia Es el mecanismo por el cual un objeto se basa en otro objeto o clase, extendiendo la implementación para reutilizar su comportamiento[72].

Supongamos que la clase Médico y Profesor, por ser Personas, comparten datos personales, esto queda plasmado en la siguiente jerarquía de clases:

Clase Medico:

```
public class Medico extends Persona {
  private String especialidad;
  private String cedula;
  //Constructor de la subclase: incluimos como parametros
al menos los del constructor de la superclase
  public Medico(String nombre, String apellidos, int edad,
String especialidad, String cedula) {
    super(nombre, apellidos, edad);
    this.especialidad = especialidad;
    this.cedula = cedula;
  }
  public void setEspecialidad(String especialidad) {
    this.especialidad = especialidad;
  }
  public String getEspecialidad() {
    return especialidad;
  }
  public void setCedula(String cedula) {
    this.cedula = cedula;
  }
  public String getCedula() {
    return cedula;
  }
  public void visualiza() {
    System.out.println ("Nombre del medico: " + getNom-
bre() + " " + getApellidos() +
  " especialidad: " + getEspecialidad() + " cedula: " + getCedula());
  }
}
```

[72] En Java no se permite herencia múltiple, pero sí en Paython y C++, permitiendo diseños más complejos.

Clase Profesor:

```
public class Profesor extends Persona {
   private String idProfesor;
   //Constructor de la subclase: incluimos como parametros
al menos los del constructor de la superclase
   public Profesor(String nombre, String apellidos, int edad,
String idProfesor) {
      super(nombre, apellidos, edad);
      this.idProfesor = idProfesor;
   }
   public void setIdProfesor (String idProfesor) {
      this.idProfesor = idProfesor;
    }
   public String getIdProfesor () {
      return idProfesor;
   }
   public void visualiza() {
      System.out.println ("Nombre del profesor: " + getNom-
bre() + " " + getApellidos() +
  " Id de profesor: " + getIdProfesor() );
   }
}
```

¿Cómo funciona?

```
public class TestHerencia {
   public static void main (String [ ] Args) {
      Profesor p1 = new Profesor ("Juan", "Nadrie Garcia",
33, "Prof 22-387-11");
      Medico m1 = new Medico ("Roberto", "Gonzalez Gar-
cia", 33, "Cirujano", "328943784");
      Persona p = new Persona ("Mauro", "Ruiz perez", 36);
      p.visualiza();
      p1.visualiza();
      m1.visualiza();
   }
}
```

Cuando le llega un objeto, el compilador, hace lo siguiente:

1. Busca la implementación en la clase del objeto a la que le llega el mensaje
2. Si no la encontra, la busca en su padre
3. Repite el paso anterior hasta que no haya más padres.
4. Si no encuentra ninguna implementación en todos los niveles, lanza un error.

A este mecanismo se lo conoce como **Method LookUp**. Algo a destacar de la herencia es que las clases hijas van a heredar lo que haya definido su superclase, tanto atributos como métodos.

Los motivaciones de utilizar herencia son:

- Reusa código.
- Genera una abstracción.
- Crea un Tipo (para lenguajes con tipado estático).

Prototipos La orientación a objetos basada en prototipos es un estilo de reutilización de comportamiento (herencia) que se logra por medio de la clonación de un objeto ya existente, que sirve como prototipo (Javascript, es el lenguaje orientado a objetos basado en prototipos más conocido).

Autoreferencia Así como un objeto puede conocer a otro objeto teniendo una referencia hacia este, también se puede conocer a sí mismo. Cualquier objeto tiene una autoreferencia, denominada *this* o *self* (según el lenguaje) para poder mandarse algún mensaje a sí mismo.

Así como el *self*, hay otra referencia, denominada *super* o *parent* la cual es equivalente al *this* o *self*, con la particularidad de que le dice al *Method LookUp* "empieza desde uno más arriba". Es decir, que no busca la implementación en la clase dónde está ejecutando el método, sino en la inmediata superior (puede que eso desemboque en que siga subiendo niveles). En el 99% de los casos, sólo deberíamos llamar a *super/parent* para ejecutar el mismo método que estamos ejecutando.

Ejemplo de código:

```
 public Profesor (String nombre, String apellidos, int edad
   , String idProfesor) {
   super(nombre, apellidos, edad);
   this.idProfesor = idProfesor;
}
```

Contratos

Diseño por contratos El diseño por contratos asume que todos los componentes del cliente que invocan una operación en un componente del servidor van a encontrar las precondiciones (y postcondiciones) especificadas como obligatorias para esa operación.

Muchas veces podemos tener algunas cosas para validar en nuestro sistema. Esto es:

- Pre condiciones
- Post condiciones
- Condiciones "permanentes" o invariantes

Por ejemplo: yo siempre que alimente a mi mascota, debo garantizar que tenga hambre. Por lo que quisiera tener algo como:

```
public class Mascota implements Domesticable {
   public void alimentarse(){
      if( tengoHambre() == true) {
         //ejecutar método
      }
   }
}
```

Lo mismo puede suceder con las post condiciones, o condiciones que se deben dar en todo momento.

```
public class Mascota implements Domesticable {
   public void alimentarse(){
      if( tengoHambre() == true) {
         //ejecutar método
```

```
            estoyLleno()
        }
    }
}
```

Para hacer cumplir estos contratos, hay 2 maneras:

1. Validación manual y lanzamiento de excepciones
2. Integrado por el lenguaje o algún Framework

Lo importante es que se entienda el concepto del contrato, para poder interactuar de cierta manera con un componente, debo cumplir ciertos requisitos, tanto antes, durante o después de la interacción.

Contratos y Herencia Básicamente, aplican las mismas ideas y conceptos que en la herencia de comportamiento. Las precondiciones, postcondiciones e invariantes se heredan de clases a subclases.

Sin embargo existen algunos condicionamientos, para garantizar el principio de intercambiabilidad. Que se pueden resumir en la siguiente frase:

Require no more, and promise no less

Relacionado con la L (Liskov Substitution Principle) de SOLID[73], la cual plantea que donde uso una clase, debería poder usar cualquier subclase de ella y que siga funcionando todo correctamente.

[73] SOLID Design Principles de Robert C. Martin's:

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion

https://es.wikipedia.org/wiki/SOLID

Tipos ¿Qué es un tipo? Un tipo describe un conjunto de valores.
La idea de tipo nos permite relacionar:

- Un conjunto de valores que tienen ese tipo o son de ese tipo
- Las operaciones que pueden ser realizadas sobre esos valores.

Los objetivos de un sistema de tipos son:

- Ayudar a detectar errores al programar.
- Guiar al programador sobre las operaciones válidas en un determinado contexto, tanto la documentación como en las ayudas automáticas que puede proveer por ejemplo un IDE.
- En algunos casos el comportamiento de una operación puede variar en función del tipo de los elementos involucrados en la misma. Polimorfismo, sobrecarga (los veremos más adelante), multimétodos, etc.

Podemos hacer 3 clasificaciones de tipado:

- El implícito o explícito. Un lenguaje va a tener tipado explícito sí:

Todos los elementos (variables, métodos, etc.) tienen un tipo definido.

Para que dos objetos sean polimórficos, debo explicitarlos (por una interfaz o por heredar de la misma clase).

- Chequeo dinámico o estático.

La diferencia está en el momento en que es ejecutado el chequeo de tipos. En los lenguajes de chequeo estático (como Java o C++) se verifica en tiempo de compilación, mientras que en los de chequeo dinámico se realiza en tiempo de ejecución (como en Python).

El chequeo dinámico da lugar a un concepto famoso denominado Duck Typing. El cual se basa en que "*si algo tiene pico de pato, camina como pato, y hace cuack, es un pato*". Es decir, toma sentido el tipo al que pertenece un objeto según qué mensaje puedo mandarle y cómo responde, no se me especifica desde antes.

- Estructural o nominal

Hace referencia a si se identifica un tipo por su nombre o por su estructura. Es muy común en los lenguajes del paradigma funcional.

Las combinaciones más frecuentes son:

- Explícito, estático y nominal
- Implícito y dinámico

Casting Es un mecanismo utilizado en los lenguajes de chequeo estático, mediante el cual le "aseguramos" al compilador que cierto objeto pertenece al tipo especificado.
Ejemplo:

```
import java.util.*;
public class TestHerencia {
   public static void main (String [ ] Args) {
      Profesor p1 = new Profesor ("Juan", "Nadrie Garcia",
33, "Prof 22-387-11");
      Medico m1 = new Medico ("Roberto", "Gonzalez Gar-
cia", 33, "Cirujano", "328943784");
      Persona p = new Persona ("Mauro", "Ruiz Perez", 36);
      List<Persona> tlist = new ArrayList<Persona>();
      tlist.add(p1);
      tlist.add(m2);
      tlist.add(p);
      for (int i = 0; i < tlist.size(); i++) tlist.get(i).visualiza();
   }
}
```

Hay que ser muy cuidadosos al usar el casteo, porque podría llevar a "confundir" al compilador y hacer que falle en tiempo de ejecución (perdiendo el beneficio que nos da el chequeo estático), por ejemplo si en el *objetotlist* que recibe objetos de las clases *Medico, Profesor y Persona*, al llamar a métodos no comunes indicados en la clase base *Persona* nuestro programa lanzará una excepción.

Variables y Métodos de Clase También se denominan métodos y variables estáticas. La implementación de los mismos varía dependiendo del lenguaje. En algunos las clases son objetos, en otros, las clases son entes particulares con una instancia asociada, que se pueden utilizar de determinada forma y enviarles mensajes y en algunos no se utiliza ninguna de estas nociones, pero siguen teniendo el mismo concepto.

Las variables de clase nos sirven cuando queremos que nuestros objetos tengan alguna referencia a algún valor, que sea el mismo para todas las instancias de esa clase, y que además pueda cambiar (por eso usamos una variable y no forzamos ese valor en el código del programa, asumiendo que es posible).

Si usáramos una variable de instancia con la intención de definir el mismo valor en todas las instancias de esa clase, y luego queremos cambiar dicho valor, debemos poder encontrar todas las instancias ya existentes para poder mandarles el mensaje para que actualicen su referencia. Y si tenemos muchos objetos, podemos tener un gran problema, no sólo por la complejidad innecesaria del problema, sino también porque el desempeño se vera afectado.

Los métodos de clase, son métodos cuyo receptor / implementador va a ser una clase en lugar de un objeto. El caso más común es el del constructor, donde la clase recibe el "New" y ella sabe generar una nueva instancia de la clase e incluso llama al constructor correspondiente.

Otro caso recurrente es el de querer tener una sola instancia por clase. Para profundizar más en este funcionamiento, ver el patrón de diseño Singleton.

Herencia V.S. Composición Uno de los objetivos que buscamos cuando programamos es reutilizar métodos y funcionalidades, para lograr una mayor mantenibilidad. Dentro de los lenguajes convencionales orientados a objetos existen varias formas de hacer esto, las dos más conocidas son: herencia de clases y composición.

Herencia de clases También conocido como "reutilización de caja blanca", debido a la visibilidad que dan ya que mediante la herencia, las implementaciones de las clases padres se hacen visible a las clases hijas.

Ventajas:

- Se define en tiempo de compilación y no tiempo de ejecución, es decir que conocemos con seguridad que se va a ejecutar.

- Es más sencillo modificar la implementación que está siendo reutilizada.

Desventajas:

- Como dijimos antes, se definen en tiempo de compilación, por lo que no se puede cambiar el comportamiento de un objeto en tiempo de ejecución, lo cual puede ser un requisito de nuestro diseño.
- Se rompe el concepto de encapsulación al exponer los detalles de la implementación en la clase padre.

Composición Conocida como "reutilización de caja negra", a diferencia del caso anterior nosotros no conocemos los detalles de la implementación, ya que la misma se encuentra encapsulada en el objeto al cual estamos invocando.

Ventajas:

- Al tener objetos que hacen referencia a otros objetos, el vínculo se define en tiempo de ejecución y como a dichos objetos se accede mediante su interfase no se rompe el principio de encapsulación (entonces podríamos reemplazar al objeto por otro cuando corre nuestro programa y tener un comportamiento totalmente distinto).
- Permite tener clases más centradas y encapsuladas, haciendo que nuestro diseño tenga más objetos (menos clases) de menor tamaño y con menos responsabilidades.

Agregación Es cuando una o más clases son parte de otra clase. En este caso el objeto no es creado dentro del objeto compuesto, es adherido mediante algún método que lo permita.

Diferencia entre Agregación y Composición

- Las relaciones en una composición son requeridas, en la agregación son opcionales.
- En la composición una clase particular no puede ser compartida por otras clases compuestas, en la agregación esto es posible.
- La relación de vida de la clase particular y la clase contenedora, es muy fuerte, de hecho es la relación más fuerte; tanto que si un objeto de la clase contenedora es destruido la clase particular también lo será. Esto en la agregación no ocurre.

Conclusión Se podría decir que conviene favorecer la composición sobre la herencia, sin embargo esta respuesta no es definitiva, cada uno tiene que identificar la necesidades y el dominio en el cual está trabajando para identificar en que casos conviene usar cada metodología.

Algunas recomendaciones:

- En lenguajes que no soportan herencia múltiple, puede resultar ventajosa la composición.
- La composición genera un diseño más desacoplado, que puede ayudar a hacer pruebas individuales de las clases y de la interacción de los objetos de una más fácil.
- Muchos patrones de diseño favorecen la composición.

7.2 Clase Abstracta

Las clases abstractas son aquellas que por sí mismas no se pueden identificar como algo "concreto" (no existen como tal en el mundo real), pero sí poseen determinadas características que son comunes a otras clases que pueden ser creadas a partir de ellas.

Clase Numero:

```
public abstract class Numero {
    // Genera un nuevo miembro de la clase
    abstract Numero nuevo();
    // Regresa verdadero si es cero
    abstract boolean esCero();
    // Suma
    abstract void suma(Numero a, Numero b);
    // Suma
    abstract void suma(Numero a);
    // Visualiza sin cambiar de linea
    abstract void visualiza();
    // Visualiza cambiando de linea
    void visualizaLN() {
    visualiza();
      System.out.println("");
    }
}
```

Con ella definimos la clase NumeroFraccionario:

```
public class Fraccion extends Numero {
  // Numerador de la fraccion
  private long P;
  // Denominador de la fraccion
  private long Q;
  // Arreglo que contendra a los primos
  static private long []Pr;
  // Numero de primos que contendra el arreglo de primos
  static final private int nPr = 1000;
  // Constructor nulo fraccion 0/1
  public Fraccion() {
    this(0,1);
  }
  // Constructor con un objeto Fraccion
  public Fraccion (Fraccion a) {
    this(a.numerador(), a.denominador());
  }
  // Constructor con entero P Fraccion P/1
  public Fraccion (long p) {
    this(p,1);
  }
  // Costructor con la fraccion P/Q
  public Fraccion (long p, long q) {
    P = p;
    Q = q;
    calculaPrimos();
  }
  // Genera un nuevo miembro de la clase
  public Numero nuevo() {
    Numero c = new Fraccion();
    return c;
  }
  // Calcula los primeros nPr primos
  private void calculaPrimos() {
    int n, i, np;
    Pr = new long[nPr];
```

```
            // Guarda los primeros 2 primos
            Pr[0] = 2;
            Pr[1] = 3;
            np = 2;
            // Empieza la busqueda de primos a partir de 4
            n = 4;
            // Ciclo para buscar los primeros NPB primos
            while (np < nPr) {
               for (i = 0; i < np; i++) {
                  if((n % Pr[i]) == 0) break;
               }
               if(i == np) {
                  Pr[i] = n;
                  np++;
               }
               n++;
            }
         }
         // Visualiza los primos encontrados
         public void visArregloPrimos() {
            System.out.println("Visualiza los primeros " + nPr + "
primos");
            for (int i = 0; i < nPr; i++) System.out.println(Pr[i]);
         }
         // Simplifica el numerador y denominador de la fraccion
         private void simplifica() {
            int i, sw;
            for (i = 0; i < nPr; i++) {
               do {
                  sw = 0;
                  if (Math.abs(P) < 2 || Q < 2) return;
                  if((P % Pr[i] == 0) && (Q % Pr[i] == 0)) {
                     P /= Pr[i];
                     Q /= Pr[i];
                     sw = 1;
                  }
               } while(sw == 1);
            }
```

```
        }
        // Retorna el Numerador
        public long numerador() {
          return P;
        }
        // Retorna el denominador
        public long denominador() {
          return Q;
        }
        // Regresa verdadero si la fraccion es cero
        public boolean esCero() {
          if (numerador() == 0) return true;
          return false;
        }
        // Suma de fracciones con dos operandos
        public void suma(Numero A, Numero B) {
          Fraccion a = new Fraccion( (Fraccion) A);
          Fraccion b = new Fraccion( (Fraccion) B);
          P = a.numerador() * b.denominador() + b.numerador()
* a.denominador();
          Q = a.denominador() * b.denominador();
        }
        // Suma de fracciones con un operando
        public void suma(Numero A) {
          Fraccion a = new Fraccion( (Fraccion) A);
          P = numerador() * a.denominador() + a.numerador() *
denominador();
          Q = denominador() * a.denominador();
        }
        // Visualiza la Fraccion
        public void visualiza() {
          simplifica();
          System.out.print(P);
          System.out.print("/");
          System.out.print(Q);
        }
    };
```

También definimos la clase NumeroComplejo:

```
public class Complejos extends Numero {
  // Parte real del complejo
  private double R;
  // Parte imaginaria del complejo
  private double I;
  // Constructores
  public Complejos() {
    R = 0;
    I = 0;
  }
  // Constructores
  public Complejos(double r, double i) {
    R = r;
    I = i;
  }
  // Constructores
  public Complejos(double r) {
    R = r;
    I = 0;
  }
  // Constructores
  public Complejos(Complejos a) {
    R = a.parteReal();
    I = a.parteImaginaria();
  }
  // Genera un nuevo miembro de la clase
  public Numero nuevo() {
    Numero c = new Complejos();
    return c;
  }
  // Retorna la parte real del complejo
    public double parteReal() {
    return R;
  }
  // Retorna la parte imaginaria del complejo
  public double parteImaginaria() {
```

```
            return I;
          }
          // Regresa verdadero si el complejo es cero
           public boolean esCero() {
           if (parteReal() == 0.0 && parteImaginaria() == 0.0)
    return true;
            return false;
          }
          // Suma con dos operandos
          public void suma(Numero A, Numero B) {
            Complejos a = new Complejos( (Complejos) A);
            Complejos b = new Complejos( (Complejos) B);
            R = a.parteReal() + b.parteReal();
            I = a.parteImaginaria() + b.parteImaginaria();
         }
          // Suma con un operando
          public void suma(Numero A) {
            Complejos a = new Complejos( (Complejos) A);
            R = parteReal() + a.parteReal();
            I = parteImaginaria() + a.parteImaginaria();
          }
          // Visualiza el complejo
          public void visualiza() {
            System.out.print(R + " + " + I + "i");
          }
      };
```

Y también definimos la clase NumeroComplejoFraccionario:

```
      public class ComplejoFraccionario extends Numero {
          // Parte real del complejo
          private Fraccion R;
          // Parte imaginaria del complejo
          private Fraccion I;
          // Constructores
          public ComplejoFraccionario() {
            R = new Fraccion();
            I = new Fraccion();
```

```
        }
        // Constructores
        public ComplejoFraccionario(long r, long i) {
          R = new Fraccion(r);
          I = new Fraccion(i);
        }
        // Constructores
        public ComplejoFraccionario(long r) {
          R = new Fraccion(r);
          I = new Fraccion();
        }
        public ComplejoFraccionario(long rn, long rd, long in, long
    id) {
          R = new Fraccion(rn, rd);
          I = new Fraccion(in, id);
        }
        // Constructores
        public ComplejoFraccionario(ComplejoFraccionario a) {
          R = a.parteReal();
          I = a.parteImaginaria();
        }
        // Genera un nuevo miembro de la clase
          public Numero nuevo() {
          Numero c = new ComplejoFraccionario();
          return c;
        }
        // Retorna la parte real del complejo
        public Fraccion parteReal() {
          return R;
        }
        // Retorna la parte imaginaria del complejo
        public Fraccion parteImaginaria() {
          return I;
        }
        // Regresa verdadero si el complejo es cero
        public boolean esCero() {
          if (parteReal().numerador() == 0.0 && parteImaginaria().numerador()
    == 0.0) return true;
```

```
            return false;
        }
        // Suma con dos operandos
        public void suma(Numero A, Numero B) {
            ComplejoFraccionario a = new ComplejoFraccionario(
(ComplejoFraccionario) A);
            ComplejoFraccionario b = new ComplejoFraccionario(
(ComplejoFraccionario) B);
            R.suma(a.parteReal(), b.parteReal());
            I.suma(a.parteImaginaria(), b.parteImaginaria());
        }
        // Suma con un operando
        public void suma(Numero A) {
            ComplejoFraccionario a = new ComplejoFraccionario(
(ComplejoFraccionario) A);
            R.suma(parteReal(), a.parteReal());
            I.suma(parteImaginaria(), a.parteImaginaria());
        }
        // Visualiza el complejo
        public void visualiza() {
            R.visualiza();
            System.out.print("+");
            I.visualiza();
            System.out.print("i");
        }
    };
```

Pero, ¿podemos definir algún objeto *Número* o todos van a ser *Fraccionario/Complejo/ComplejoFraccionario*/etc.?

En este caso, la clase *Número* no debería ser instanciable, solo quiero utilizarla para que las clases que heredan de esta puedan usar el código definido en ella.

A esto se le llama una **clase abstracta**. Al contrario de las abstractas, las clases que sí son instanciables, se les denomina concretas. Una clase abstracta define sólo la interfaz o firma -el nombre, los parámetros del método y lo que retorna- de algunos de sus métodos. Podemos tener:

- Clases abstractas puras, (véase ejemplo en 7.5.1).

- Clases parcialmente abstractas, (véase ejemplo en 7.2).

Manteniendo el concepto, una clase abstracta puede obligar a que sus subclases implementen cierto método, pero sin definir ningún comportamiento por omisión. A estos métodos, naturalmente, se los denomina métodos abstractos.

Si una clase tiene al menos un método abstracto, entonces debe ser abstracta, porque no tiene sentido que haya un objeto que sea instancia de ella. A fines de comprensión de concepto, podría decirse que una interfaz es una clase abstracta con todos sus métodos abstractos.

Interfaces Una interfaz define un tipo y es una colección de métodos abstractos. De forma similar a una clase abstracta, obliga a quienes implementen la interfaz a implementar los métodos.

Las interfaces no pueden definir la implementación de los métodos de los objetos que las van a utilizar, pero si obligan a dichos objetos a definir esa implementación.

Las interfaces sirven para solventar la limitación de muchos lenguajes, que no soportan la herencia múltiple.

Clases Abstractas Puras vs Interfaces

- Solo se puede extender de una clase abstracta, pero se pueden implementar más de una interfaz en una clase.
- Una interfaz no puede definir atributos.
- Así como no se puede crear instancias de clases abstractas, tampoco es posible crear instancias de interfaces.

7.3 Polimorfismo y Programación Genérica

Es la capacidad que tiene un objeto de poder tratar indistintamente a otros que sean potencialmente distintos, es decir, es la capacidad que tienen distintos objetos de entender un mismo mensaje.

Clase Matriz:

```
public class Matriz <T extends Numero> {
    // Coeficientes de la Matriz
```

```
private Numero [][] M;
// Numero de Renglones
private int ren;
// Numero de Columnas
private int col;
// Constructor de la clase
public Matriz() {
   col = 0;
   ren = 0;
}
// Asigna coeficientes
public void asignaCoeficientes(Numero [][]coef) {
   int i,j;
   ren = coef.length;
   col = coef[0].length;
   M = new Numero[ren][col];
   for (i = 0; i < ren; i++) {
      for (j = 0; j < col; j++) {
         M[i][j] = coef[i][0].nuevo();
         M[i][j] = coef[i][j];
      }
   }
}
// Retorna los coeficientes
Numero coeficiente(int i, int j) {
   if (i < 0 || i >= ren || j < 0 || j >= col) System.out.print("Error");
   return M[i][j];
}
// Retorna el numero de renglones
int renglones() {
   return ren;
}
// Retorna el numero de columnas
int columnas() {
   return col;
}
// Visualiza la Matriz
public void visualiza() {
```

```
            int i, j;
            for (i = 0; i < ren; i++) {
               for (j = 0; j < col; j++) {
                  M[i][j].visualiza();
                  System.out.print(" ");
               }
               System.out.println("");
            }
         }
         // Visualiza la Matriz
         public void visualizaLN() {
            visualiza();
            System.out.println("");
         }
         // Suma con dos operandos
         public void suma(Matriz a, Matriz b) {
            int i, j;
            ren = 0;
            col = 0;
            // Revisa el tamano de las Matrices a sumar
            if(a.renglones() != b.renglones() || a.columnas() != b.columnas())
{
               System.out.println("Error, las dimensiones no son iguales");
            } else {
               // Solicita memoria para la Matriz resultante
               ren = a.renglones();
               col = a.columnas();
               M = new Numero[ren][col];
               for (i = 0; i < ren; i++) {
                  for (j = 0; j < col; j++) {
                     M[i][j] = a.coeficiente(i,j).nuevo();
                     M[i][j].suma(a.coeficiente(i,j), b.coeficiente(i,j));
                  }
               }
            }
         }
         // Suma con un operando
         public void suma(Matriz a) {
```

```
        int i, j;
        ren = 0;
        col = 0;
        // Revisa el tamano de las Matrices a sumar
        if(renglones() != a.renglones() || columnas() != a.columnas())
{
          System.out.println("Error, las dimensiones no son iguales");
        } else {
          // Solicita memoria para la Matriz resultante
          for (i = 0; i < ren; i++) {
            for (j = 0; j < col; j++) {
              M[i][j].suma(a.coeficiente(i,j));
            }
          }
        }
      }
      // Funcion Principal ....
      public static void main(String[] args) {
        // Ejemplito de Matrices de coeficientes Complejos
        Complejos [][]x = new Complejos[2][2];
        x[0][0] = new Complejos(2, 3);
        x[1][0] = new Complejos(2, 2);
        x[0][1] = new Complejos(3, 2);
        x[1][1] = new Complejos(3, 2);
        Complejos [][]y = new Complejos[2][2];
        y[0][0] = new Complejos(-2, -3);
        y[1][0] = new Complejos(2, 3);
        y[0][1] = new Complejos(3, 3);
        y[1][1] = new Complejos(3, 3);
        Matriz<Complejos> X = new Matriz<Complejos>();
        X.asignaCoeficientes(x);
        Matriz<Complejos> Y = new Matriz<Complejos>();
        Y.asignaCoeficientes(y);
        Matriz<Complejos> Z = new Matriz<Complejos>();
        System.out.println("Complejo");
        X.visualizaLN();
        System.out.println("+");
        Y.visualizaLN();
```

```
            Z.suma(X, Y);
            System.out.println("======================");
            Z.visualizaLN();
            System.out.println("");
            System.out.println("");

            // Ejemplito de Matrices de coeficionets ComplejoFrac-
    cionario
            ComplejoFraccionario [][]xx = new ComplejoFraccionario[2][2];
            xx[0][0] = new ComplejoFraccionario(2, 3, 4, 5);
            xx[1][0] = new ComplejoFraccionario(2, 2, 3, 7);
            xx[0][1] = new ComplejoFraccionario(3, 2);
            xx[1][1] = new ComplejoFraccionario(3, 2);
            ComplejoFraccionario [][]xy = new ComplejoFraccionario[2][2];
            xy[0][0] = new ComplejoFraccionario(-2, 3, 5, 6);
            xy[1][0] = new ComplejoFraccionario(2, 3, 7, 9);
            xy[0][1] = new ComplejoFraccionario(3, 3);
            xy[1][1] = new ComplejoFraccionario(3, 3);
            Matriz<ComplejoFraccionario> xX = new Matriz<ComplejoFraccionario>();
            xX.asignaCoeficientes(xx);
            Matriz<ComplejoFraccionario> xY = new Matriz<ComplejoFraccionario>();
            xY.asignaCoeficientes(xy);
            Matriz<ComplejoFraccionario> xZ = new Matriz<ComplejoFraccionario>();
            System.out.println("Complejo Fraccionario");
            xX.visualizaLN();
            System.out.println("+");
            xY.visualizaLN();
            xZ.suma(xX, xY);
            System.out.println("======================");
            xZ.visualizaLN();
            System.out.println("");
            System.out.println("");
        }
    }
```

Este concepto se conoce como polimorfismo. Es la capacidad de intercambiar un objeto con otro, abstrayendo al que se conoce desde su implementación.

¿Por qué es importante diseñar polimórficamente? Para ganar extensibilidad. Así, al implementar otra clase que extienda a la de tipo *Numero*, no necesitaremos modificar la clase *Matriz*, es suficiente con crearle una implementación y encajará perfectamente en el sistema.

Entonces, para hacerlo genérico, se establece un contrato: "Todas las clases de tipo *Numero* deben entender el mensaje de *sumar*".

En general, un contrato establece un acuerdo entre dos (o más) partes. Si lo cumplimos, el sistema va a tener una funcionalidad o comportamiento definido. De alguna forma regula la interacción entre dos módulos de nuestra aplicación. Entonces intercambiando un módulo por otro que cumple el mismo contrato debería ser transparente para el otro módulo.

La parte del contrato más frecuentemente usada, va a ser lo que denominamos interfaz. La interfaz de un objeto es el conjunto de mensajes que entiende.

Adicionalmente, están las construcciones específicas denominadas interfaces. Estas, no son más que una especificación del conjunto de mensajes que tienen que cumplir aquellos que la implementen.

Una buena regla para identificar la necesidad de una interfaz es establecer un adjetivo común a los objetos que quiero que la implementen (*sumar/ restar/ dividir/ multiplicar*), a diferencia de una superclase, las cuales suelen ser sustantivos (*Animal/ Persona/ Auto*).

Aprovechando todo lo desarrollado hasta aquí, mostramos un par de ejemplos de programación genérica, mediante el ejemplo de la clase Vector y la clase Polinomio genérico y que soporta los tipos de números que extienden a la clase *Numero* desarrollados anteriormente y otros más.

7.4 Excepciones

¿Cuántas veces al trabajar con un programa de cómputo, este nos manda algún mensaje de error poco descriptivo?, esto probablemente se deba a un bajo nivel de detalle en el manejo de excepciones dentro de nuestro programa. El manejo de excepciones consiste en controlar los errores que surjan dentro de nuestro programa para poder tratarlos debidamente.

Tratarlos debidamente implica:

- Mostrar un mensaje más amigable a los usuarios (la más común).
- Revertir (Rollback) alguna transacción.

- Escribir en un archivo tipo registro (Log) el error para analizarlo.
- Si el error no afecta el flujo de trabajo: contenerlo para que el flujo siga.

En Java los errores en tiempo de ejecución (cuando se esta ejecutando el programa) se denominan excepciones, y esto ocurre cuando se produce un error en alguna de las instrucciones de nuestro programa, como por ejemplo cuando se hace una división entre cero, cuando un objeto es *null* y no puede serlo, cuando no se abre correctamente un archivo, etc. Cuando se produce una excepción se muestra en la pantalla un mensaje de error y finaliza la ejecución del programa.

En Java (al igual que en otros lenguajes de programación), existen mucho tipos de excepciones. En lo referente a las excepciones hay que decir que se aprenden a base experiencia, de encontrarte con ellas y de saber solucionarlas. Cuando en Java se produce una excepción se crear un objeto de una determina clase -dependiendo del tipo de error que se haya producido-, que mantendrá la información sobre el error producido y nos proporcionará los métodos necesarios para obtener dicha información. Estas clases tienen como clase padre la clase *Throwable*, por tanto se mantiene una jerarquía en las excepciones.

A continuación mostramos algunas de las clases para que nos hagamos una idea de la jerarquía que siguen las excepciones, pero existen muchísimas más excepciones que las que mostramos:

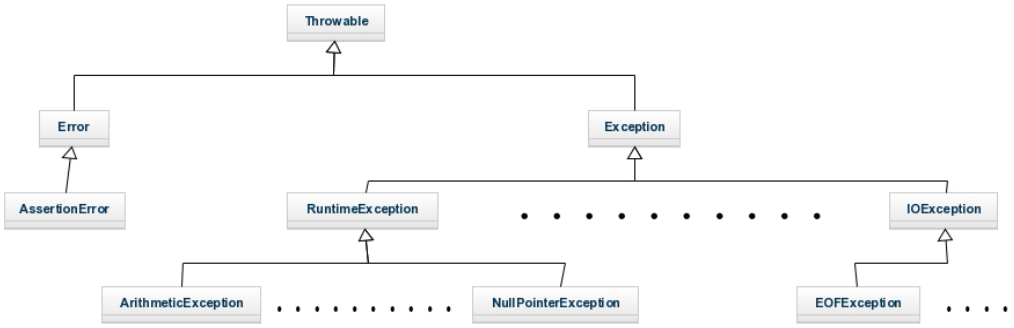


Figura 1: Jerarquía de clases para el manejo de excepciones en Java.

En Java, cuando se produce un error en un método, "se lanza" un objeto *Throwable*, cualquier método que haya llamado al método puede "capturar la

excepción" y tomar las medidas que estime oportunas. Tras capturar la excepción, el control no vuelve al método en el que se produjo la excepción, sino que la ejecución del programa continúa en el punto donde se haya capturado la excepción.

Consecuencia:

> Nunca más tendremos que preocuparnos de "diseñar" códigos de error.

Throwable Clase base que representa todo lo que se puede "lanzar" en Java, contiene una instantánea del estado de la pila en el momento en el que se creó el objeto ("*Stack Trace*" o "*Call Chain*"), también almacena un mensaje (variable de instancia de tipo *String*) que podemos utilizar para detallar qué error se produjo. Además puede tener una causa, también de tipo *Throwable*, que permite representar el error que causó este error.

Error Subclase de *Throwable* que indica problemas graves que una aplicación no debería intentar solucionar (documentación de Java), ejemplos:

> Memoria agotada, error interno de la JVM, etc.

Exception *Exception* y sus subclases indican situaciones que una aplicación debería tratar de forma razonable.

Los dos tipos principales de excepciones son:

- *RuntimeException*, errores del programador, como una división por cero o el acceso fuera de los límites de un array.
- *IOException*, errores que no puede evitar el programador, generalmente relacionados con la entrada/salida del programa.

Captura de excepciones: Bloques *try...catch* Se utilizan en Java para capturar las excepciones que se hayan podido producir en el bloque de código delimitado por *try* y *catch*. En cuanto se produce la excepción, la ejecución del bloque *try* termina y la cláusula *catch* recibe como argumento un objeto *Throwable*.

Ejemplo básico:

```
// Bloque 1
try {
   // Bloque 2
} catch (Exception error) {
   // Bloque 3
}
// Bloque 4
```

Flujo de la ejecución en el programa:

Sin excepciones: 1 -> 2 -> 4

Con una excepción en el bloque 2: 1 -> 2(excepción) -> 3 -> 4

Con una excepción en el bloque 1: 1(excepción)

Otro ejemplo:

```
// Bloque 1
try {
   // Bloque 2
} catch (ArithmeticException ae) {
   // Bloque 3
} catch (NullPointerException ne) {
   // Bloque 4
}
// Bloque 5
```

Flujo de la ejecución en el programa:

Sin excepciones: 1 -> 2 -> 5

Excepción de tipo aritmético: 1 -> 2(excepción) -> 3 -> 5

Acceso a un objeto nulo (null): 1 -> 2(excepción) -> 4 -> 5

Excepción de otro tipo diferente: 1 -> 2(excepción)

Un ejemplo más:

```
// Bloque1
try {
   // Bloque 2
} catch (ArithmeticException ae) {
   // Bloque 3
} catch (Exception error) {
   // Bloque 4
}
// Bloque 5
```

Flujo de la ejecución en el programa:

Sin excepciones: 1 -> 2 -> 5

Excepción de tipo aritmético: 1 -> 2(excepción) -> 3 -> 5

Excepción de otro tipo diferente: 1 -> 2(excepción) -> 4 -> 5

¡Ojo! Las cláusulas *catch* se comprueban en orden, ejemplo:

```
// Bloque1
try {
   // Bloque 2
} catch (Exception error) {
   // Bloque 3
} catch (ArithmeticException ae) {
   // Bloque 4
}
// Bloque 5
```

Flujo de la ejecución en el programa:

Sin excepciones: 1 -> 2 -> 5

Excepción de tipo aritmético: 1 -> 2(excepción) -> 3 -> 5

Excepción de otro tipo diferente: 1 -> 2(excepción) -> 3 -> 5

¡ El bloque 4 nunca se llegará a ejecutar !

La cláusula *finally* En ocasiones, nos interesa ejecutar un fragmento de código independientemente de si se produce o no una excepción (por ejemplo, cerrar un archivo que estemos manipulando).

Ejemplo:

```
// Bloque1
try {
   // Bloque 2
} catch (ArithmeticException ae) {
   // Bloque 3
} finally {
   // Bloque 4
}
// Bloque 5
```

Flujo de la ejecución en el programa:

Sin excepciones: 1 -> 2 -> 4 -> 5

Excepción de tipo aritmético: 1 -> 2(excepción) -> 3 -> 4 -> 5

Excepción de otro tipo diferente: 1 -> 2(excepción) -> 4

Si el cuerpo del bloque *try* llega a comenzar su ejecución, el bloque *finally* siempre se ejecutará:

- Detrás del bloque *try* si no se producen excepciones
- Después de un bloque *catch* si este captura la excepción.
- Justo después de que se produzca la excepción si ninguna cláusula *catch* captura la excepción y antes de que la excepción se propague hacia arriba.

Lanzamiento de Excepciones La sentencia *throw* se utiliza en Java para lanzar objetos de tipo *Throwable*

```
throw new Exception("Mensaje de error...");
```

Cuando se lanza una excepción:

- Se sale inmediatamente del bloque de código actual.
- Si el bloque tiene asociada una cláusula *catch* adecuada para el tipo de la excepción generada, se ejecuta el cuerpo de la cláusula *catch*.
- Si no, se sale inmediatamente del bloque (o método) dentro del cual está el bloque en el que se produjo la excepción y se busca una cláusula *catch* apropiada.
- El proceso continúa hasta llegar al método main de la aplicación. Si ahí tampoco existe una cláusula *catch* adecuada, la máquina virtual Java finaliza su ejecución con un mensaje de error.

Propagación de Excepciones (throws) Si en el cuerpo de un método se lanza una excepción -de un tipo derivado de la clase *Exception*-, en la cabecera del método hay que añadir la cláusula *throws* que incluye una lista de los tipos de excepciones que se pueden producir al invocar el método.

Ejemplo:

```
public String leerFichero (String nombreFichero)
throws IOException
...
```

Las excepciones de tipo *RuntimeException* (que son muy comunes) no es necesario declararlas en la cláusula throws. Al implementar un método, hay que decidir si las excepciones se propagarán hacia arriba (*throws*) o se capturarán en el propio método (*catch*)

1. Ejemplo de un método que propaga una excepción:

```
public void f() throws IOException {
   // Fragmento de codigo que puede
   // lanzar una excepción de tipo IOException
}
```

NOTA: Un método puede lanzar una excepción al crear explícitamente un objeto *Throwable* y lanzarlo con *throw*, o bien porque llame a un método que genere la excepción y no la capture.

2. Ejemplo de un método equivalente que no propaga la excepción:

```
public void f() {
   // Fragmento de código libre de excepciones
   try {
      // Fragmento de codigo que puede
      // lanzar una excepcion de tipo IOException
      // (p.ej. Acceso a un archivo)
   } catch (IOException error) {
      // Tratamiento de la excepcion
   } finally {
      // Liberar recursos (siempre se hace)
   }
}
```

Creación de nuevos tipos de excepciones Un nuevo tipo de excepción puede crearse fácilmente, basta con definir una subclase de un tipo de excepción ya existente.

```
public DivideByZeroException
extends ArithmeticException {
   public DivideByZeroException(String Message) {
      super(message);
   }
}
```

Una excepción de este tipo puede entonces lanzarse como cualquier otra excepción:

```
public double dividir(int num, int den)
throws DivideByZeroException {
   if (den==0) throw new DivideByZeroException("Error!");
   return ((double) num/(double)den);
}
```

NOTA: Las aplicaciones suelen definir sus propias subclases de la clase *Exception* para representar situaciones excepcionales específicas de cada aplicación.

7.5 Altas, Bajas y Cambios

En programación, crear, leer, actualizar y borrar (con el acrónimo CRUD[74]) son las cuatro funciones básicas de la persistencia de Bases de Datos. Términos alternativos son usados a veces cuando se definen las cuatro funciones básicas de CRUD, como "recuperar" en vez de "leer", "modificar" en vez de "actualizar" o "destruir" en vez de "borrar". CRUD se usa también a veces para describir convenciones de interfaz de usuario que facilita la vista, búsqueda y modificación de la información; a menudo se usa en programación de formularios (Forms) e informes (Reports). El acrónimo puede extenderse a CRUDL[75] para cubrir el listado de gran cantidad de datos que conllevan una complejidad tal como paginación cuando los registros de datos son demasiado grandes para alojarse fácilmente en memoria.

7.5.1 Ejemplo en Java

En esta sección, desarrollaremos un sistema a modo de ejemplo de Altas, Bajas y Cambios (ABC) además de Leer y Grabar los datos capturados -para manipular la lista de los registros usaremos un *ArrayList*[76]- usando programación genérica para desarrollar un par de ejemplos -Directorio Telefónico y Catálogo de CDs- que pueden ser usados por separado o como además se mostrará, en un arreglo de objetos que permite su manipulación de forma simultáneo, mediante un único sistema de cómputo que aproveche las facilidades de la programación genérica mostrada en la siguiente jerarquía de

[74] En informática, CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un Software.

En algunos lugares, se utilizan las siglas ABM para lo mismo ("Alta, Baja y Modificación"), obviando la operación de "obtener"; el acrónimo ABC para "Altas, Bajas y Cambios"; ABML siendo la última letra (L) de "listar, listado o lectura"; ABMC siendo la 'C' de "Consulta"; o bien CLAB que sería la traducción literal del acrónimo ("Crear, Leer, Actualizar y Borrar") también se llega a usar el acrónimo ABCC ("Altas, Bajas, Cambios y Consultas").

[75] El término fue popularizado por primera vez por James Martín en su libro del año 1980 Managing the Data-base Enviroment.

[76] Para manipular estructuras de datos que nos permita añadir, eliminar y modificar elementos —pueden ser objetos o elementos atómicos— de forma transparente para el programador en Java se desarrollo entre otras opciones al *ArrayList*. Esta clase permite almacenar datos en memoria de forma similar a los *Arrays*, con la ventaja de que el número de elementos que almacena, lo hace de forma dinámica.

clases.

Para ello, primero desarrollamos una clase abstracta pura que defina los comportamientos que harán la manipulación de cada componente (registro) de nuestro sistema de Altas, Bajas y Cambios, mediante el siguiente código:

```
import java.io.Serializable;[77]
/// Clase Base para manipular registros
public abstract class Registro implements Serializable {
  private static final long serialVersionUID = 1L;
  abstract int visualizaRegistro();
  abstract int modificarRegistro();
  abstract int adicionaRegistro();
};
```

Para luego definir los comportamientos mediante la herencia, para el directorio telefónico:

```
import java.util.Scanner;[78]
/// Clase para manipular registros telefonicos
  public class RegTelefonico extends Registro {
  private String Nombre;
  private String Direccion;
  private String Telefonos;
  /// Visualiza el contenido del registro
  int visualizaRegistro() {
    System.out.println("Nombre: " + Nombre);
    System.out.println("Direccion: " + Direccion);
    System.out.println("Telefonos: " + Telefonos);
    return 0; //Ok
  }
  /// Adiciona un registro
  int adicionaRegistro() {
    Scanner teclado = new Scanner(System.in);
```

[77] Para usar *ArrayList* y que nos permite leer y grabar al conjunto de registros de forma unificada, necesitamos usar *java.io.Serializable*. E implementar la clase *Registro* a partir de la clase *Serializable*, además de definir:

private static final long serialVersionUID = 1L;

[78] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
        System.out.print("Nombre: ");
        Nombre = teclado.nextLine();
        System.out.print("Direccion: ");
        Direccion = teclado.nextLine();
        System.out.print("Telefonos: ");
        Telefonos = teclado.nextLine();
        return 0; // Ok
      }
      /// Modifica el contenido del registro
      int modificarRegistro() {
        Scanner teclado = new Scanner(System.in);
        // Visualiza el nombre y permite su modificacion
        System.out.println("Nombre: " + Nombre);
        System.out.print("Nuevo Nombre: ");
        Nombre = teclado.nextLine();
        System.out.println("Direccion: " + Direccion);
        System.out.print("Nueva Direccion: ");
        Direccion = teclado.nextLine();
        System.out.println("Telefonos: " + Telefonos);
        System.out.print("Nuevos Telefonos: ");
        Telefonos = teclado.nextLine();
        return 0; // Ok
      }
    };
```

y del catálogo de CDs, mediante:

```
    import java.util.Scanner;[79]
    /// Clase para manipular registros de Cds
    public class RegCDs extends Registro {
      private String Titulo;
      private String Artista;
      private int NumCanciones;
      private String Canciones;
      /// Visualiza el contenido del registro
      public int visualizaRegistro() {
```

[79] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
        System.out.println("Titulo: " + Titulo);
        System.out.println("Artista: " + Artista);
        System.out.println("Numero de caciones: "
    + NumCanciones);
        System.out.println("Canciones: " + Canciones);
        return 0; //Ok
      }
      /// Adiciona un registro
      public int adicionaRegistro() {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Titulo: ");
        Titulo = teclado.nextLine();
        System.out.print("Artista: ");
        Artista = teclado.nextLine();
        System.out.print("Numero de caciones: ");
        NumCanciones = teclado.nextInt();
        System.out.print("Canciones: ");
        Canciones = teclado.nextLine();
        Canciones = teclado.nextLine();
        return 0; // Ok
      }
      /// Modifica el contenido del registro
      int modificarRegistro() {
        Scanner teclado = new Scanner(System.in);
        // Visualiza el nombre y permite su modificacion
        System.out.println("Titulo: " + Titulo);
        System.out.print("Nuevo Titulo: ");
        Titulo = teclado.nextLine();
        System.out.println("Artista: " + Artista);
        System.out.print("Nuevo Artista: ");
        Artista = teclado.nextLine();
        System.out.println("Numero de caciones: "
    + NumCanciones);
        System.out.print("Nuevo Numero de caciones: ");
        NumCanciones = teclado.nextInt();
        System.out.println("Canciones: " + Canciones);
        System.out.print("Nuevas Canciones: ");
        Canciones = teclado.nextLine();
```

```
        Canciones = teclado.nextLine();
        return 0; // Ok
    }
};
```

Los códigos anteriores definen los comportamientos genéricos que son la implementación de los comportamientos abstractos:

```
abstract int visualizaRegistro();
abstract int modificarRegistro();
abstract int adicionaRegistro();
```

Ahora implementamos la clase abstracta que soportará los comportamientos que manipulen a las clases anteriores mediante un *ArrayList* y que permitan la implementación de las Altas, Bajas y Cambios, mediante:

```
import java.io.*;[80]
import java.util.*;[81]
/// Clase base para manipular Altas-Bajas-Cambios.
public abstract class EstructuraABC {
  /// Lista
  public ArrayList<Registro> lista;
  public EstructuraABC() {
    lista = new ArrayList<Registro>();
  }
  /// Borra todo el contenido de la lista
  private void borrarTodo() {
    lista.clear();
  }
  /// Adiciona un registro al final de la lista
  abstract int adicionar();
  /// Visualiza el contenido del registro solicitado
  public int visualiza(int pos) {
    if (pos >= 0 && pos < lista.size()) {
      lista.get(pos).visualizaRegistro();
      return 0;
```

[80] Para permitir leer y grabar archivos.
[81] Para definir *ArrayList*.

```
    }
    return 1;
  }
  /// Visualiza todo el contenido de la lista
  public int visualizaTodos() {
    Iterator<Registro> iter = lista.iterator();
    while (iter.hasNext()) {
      iter.next().visualizaRegistro();
    }
    return 0;
  }
  /// Visualiza el registro de la lista solicitado
  int borrar(int pos) {
    if (pos >= 0 && pos < lista.size()) {
      lista.remove(pos);
      return 0;
    }
    return 1;
  }
  /// Modifica el registro de la lista solicitado
  int modificar(int pos) {
    if (pos >= 0 && pos < lista.size()) {
      lista.get(pos).modificarRegistro();
      return 0;
    }
    return 1;
  }
  /// Leer una lista de disco
  @SuppressWarnings("unchecked")
  int leer(String arch) {
    try {
      FileInputStream fileIn = new FileInputStream(arch);
      ObjectInputStream in = new ObjectInputStream(fileIn);
      Object obj = in.readObject();
      lista = (ArrayList<Registro>) obj;
      in.close();
      fileIn.close();
      visualizaTodos();
```

```
    } catch(Exception e){e.printStackTrace();}
    return 0;
  }
  /// Grabar una lista en disco
  int grabar(String arch) {
    try {
      FileOutputStream fileOut = new FileOutputStream(arch);
      ObjectOutputStream out = new ObjectOutputStream(fileOut);
      out.writeObject(lista);
      out.close();
      fileOut.close();
    } catch(Exception e){}
    return 0;
  }
  /// Regresa el numero de registros en la lista
  int regresaNRegs() {
    return lista.size();
  }
};
```

Una vez definidos los códigos anteriores, podemos ya implementar la clase que permita definir mediante herencia la manipulación de los registros telefónicos haciendo uso de la herencia del código de Altas, Bajas y Cambios, mediante:

```
/// Clase para manipular directorio telefonicos
class DirecTelefonico extends EstructuraABC {
  // Adiciona registro al final de la lista
  public int adicionar() {
    RegTelefonico reg = new RegTelefonico();
    reg.adicionaRegistro();
    lista.add(reg);
    return 0; // Ok
  }
};
```

Y para el Catálogo de CDs, mediante:

```
/// Clase para manipular catalogos de CDs
public class CatalogoCDs extends EstructuraABC {
  // Adiciona registro al final de la lista
  int adicionar() {
    RegCDs reg = new RegCDs();
    reg.adicionaRegistro();
    lista.add(reg);
    return 0;
  }
};
```

Ahora definiremos un menú para que el usuario manipule nuestros sistemas de Altas, Bajas y Cambios, por ejemplo para el Directorio Telefónico, mediante:

```
import java.util.Scanner;[82]
public class Test {
  public static void limpiar() {
    for (int i = 0; i < 20; i++) System.out.println("");
  }
  public static void visualizaMenu() {
    System.out.println("Menu");
    System.out.println("");
    System.out.println("1) Agregar");
    System.out.println("2) Modificar");
    System.out.println("3) Borrar");
    System.out.println("4) Visualizar todos");
    System.out.println("8) Leer");
    System.out.println("9) Grabar");
    System.out.println("0) Salir");
  }
  // Funcion Principal ....
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    // Manipulacion del directorio telefonico
    DirecTelefonico man = new DirecTelefonico();
```

[82] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
        limpiar();
        int op = 1, reg;
        do {
          System.out.println("Numero de registros: "
    + man.regresaNRegs());
          visualizaManu();
          System.out.println("");
          System.out.println("Opcion: ");
          op = teclado.nextInt();
          switch(op) {
            case 1:
              man.adicionar();
              break;
            case 2:
              System.out.println("Registro: ");
              reg = teclado.nextInt();
              man.modificar(reg);
              break;
            case 3:
              System.out.println("Registro: ");
              reg = teclado.nextInt();
              man.borrar(reg);
              break;
            case 4:
              man.visualizaTodos();
              break;
            case 8:
              man.leer("DirTelefonico.dat");
              break;
            case 9:
              man.grabar("DirTelefonico.dat");
              break;
          }
        } while(op != 0);
      }
    }
```

Y para el Catálogo de CDs, mediante:

```
import java.util.Scanner;[83]
public class Test {
  public static void limpiar() {
    for (int i = 0; i < 20; i++) System.out.println("");
  }
  public static void visualizaMenu() {
    System.out.println("Menu");
    System.out.println("");
    System.out.println("1) Agregar");
    System.out.println("2) Modificar");
    System.out.println("3) Borrar");
    System.out.println("4) Visualizar todos");
    System.out.println("8) Leer");
    System.out.println("9) Grabar");
    System.out.println("0) Salir");
  }
  // Funcion Principal ....
  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    // Manipulacion del directorio telefonico
    CatalogoCDs man = new CatalogoCDs();
    limpiar();
    int op = 1, reg;
    do {
      System.out.println("Numero de registros: "
+ man.regresaNRegs());
      visualizaManu();
      System.out.println("");
      System.out.println("Opcion: ");
      op = teclado.nextInt();
      switch(op) {
        case 1:
          man.adicionar();
          break;
        case 2:
```

[83] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
                System.out.println("Registro: ");
                reg = teclado.nextInt();
                man.modificar(reg);
                break;
            case 3:
                System.out.println("Registro: ");
                reg = teclado.nextInt();
                man.borrar(reg);
                break;
            case 4:
                man.visualizaTodos();
                break;
            case 8:
                man.leer("DirTelefonico.dat");
                break;
            case 9:
                man.grabar("DirTelefonico.dat");
                break;
          }
        } while(op != 0);
    }
}
```

Aún mejor, podemos usar el Directorio telefónico y el Catálogo de CDs simultáneamente, permitiendo al usuario cambiarse entre uno u otro según sea necesario, mediante:

```
import java.util.Scanner;[84]
public class Test {
    public static String[] Archivo = {"DirTelefonico.dat", "Cat-
alogoCDs.dat"};
    public static void limpiar() {
        for (int i = 0; i < 20; i++) System.out.println("");
    }
    public static void visualizaMenu() {
        System.out.println("Menu");
```

[84] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
            System.out.println("");
            System.out.println("1) Agregar");
            System.out.println("2) Modificar");
            System.out.println("3) Borrar");
            System.out.println("4) Visualizar todos");
            System.out.println("8) Leer");
            System.out.println("9) Grabar");
            System.out.println("\n10) Cambiar entre DirecTelefon-
ico y CatalogoCDs");
            System.out.println("0) Salir");
         }
         // Funcion Principal ....
         public static void main(String[] args) {
            Scanner teclado = new Scanner(System.in);
            EstructuraABC []man= new EstructuraABC[2];
            man[0] = new DirecTelefonico();
            man[1] = new CatalogoCDs();
            limpiar();
            int op = 1, act = 0, reg;
            do {
               System.out.println("Numero de registros: "
   + man[act].regresaNRegs() + " en " + Archivo[act]);
               System.out.println("");
               visualizaMenu();
               System.out.println("Opcion: ");
               op = teclado.nextInt();
               switch(op) {
                  case 1:
                     man[act].adicionar();
                     break;
                  case 2:
                     System.out.println("Registro: ");
                     reg = teclado.nextInt();
                     man[act].modificar(reg);
                     break;
                  case 3:
                     System.out.println("Registro: ");
                     reg = teclado.nextInt();
```

```
                    man[act].borrar(reg);
                    break;
                case 4:
                    man[act].visualizaTodos();
                    break;
                case 8:
                    man[act].leer(Archivo[act]);
                    break;
                case 9:
                    man[act].grabar(Archivo[act]);
                    break;
                case 10:
                    if (act == 0) act = 1;
                    else act = 0;
                    break;
                default:
                    System.out.println("Opcion no reconocidad");
            }
        } while(op != 0);
    }
}
```

De esta forma, tenemos una clase genérica de Altas, Bajas y Cambios que nos permite manipular uno o múltiples ejemplos -Directorio Telefónico y el Catálogo de CDs- para hacer Altas, Bajas y Cambios en un solo sistema computacional que es flexible, extendible y de fácil mantenimiento, además de permitir el manejo simultáneo de múltiples sistemas de ABC.

7.5.2 Ejemplo en Python

En esta sección, desarrollaremos un sistema a modo de ejemplo de Altas, Bajas y Cambios (ABC) además de Leer y Grabar los datos capturados - para manipular la lista de los registros usaremos *list*[85] *y pickle*[86] - usando

[85] Para manipular estructuras de datos que nos permita añadir, eliminar y modificar elementos -pueden ser objetos o elementos atómicos- de forma transparente para el programador. Esta clase permite almacenar datos en memoria, con la ventaja de que el número de elementos que almacena lo hace de forma dinámica.

[86] Alternativamente mostramos el mismo ejemplo usando shelve en vez de pickle en la siguiente jerarquía de clases.

programación genérica para desarrollar un par de ejemplos de productos - Televisor y Reproductor- que pueden ser usados por separado o como además se mostrará, en una lista de objetos que permite su manipulación de forma simultánea, mediante un único sistema de cómputo que aproveche las facilidades de la programación mostrada en la siguiente jerarquía de clases.

Para ello, primero desarrollamos una clase base *Producto* que defina los comportamientos comunes de nuestro sistema de Altas, Bajas y Cambios, mediante el siguiente código:

```
class Producto:
  """Definicion de la clase producto"""
  def __init__(self):
    """Constructor de la clase producto"""
    self.__id = None
    self.__nomb = None
    self.__precio = None
    self.__cant = None
  def inicializa(self, id, nomb, precio, cant):
    """Inicializa la clase producto"""
    self.__id = id
    self.__nomb = nomb
    self.__precio = precio
    self.__cant = cant
  def captura(self):
    """Captura la clase producto"""
    self.__id = input("Identificador: ")
    self.__nomb = input("Nombre: ")
    self.__precio = float(input("Precio: "))
    self.__cant = int(input("Cantidad: "))
  def identificador(self):
    """Retorna el identificador producto"""
    return self.__id
  # Acceso al valor self.__id mediante la variable id
  id = property(identificador)
  def visualizar(self):
    """Visualiza la clase producto"""
    print(self)
  def __repr__(self):
```

```
        """Visualiza la clase producto"""
        a = "\nIdentificador: {0}\n".format(self.__id)
        a += "Nombre: {0}\n".format(self.__nomb)
        a += "Precio: {:,.2f}\n".format(self.__precio)
        a += "Cantidad: {0}\n".format(self.__cant)
        return a
```

Para luego definir los comportamientos mediante la herencia, para el producto Televisor:

```
class Televisor(Producto):
    """Definicion de la clase Televisor"""
    def __init__(self):
        """Constructor de la clase Televisor"""
        Producto.__init__(self)
        self.__pulg = None
        self.__resol = None
    def inicializa(self, id, nomb, precio, cant, pulg, resol):
        """Inicializa la clase Televisor"""
        Producto.inicializa(self, id, nomb, precio, cant)
        self.__pulg = pulg
        self.__resol = resol
    def captura(self):
        """Captura la clase Televisor"""
        Producto.captura(self)
        self.__pulg = int(input("Pulgadas: "))
        self.__resol = input("Resolucion: ")
    def __repr__(self):
        """Visualiza la clase Televisor"""
        a = Producto.__repr__(self)
        a += "Pulgadas: {0}\n".format(self.__pulg)
        a += "Resolucion: {0}\n".format(self.__resol)
        return a
```

y el producto Reproductor, mediante:

```
class Reproductor(Producto):
    """Definicion de la clase Reproductor"""
```

```
    def __init__(self):
        """Constructor de la clase Reproductor"""
        Producto.__init__(self)
        self.__bocinas = None
    def inicializa(self, id, nomb, precio, cant, bocinas):
        """Inicializa la clase Reproductor"""
        Producto.inicializa(self, id, nomb, precio, cant)
        self.__bocinas = bocinas
    def captura(self):
        """Captura la clase Reproductor"""
        Producto.captura(self)
        self.__bocinas = int(input("Bocinas: "))
    def __repr__(self):
        """Visualiza la clase Reproductor"""
        a = Producto.__repr__(self)
        a += "Bocinas: {0}\n".format(self.__bocinas)
        return a
```

Ahora implementamos la clase que soportará los comportamientos que manipulen a las clases anteriores mediante una lista *list* y que permitan la implementación de las Altas, Bajas y Cambios, mediante:

```
class Prueba:
    """ Clase de prueba """
    def __init__(self):
        """Constructor de la clase"""
        self.__lista = [] # Lista para los objetos creados
        self.__archivo = "ArchivoBD"
        self.__msw = 1
        if os.path.exists(self.__archivo):
            with open(self.__archivo, "rb") as fp:
                self.__lista = pickle.load(fp)
    def __del__(self):
        """Destructor"""
    def menu(self):
        """menu """
        sw = {
            "1": self.altas,
```

```
            "2": self.bajas,
            "3": self.cambios,
            "4": self.visualiza,
            "9": self.salir,
        }
        while self.__msw:
            # os.system('clear') # NOTA para windows tienes que
cambiar clear por cls
            print("Selecciona una opcion")
            print("\t1 - Altas")
            print("\t2 - Bajas")
            print("\t3 - Cambios")
            print("\t4 - Visualizar")
            print("\t9 - salir")
            # solicituamos una opcion al usuario
            opcionMenu = input("inserta un numero >> ")
            sw.get(opcionMenu, self.otro)()
    def altas(self):
        """ Altas """
        print("")
        opcion = input("Deseas dar de alta un Televisor (1) o un
reproductor(2)? ")
        if opcion == "1":
            a = Televisor()
            a.captura()
            for i in self.__lista:
                if a.id == i.id:
                    print("Error identificador existe")
                    break
                else:
                    self.__lista.append(a)
        elif opcion == "2":
            b = Reproductor()
            b.captura()
            for i in self.__lista:
                if b.id == i.id:
                    print("Error identificador existe")
                    break
```

```
                else:
                    self.__lista.append(b)
    def bajas(self):
        """ Bajas"""
        print("")
        id = input("Identificador a borrar?")
        for i in self.__lista:
            if id == i.id:
                print("identificador borrado")
                print(i.identificador())
                self.__lista.remove(i)
    def cambios(self):
        """ Cambios"""
        print("")
        id = input("Identificador a cambiar?")
        for i in self.__lista:
            if id == i.id:
                i.captura()
            else:
                print("identificador no encontrado")
    def visualiza(self):
        """ Visualizar"""
        # print("Visualiza la lista de articulos desde la lista")
        # print(self.__lista) # Visualiza el objeto lista
        print("Visualiza la lista de articulos")
        for i in self.__lista:
            # i.visualizar()
            # i.visualizaAtributosJeraquia()
            print(i)
    def salir(self):
        """Salir"""
        self.__msw = 0
        with open(self.__archivo, "wb") as fp:
            pickle.dump(self.__lista, fp)
    def otro(self):
        """En caso de no ser alguna de las opciones definidas"""
        print("")
        input(
```

```
            "No has pulsado ninguna opcion correcta...\npulsa una
    tecla para continuar"
            )
```

Una vez definidos los códigos anteriores, podemos probar el programa mediante:

```
"""
Prueba de las clases
"""
if __name__ == "__main__":
    a = Prueba()
    a.menu()
```

De esta forma, tenemos una clase de Altas, Bajas y Cambios que nos permite manipular uno o múltiples ejemplos -Televisor y Reproductor- en un solo sistema computacional que es flexible, extendible y de fácil mantenimiento, además de permitir el manejo simultáneo de múltiples sistemas de ABC.

8 Entornos de Desarrollo y Herramientas de Programación

En los últimos años los lenguajes de programación han ido evolucionado en el desarrollo de sistemas o Software, con el objetivo principal de facilitar al usuario las actividades que realiza día con día; por tal motivo, como programador, es importante conocer los conceptos básicos de programación, los tipos de lenguajes que se utilizan para el desarrollo y su funcionamiento para la interpretación de algoritmos, así como para dar solución a los problemas que pudieran presentarse.

En términos generales, un lenguaje de programación es una herramienta que permite desarrollar Software o programas para computadora. Los lenguajes de programación son empleados para diseñar e implementar programas encargados de definir y administrar el comportamiento de los dispositivos físicos y lógicos de una computadora. Lo anterior se logra mediante la creación e implementación de algoritmos de precisión que se utilizan como una forma de comunicación humana con la computadora.

Los 5 lenguajes de programación más populares en la actualidad son Java, C, Python, C++, y #C según el índice de TIOBE[87] actualizado en enero de 2021, por otro lado en la lista de la IEEE a septiembre del 2021 los 10 lenguajes de programación más populares son Python, Java, C, C++, y JavaScript, C#, R, Go, HTML, Swift.

A grandes rasgos, un lenguaje de programación se conforma de una serie de símbolos y reglas de sintaxis y semántica que definen la estructura principal del lenguaje y le dan un significado a sus elementos y expresiones.

Programación es el proceso de análisis, diseño, implementación, prueba y depuración de un algoritmo, a partir de un lenguaje que compila y genera un código fuente ejecutado en la computadora.

La función principal de los lenguajes de programación es escribir programas que permiten la comunicación usuario-máquina. Unos programas especiales (compiladores o intérpretes) convierten las instrucciones escritas en código fuente, en instrucciones escritas en lenguaje máquina.

[87] Pero si vemos los cinco lenguajes más populares a septiembre de 2020 según el índice TIOBE. La lista consta de lenguajes tan conocidos como C, Java, Python, C++ y C#. Salvo Python, los otros cuatro ya estaban en el TOP 5 allá por 2015 y por 2010. Es más, si comparamos el índice de 2020 y 2019, el único cambio es entre C y Java, que pasan de ser segundo y primero a primero y segundo. El resto permanece igual.

Los intérpretes leen la instrucción línea por línea y obtienen el código máquina correspondiente.

En cuanto a los compiladores, traducen los símbolos de un lenguaje de programación a su equivalencia escrita en lenguaje máquina (proceso conocido como compilación). Por último, se obtiene un programa ejecutable.

Para programar, es necesario como mínimo contar con un editor de texto -como *vi*, *nano o micro*- y acceso al compilador o intérprete del lenguaje que nos interese. En Linux se tiene una gran variedad de lenguajes y herramientas de desarrollo -Linux fue hecho por programadores para programadores- que se pueden instalar. Pero, también están los entornos de desarrollo integrado o entorno de desarrollo interactivo -en inglés Integrated Development Environment (IDE)-, estas son aplicaciones informáticas que proporcionan servicios integrales para facilitarle al programador el desarrollo de Software.

Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse. El límite entre un IDE y otras partes del entorno de desarrollo de Software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se integran un sistema controlador de versión y varias herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de Software orientado a objetos.

Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de Software. Esto contrasta con el desarrollo de Software utilizando herramientas no relacionadas, como *vi*, GNU Compiler Collection (*gcc*) o *make*.

Uno de los propósitos de los IDE es reducir la configuración necesaria para reconstruir múltiples utilidades de desarrollo, en vez de proveer el mismo conjunto de servicios como una unidad cohesiva. Reduciendo ese tiempo de ajustes, se puede incrementar la productividad de desarrollo, en casos donde aprender a usar un IDE es más rápido que integrar manualmente todas las herramientas por separado.

Una mejor integración de todos los procesos de desarrollo hace posible mejorar la productividad en general, que únicamente ayudando con los ajustes de configuración. Por ejemplo, el código puede ser continuamente armado, mientras es editado, previendo retroalimentación instantánea, como cuando hay errores de sintaxis. Esto puede ayudar a aprender un nuevo lenguaje de programación de una manera más rápida, así como sus librerías asociadas.

Algunos IDE están dedicados específicamente a un lenguaje de programación, permitiendo que las características sean lo más cercanas al paradigma de programación de dicho lenguaje. Por otro lado, existen muchos IDE de múltiples lenguajes tales como *Eclipse, ActiveState Komodo, IntelliJ IDEA, MyEclipse, Oracle JDeveloper, NetBeans, Codenvy y Microsoft Visual Studio.* Por otro lado *Xcode, Xojo y Delphi* están dedicados a un lenguaje cerrado o a un tipo de lenguajes de programación.

Los IDE ofrecen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como *C++, Python, Java, C#, Delphi, Visual Basic,* etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es deseable que un IDE cuente con las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework populares
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda

- Componentes
- Editor de texto
- Compilador.
- Intérprete
- Herramientas de automatización
- Depurador
- Posibilidad de ofrecer un sistema de control de versiones
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuarios

Algunos de los más usados son: *Eclipse, Aptana, NetBeans, Sublime Text, Geany, Visual Studio, Brackets, Monodevelop, Komodo, Anjuta, CodeLite, Code::Blocks, PyDev, Eric, PyCharm, PTK, Spyder, Bluefish, Glade, Kdevelop, Emacs, QtCreator, Android SDK, WxFormBuilder,* etc.

8.1 Java

Java (véase [7]) es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones «escriban el programa una vez y lo ejecuten en cualquier dispositivo (Write Once, Run Anywhere» o WORA)», lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para ejecutarse en otra.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (constituida en 1982 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales y librerías de clases en 1991, y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento de las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU (véase [15]). Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

Orientado a Objetos La primera característica, orientado a objetos (OO), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el Software de forma que los distintos tipos de datos que usen, estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema Software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos.

Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del Software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software.

La reutilización del Software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de código abierto quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

Independencia de la Plataforma La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java

pueden ejecutarse igualmente en cualquier tipo de Hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run anywhere".

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode" (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su Hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por «compilación al vuelo JIT (Just In Time)».

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de compiladores sólo puede ejecutarse en un tipo de arquitectura.

La licencia sobre Java de Sun insiste en que todas las implementaciones sean "compatibles". Esto dio lugar a una disputa legal entre Microsoft y Sun, cuando este último alegó que la implementación de Microsoft no daba soporte a las interfaces RMI y JNI además de haber añadido características "dependientes" de su plataforma. Sun demandó a Microsoft y ganó por daños y perjuicios (unos 20 millones de dólares), así como una orden judicial forzando el acatamiento de la licencia de Sun. Como respuesta, Microsoft no ofrece Java con su versión de sistema operativo, y en recientes versiones de Windows, su navegador Internet Explorer no admite la ejecución de applets sin un «conector (Plugin)» aparte. Sin embargo, Sun y otras fuentes ofrecen versiones gratuitas para distintas versiones de Windows.

Las primeras implementaciones del lenguaje usaban una máquina virtual interpretada para conseguir la portabilidad. Sin embargo, el resultado eran programas que se ejecutaban comparativamente más lentos que aquellos escritos en C o C++. Esto hizo que Java se ganase una reputación de lento en rendimiento. Las implementaciones recientes de la JVM dan lugar a programas que se ejecutan considerablemente más rápido que las versiones antiguas,

empleando diversas técnicas, aunque sigue siendo mucho más lentos que otros lenguajes.

La primera de estas técnicas es simplemente compilar directamente en código nativo como hacen los compiladores tradicionales, eliminando la etapa del bytecode. Esto da lugar a un gran rendimiento en la ejecución, pero tapa el camino a la portabilidad. Otra técnica, conocida como «compilación al vuelo JIT (Just In Time)», convierte el bytecode a código nativo cuando se ejecuta la aplicación. Otras máquinas virtuales más sofisticadas usan una "recompilación dinámica" en la que la VM es capaz de analizar el comportamiento del programa en ejecución y recompila y optimiza las partes críticas. La recompilación dinámica puede lograr mayor grado de optimización que la compilación tradicional (o estática), ya que puede basar su trabajo en el conocimiento que de primera mano tiene sobre el entorno de ejecución y el conjunto de clases cargadas en memoria. La compilación JIT y la recompilación dinámica permiten a los programas Java aprovechar la velocidad de ejecución del código nativo sin por ello perder la ventaja de la portabilidad en ambos.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en múltiples plataformas de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run anywhere" cómo "Write once, debug everywhere" (o "Escríbelo una vez, ejecútalo en cualquier parte" por "Escríbelo una vez, depúralo en todas partes").

El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

El Recolector de Basura En Java el problema de las fugas de memoria se evita en gran medida gracias a la «recolección automática de basura (o automatic garbage collector)». El programador determina cuándo se crean los objetos y el entorno en «tiempo de ejecución de Java (Java runtime)» es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos, pueden tener localizado un objeto mediante una referencia a este. Cuando no quedan referencias a un objeto, el recolector de basura de Java

borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de este; al salir del método el objeto es eliminado). Aun así, es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios; es decir, pueden aún ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos y mayor seguridad.

Instalación de Java e IDEs Existen diversas versiones de Java para Linux, la más usada es JDK de Oracle pero también está una versión abierta llamada OpenJDK, para instalar por ejemplo OpenJDK 17 en Debian GNU/Linux es necesario hacer:

```
# apt install default-jdk
```

o

```
# apt install openjdk-17-jre openjdk-17-jdk openjdk-17-doc
```

si se desea instalar solo el Run-Time JRE, para ello usamos:

```
# apt install default-jre
```

o

```
# apt install openjdk-17-jre
```

y si hay más de una versión instalada, podemos actualizar la versión por omisión de Java:

```
# update-java-alternatives -s java-1.17.0-openjdk-amd64
```

para conocer la versión instalada usamos:

```
$ java –version
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Java, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install eclipse eclipse-cdt eclipse-pydev netbeans \
bluefish bluefish-plugins codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Además, es posible instalar varios editores especializados de las páginas oficiales de sus proyectos:

https://netbeans.apache.org/download/index.html
https://www.eclipse.org/downloads/
http://brackets.io/
https://www.jetbrains.com/idea/download/#section=Linux
https://www.oracle.com/tools/downloads/Jdeveloper-12c-downloads.html
http://www.drjava.org/
https://www.jgrasp.org/
https://www.bluej.org/
http://www.jcreator.com/index.htm
https://codenvy.com/
https://atom.io/
https://www.sublimetext.com/

Compilar y Ejecutar Para compilar el archivo ejemplo.java usamos *javac* en línea de comandos mediante:

```
$ javac ejemplo.java
```

y lo ejecutamos con:

```
$ java ejemplo
```

Monitoreo de Aplicaciones Java Java Development Kit (JDK) provee binarios, herramientas y compiladores para el desarrollo de aplicaciones en Java. Además incluye una poderosa herramienta de monitoreo que es *jconsole*. Podemos ejecutarla en una terminal de Linux usando:

```
$ jconsole
```

al lanzar la aplicación, nos preguntará si deseamos hacer el monitoreo local o de algún equipo conectado en red. Si seleccionamos local, entonces podemos ver el consumo de las aplicaciones que corren en la máquina virtual de Java en tiempo real de CPU, memoria, Threads, Clases, etc.

Además podemos usar el comando *jps* (Java Virtual Machine Process Status) que nos permite conocer cada proceso que corre en la máquina virtual de Java. El uso básico es:

```
$ jps
```

pero podemos pedirle más información usando:

```
$ jps -v
```

esta última nos proporciona el indicador de proceso y el nombre de la clase o archivo *Jar* que se detecte en cada instancia.

Crear y Ejecutar Archivos .jar Un archivo *.jar* (Java ARchive) es un formato de archivo independiente de la plataforma que se utiliza para agregar muchos archivos de clase Java, metadatos y recursos asociados, como texto, imágenes, etc., en un solo archivo para su distribución.

Permite que los tiempos de ejecución de Java implementen de manera eficiente una aplicación completa en un archivo de almacenamiento y brinda muchos beneficios, como seguridad, sus elementos pueden comprimirse, acortar los tiempos de descarga, permite el sellado y control de versiones de paquetes, admite la portabilidad. También es compatible con el empaquetado para extensiones.

Para crear y ejecutar archivos .jar necesitamos hacer lo siguiente:

1. Primero comencemos escribiendo una clase Java simple con un método principal para una aplicación llamada MiApp, con fines de demostración.

   ```
   $ nano MiApp.java
   ```

   Copie y pegue el siguiente código en el archivo MiApp.java.

   ```
   public class MiApp {
       public static void main(String[] args){
           System.out.println("Solo ejecuta MiApp");
       }
   }
   ```

   Grabe el archivo y cierre este.

2 A continuación, necesitamos compilar y empaquetar la clase en un archivo JAR usando las utilidades javac y jar como se muestra:

   ```
   $ javac -d . MiApp.java
   $ ls
   $ jar cvf MiApp.jar MiApp.class
   $ ls
   ```

3 Una vez creado MiApp.jar, ahora podemos ejecutar el archivo usando el comando java como se muestra:

   ```
   $ java -jar MiApp.jar
   no main manifest attribute, in MiApp.jar
   ```

De la salida del comando anterior, encontramos un error. La JVM (Java Virtual Machine) no pudo encontrar nuestro atributo de manifiesto principal, por lo que no pudo ubicar la clase principal que contiene el método principal (public static void main (String [] args)).

El archivo JAR debe tener un manifiesto que contenga una línea con el formato Main-Class: classname que defina la clase con el método principal que sirve como punto de partida de nuestra aplicación.

4 Para corregir el error anterior, necesitaremos actualizar el archivo JAR para incluir un atributo de manifiesto junto con nuestro código. Creemos un archivo MANIFEST.MF:

```
$ nano MANIFEST.MF
```

Copie y pegue la siguiente línea en el archivo MANIFEST.MF:

```
Main-Class: MiApp
```

Guarde el archivo y agreguemos el archivo MANIFEST.MF a nuestro MiApp.jar usando el siguiente comando:

```
$ jar cvmf MANIFEST.MF MiApp.jar MiApp.class
```

5 Finalmente, cuando ejecutamos el archivo JAR nuevamente, debería producir el resultado esperado como se muestra en la salida:

```
$ java -jar MiApp.jar
Solo ejecuta MiApp
```

Para obtener más información, debemos consultar las páginas de manual de los comandos java, javac y jar.

```
$ man java
$ man javac
$ man jar
```

Aprender a Programar en Java En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a programar cada uno de los distintos aspectos de Java, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Lenguaje Java

8.2 Python

Python (véase [8]) es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores (véase apéndice 12.2).

Python fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba. Van Rossum es el principal autor de Python, y su continuo rol central en decidir la dirección de Python es reconocido, refiriéndose a él como Benevolente Dictador Vitalicio (en inglés: Benevolent Dictator for Life, BDFL).

Características y paradigmas Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en *C* o *C++*. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

El intérprete de Python estándar incluye un modo interactivo en el cual se escriben las instrucciones en una especie de intérprete de comandos: las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados.

Existen otros programas, tales como IDLE, bpython o IPython, que añaden funcionalidades extra al modo interactivo, como el autocompletado de código y el coloreado de la sintaxis del lenguaje.

Elementos del lenguaje Python fue diseñado para ser leído con facilidad. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos: !, || y &&, en Python se escriben; not, or y and, respectivamente. Curiosamente el lenguaje Pascal es junto con COBOL uno de los lenguajes con muy clara sintaxis y ambos son de la década de los 70. La idea del código claro y legible no es algo nuevo.

El contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores, conocidos como indentación, antes de cada línea de órdenes pertenecientes al bloque. Python se diferencia así de otros lenguajes de programación que mantienen como costumbre declarar los bloques mediante un conjunto de caracteres, normalmente entre llaves {}. Se pueden utilizar tanto espacios como tabuladores para indentar el código, pero se recomienda no mezclarlos.

Debido al significado sintáctico de la indentación, cada instrucción debe estar contenida en una sola línea. No obstante, si por legibilidad se quiere dividir la instrucción en varias líneas, añadiendo una barra invertida: \ al final de una línea, se indica que la instrucción continúa en la siguiente.

Variables Las variables se definen de forma dinámica, lo que significa que no se tiene que especificar cuál es su tipo de antemano y puede tomar distintos valores en otro momento, incluso de un tipo diferente al que tenía previamente. Se usa el símbolo = para asignar valores.

Módulos Existen muchas propiedades que se pueden agregar al lenguaje importando módulos, que son "minicódigos" (la mayoría escritos también en Python) que proveen de ciertas funciones y clases para realizar determinadas tareas. Un ejemplo es el módulo: Tkinter, que permite crear interfaces gráficas basadas en la biblioteca Tk. Otro ejemplo es el módulo: os, que provee acceso a muchas funciones del sistema operativo. Los módulos se agregan a los códigos escribiendo la palabra reservada *import* seguida del nombre del módulo que queramos usar.

Python tiene una gran biblioteca estándar, usada para una diversidad de tareas. Esto viene de la filosofía "pilas incluídas" ("batteries included") en referencia a los módulos de Python[88]. Los módulos de la biblioteca estándar pueden mejorarse por módulos personalizados escritos tanto en *C* como en Python. Debido a la gran variedad de herramientas incluidas en la biblioteca estándar, combinada con la habilidad de usar lenguajes de bajo nivel como *C* y *C++*, los cuales son capaces de interactuar con otras bibliotecas, Python es un lenguaje que combina su clara sintaxis con el inmenso poder de lenguajes menos elegantes.

Algunos Módulos para Python

TensorFlow Models sirve para el aprendizaje automático y aprendizaje profundo. TensorFlow Models es el repositorio de fuente abierta para encontrar muchas bibliotecas y modelos relacionados con el aprendizaje profundo.

Keras es una API de redes neuronales de alto nivel, escrita en Python y es capaz de ejecutarse sobre TensorFlow, CNTK o Theano. Fue desarrollado con un enfoque para permitir la experimentación rápida.

[88] Una lista de módulos disponibles en Python está en su página oficial.
Para la versión 2 en: https://docs.python.org/2/py-modindex.html
Para la versión 3 en: https://docs.python.org/3/py-modindex.html

Frasco es un framework ligero de aplicaciones Web WSGI. está diseñado para que el inicio sea rápido y fácil, con la capacidad de escalar hasta aplicaciones complejas. Comenzó como un simple envoltorio alrededor de Werkzeug y Jinja y se ha convertido en uno de los frameworks de aplicación Web Python más populares.

Scikit-learn es un módulo de Python para el aprendizaje automático construido sobre SciPy y distribuido bajo la licencia BSD.

Zulip es una poderosa aplicación de chat grupal de código abierto que combina la inmediatez del chat en tiempo real con los beneficios de productividad de las conversaciones enhebradas. Zulip es utilizado por proyectos de código abierto, compañías de Fortune 500, cuerpos de grandes estándares y otros que necesitan un sistema de chat en tiempo real que les permita a los usuarios procesar fácilmente cientos o miles de mensajes al día. Con más de 300 colaboradores que fusionan más de 500 commits por mes, Zulip es también el proyecto de chat grupal de código abierto más grande y de más rápido crecimiento.

Django es un framework Web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático de desarrollo Web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo-vista-template. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kanzas y fue liberada al público bajo una licencia BSD en julio del 2005.

Rebound es una herramienta de línea de comandos que obtiene instantáneamente los resultados de desbordamiento de pila cuando se produce un error de compilación.

Google Images Download Este es un programa de línea de comando de Python para buscar palabras clave / frases clave en Google Imágenes y opcionalmente descargar imágenes a su computadora. También puede invocar este script desde otro archivo Python.

YouTube-dl es usado para descargar videos de: youtube.com u otras plataformas de video.

System Design Primer este repositorio es una colección organizada de recursos para ayudar a aprender a construir sistemas a escala.

Mask R-CNN es para detección y segmentación de objetos. Esta es una implementación de Mask R-CNN en Python 3, Keras y TensorFlow. El modelo genera cuadros de delimitación y máscaras de segmentación para cada instancia de un objeto en la imagen. Se basa en Feature Pyramid Network (FPN) y ResNet101 backbone.

Face Recognition es usado para reconocer y manipular caras desde Python o desde la línea de comandos con la biblioteca de reconocimiento facial más simple del mundo. Esto también proporciona una herramienta de línea de comandos: face_recognition simple que permite hacer reconocimiento de rostros en una carpeta de imágenes desde la línea de comandos.

Snallygaster Herramienta para buscar archivos secretos en servidores HTTP.

Ansible es un sistema de automatización de TI radicalmente simple. Maneja la administración de configuraciones, la implementación de aplicaciones, el aprovisionamiento en la nube, la ejecución de tareas ad-hoc y la orquestación multinodo, incluida la trivialización de cosas como actualizaciones continuas de tiempo de inactividad cero con balanceadores de carga.

Detectron es el sistema de software de Facebook AI Research que implementa algoritmos de detección de objetos de última generación, incluyendo Mask R-CNN, está escrito en Python y funciona con el marco de aprendizaje profundo Caffe2.

Asciinema registrador de sesión de terminal y el mejor compañero de asciinema.org.

HTTPie es un cliente HTTP de línea de comando. Su objetivo es hacer que la interacción de la CLI con los servicios Web sea lo más amigable posible para los humanos. Proporciona un comando http simple que permite el envío de solicitudes HTTP arbitrarias utilizando una sintaxis simple y natural, y

muestra una salida coloreada. HTTPie se puede usar para probar, depurar y, en general, interactuar con servidores HTTP.

You-Get es una pequeña utilidad de línea de comandos para descargar contenidos multimedia (videos, audios, imágenes) desde la Web, en caso de que no haya otra forma práctica de hacerlo.

Sentry es un servicio que ayuda a controlar y corregir fallas en tiempo real. El servidor está en Python, pero contiene una API completa para enviar eventos desde cualquier lenguaje, en cualquier aplicación.

Tornado es un framework Web de Python y una biblioteca de red asíncrona, desarrollada originalmente en FriendFeed. Mediante el uso de E/S de red sin bloqueo, Tornado puede escalar a decenas de miles de conexiones abiertas, lo hace ideal para largos sondeos, WebSockets y otras aplicaciones que requieren una conexión de larga duración para cada usuario.

Magenta es un proyecto de investigación que explora el papel del aprendizaje automático en el proceso de creación de arte y música. Principalmente, esto implica desarrollar nuevos algoritmos de aprendizaje profundo y aprendizaje de refuerzo para generar canciones, imágenes, dibujos y otros materiales. Pero también es una exploración en la construcción de herramientas e interfaces inteligentes que permiten a artistas y músicos ampliar sus procesos utilizando estos modelos.

ZeroNet crea sitios Web descentralizados utilizando Bitcoin Crypto y la red BitTorrent.

Gym OpenAI Gym es un conjunto de herramientas para desarrollar y comparar algoritmos de aprendizaje de refuerzo. Esta es la biblioteca de código abierto de Gym, que le da acceso a un conjunto estandarizado de entornos.

Pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para que trabajar con datos "relacionales" o "etiquetados" sea fácil e intuitivo. Su objetivo es ser el componente fundamental de alto nivel para hacer un análisis práctico y real de

datos en Python. Además, tiene el objetivo más amplio de convertirse en la herramienta de análisis / manipulación de datos de código abierto más potente y flexible disponible en cualquier lenguaje.

Luigi es un paquete de Python que te ayuda a construir tuberías complejas de trabajos por lotes. Maneja la resolución de dependencia, la administración del flujo de trabajo, la visualización, el manejo de fallas, la integración de línea de comando y mucho más.

SpaCy (by Explosion AI) es una biblioteca para el procesamiento avanzado del lenguaje natural en Python y Cython, está basado en las últimas investigaciones y fue diseñado desde el primer día para ser utilizado en productos reales. SpaCy viene con modelos estadísticos precompilados y vectores de palabras, y actualmente admite tokenización para más de 20 lenguajes. Cuenta con el analizador sintáctico más rápido del mundo, modelos de redes neuronales convolucionales para etiquetado, análisis y reconocimiento de una entidad nombrada y fácil integración de aprendizaje profundo.

Theano es una biblioteca de Python que permite definir, optimizar y evaluar expresiones matemáticas que involucran matrices multidimensionales de manera eficiente. Puede usar GPU y realizar una diferenciación simbólica eficiente.

TFlearn es una biblioteca de aprendizaje profundo modular y transparente construida sobre Tensorflow. Fue diseñada para proporcionar una API de nivel superior a TensorFlow con el fin de facilitar y agilizar la experimentación, sin dejar de ser totalmente transparente y compatible con ella.

Kivy es un framework Python de código abierto y plataforma para el desarrollo de aplicaciones que hacen uso de interfaces de usuario innovadoras y multitáctiles. El objetivo es permitir un diseño de interacción rápido y fácil y un prototipado rápido a la vez que hace que su código sea reutilizable.

Mailpile es un cliente de correo electrónico moderno y rápido con características de cifrado y privacidad fáciles de usar. El desarrollo de Mailpile esta financiado por una gran comunidad de patrocinadores y todo el código

relacionado con el proyecto es y será lanzado bajo una licencia de Software Libre aprobada por OSI.

Matplotlib es una biblioteca de trazado 2D de Python que produce figuras con calidad de publicación en una variedad de formatos impresos y entornos interactivos en todas las plataformas. Matplotlib se puede utilizar en scripts Python, el shell Python e IPython, así como en servidores de aplicaciones Web y varios toolkits de interfaz gráfica de usuario.

YAPF (by Google) toma el código y lo reformatea con el mejor formato que cumpla con la guía de estilo, incluso si el código original no viola la guía de estilo.

Cookiecutter una utilidad de línea de comandos que crea proyectos desde cookiecutters (plantillas de proyecto), por ejemplo creando un proyecto de paquete Python a partir de una plantilla de proyecto de paquete Python.

HTTP Prompt es un cliente HTTP interactivo de línea de comandos con autocompletado y resaltado de sintaxis, basado en prompt_toolkit y HTTPie.

Speedtest-cli interfaz de línea de comandos para probar el ancho de banda de Internet con speedtest.net: http://www.speedtest.net/

Pattern es un módulo de minería Web para Python. Tiene herramientas para Minería de datos, Procesamiento de lenguaje natural, Aprendizaje automático y Análisis de red.

Gooey (Beta) convierte (casi) cualquier programa de consola Python 2 o 3 en una aplicación GUI con una línea.

Wagtail CMS es un sistema de gestión de contenido creado en Django. Se centra en la experiencia del usuario y ofrece un control preciso para diseñadores y desarrolladores.

Bottle es un micro-Framework WSGI rápido, simple y liviano para Python. Se distribuye como un módulo de archivo único y no tiene dependencias distintas de la biblioteca estándar de Python.

Prophet (by Facebook) es un procedimiento para pronosticar datos de series temporales. Se basa en un modelo aditivo en el que las tendencias no lineales se ajustan a la estacionalidad anual y semanal, más las vacaciones. Funciona mejor con datos de periodicidad diaria con al menos un año de datos históricos. Prophet es robusto para datos faltantes, cambios en la tendencia y grandes valores atípicos.

Falcon es un marco Web de Python confiable y de alto rendimiento para construir Backend de aplicaciones a gran escala y microservicios. Fomenta el estilo arquitectónico REST e intenta hacer lo mínimo posible sin dejar de ser altamente efectivo.

Mopidy es un servidor de música extensible escrito en Python. Mopidy reproduce música desde el disco local, Spotify, SoundCloud, Google Play Music y más. Edita la lista de reproducción desde cualquier teléfono, tableta o computadora usando una gama de clientes MPD y Web.

Hug tiene como objetivo hacer que el desarrollar APIs impulsadas por Python sea lo más simple posible, pero no más simple. Como resultado, simplifica drásticamente el desarrollo de la API de Python.

SymPy es una biblioteca de Python para matemática simbólica.

Visdom es una herramienta flexible para crear, organizar y compartir visualizaciones de datos vivos y enriquecidos. Admite Torch y Numpy.

Pygame es una biblioteca de plataforma cruzada diseñada para facilitar la escritura de software multimedia, como juegos en Python.

Requests es una biblioteca de Python que le permite enviar solicitudes HTTP / 1.1, agregar encabezados, datos de formularios, archivos multiparte y parámetros con simples diccionarios de Python. También le permite acceder a los datos de respuesta de la misma manera.

Statsmodels es un paquete de Python que proporciona un complemento para Scipy para cálculos estadísticos que incluyen estadística descriptiva y estimación e inferencia para modelos estadísticos.

Scrapy es ampliamente utilizada en la biblioteca de raspado Web de Python. Se usa para crear programas de rastreo. Inicialmente, fue diseñado para raspar, como su nombre indica, pero ahora se usa para muchos propósitos, incluida la extracción de datos, las pruebas automatizadas, etc. Scrapy es de código abierto.

PyTorch es una biblioteca de código abierto, básicamente es un reemplazo de la biblioteca Numpy y está equipada con funcionalidades de nivel superior para construir redes neuronales profundas. Se puede usar otro lenguaje como Scipy, Cython y Numpy, que ayudan a extender PyTorch cuando sea necesario. Muchas organizaciones, incluyendo Facebook, Twitter, Nvidia, Uber y otras organizaciones usan Pytorch para la creación rápida de prototipos en investigación y para entrenar modelos de aprendizaje profundo.

Requests es una de las famosas bibliotecas de Python que tiene licencia bajo Apache2 y esta escrita en Python. Esta biblioteca ayuda a los humanos a interactuar con los lenguajes. Con la biblioteca de solicitudes, no es necesario que agregue consultas, cadenas manualmente a las URL ni codificar los datos POST. Se puede enviar solicitudes HTTP al servidor mediante la biblioteca de solicitudes y se puede agregar datos de formularios, contenido como encabezado, archivos en varias partes, etc.

PyFlux es una biblioteca de Python que se usa para predecir y analizar series temporales, está desarrollado por Ross Taylor, esta biblioteca tiene muchas opciones para la interfaz y contiene muchas clases nuevas de tipos de modelos. Pyflux permite a los usuarios implementar muchos modelos modernos de series de tiempo como GARCH y predecir la naturaleza de cómo reaccionará en el futuro.

Zappa es uno de los mejores paquetes de Python creados por Miserlou, es tan fácil de construir e implementar aplicaciones sin servidor en API Gateway y Amazon Web Services Lambda. Dado que AWS maneja la escala horizontal de forma automática, por lo que no habrá tiempo de espera

de solicitud. Con Zappa, puede actualizar su código en una sola línea con Zappa.

Arrow es una famosa biblioteca de Python amigable para los humanos que ofrece funciones sensatas como crear, formatear, manipular y convertir fechas, horas y marcas de tiempo. Es compatible con Python 2 y 3 y es una alternativa de fecha y hora, ofrece funciones completas con una interfaz más agradable.

Pendulum es un paquete de Python que se utiliza para manipular fechas y horas, el código seguirá funcionando si se reemplazan todos los elementos de DateTime. Con Pendulum, se puede analizar DateTime y mostrar la fecha y hora con la zona horaria. Básicamente, Pendulum es una versión mejorada de la biblioteca Arrow y tiene todos los métodos útiles como redondear, truncar, convertir, analizar, formatear y aritmética.

Theano es una biblioteca de aprendizaje profundo de Python, que se utiliza para optimizar, definir y evaluar ecuaciones numéricas matemáticas y matrices multidimensionales, está desarrollado por el grupo de aprendizaje automático, por lo que, básicamente, Theano es un compilador de expresión matemática y proporciona una estrecha integración con Numpy y proporciona una optimización rápida y estable.

IPython esta es una de las herramientas de Python más útiles, ya que proporciona una rica arquitectura para el usuario. Esta herramienta permite escribir y ejecutar el código Python en el navegador. Ipython funciona en varios sistemas operativos, incluidos Windows, Mac OS X, Linux y la mayoría de los sistemas operativos Unix. IPython brinda todas las características que obtendrá en el intérprete básico con algunas características adicionales como números, más funciones, funciones de ayuda, edición avanzada, etc.

Imbalanced-learn en un mundo ideal, tendríamos conjuntos de datos perfectamente equilibrados y todos entrenaríamos modelos y seríamos felices. Desafortunadamente, el mundo real no es así, y ciertas tareas favorecen datos muy desequilibrados. Por ejemplo, al predecir el fraude en las transacciones de tarjetas de crédito, es de esperar que la gran mayoría de las transacciones

(+ 99.9%) sean realmente legítimas. El entrenamiento ingenuo de algoritmos de ML conducirá a un rendimiento deprimente, por lo que se necesita cuidado adicional al trabajar con estos tipos de conjuntos de datos. Afortunadamente, este es un problema de investigación estudiado y existe una variedad de técnicas. Imbalanced-learn es un paquete de Python que ofrece implementaciones de algunas de esas técnicas, para hacer la vida mucho más fácil. Es compatible con Scikit-learn y es parte de los proyectos Scikit-learn-contrib.

Caffe2 el marco original de Caffe ha sido ampliamente utilizado durante años, y es conocido por su rendimiento incomparable y base de código probado en batalla. Sin embargo, las tendencias recientes en DL hicieron que el marco se estancara en algunas direcciones. Caffe2 es el intento de llevar Caffe al mundo moderno. Admite formación distribuida, implementación (incluso en plataformas móviles), las CPU más nuevas y Hardware compatible con CUDA. Si bien PyTorch puede ser mejor para la investigación, Caffe2 es adecuado para despliegues a gran escala como se ve en Facebook.

Dash es una biblioteca de código abierto para crear aplicaciones Web, especialmente aquellas que hacen un buen uso de la visualización de datos, en Python puro. esta construido sobre Flask, Plotly.js y React, y proporciona abstracciones que te liberan de tener que aprender esos Frameworks y permitirte ser productivo rápidamente. Las aplicaciones se representan en el navegador y responderán para que se puedan usar en dispositivos móviles. No se requiere JavaScript.

Fire es una biblioteca de código abierto que puede generar automáticamente una CLI para cualquier proyecto de Python. La clave aquí es automática: ¡casi no es necesario escribir ningún código o docstrings para construir una CLI!. Para hacer el trabajo, solo se tiene que llamar a un método Fire y pasarlo como se quiera para convertirlo en una CLI: una función, un objeto, una clase, un diccionario, o incluso no pasar ningún tipo de argumento (lo que convertirá todo el código en una CLI).

Flashtext es una biblioteca para búsqueda y reemplazo de palabras en un documento. La belleza de FlashText es que el tiempo de ejecución es el mismo sin importar cuántos términos de búsqueda se tenga, en contraste

con la expresión regular en la que el tiempo de ejecución aumentará casi linealmente con el número de términos.

Pipenv con Pipenv, se especifican todas las dependencias en un Pipfile, que normalmente se genera mediante el uso de comandos para agregar, eliminar o actualizar dependencias. La herramienta puede generar un archivo Pipfile.lock, lo que permite que las compilaciones sean deterministas, ayudando a evitar esos errores difíciles de detectar debido a una dependencia poco clara que ni siquiera se cree que es necesaria.

Luminoth las imágenes están en todas partes hoy en día y comprender su contenido puede ser crítico para varias aplicaciones. Afortunadamente, las técnicas de procesamiento de imágenes han avanzado mucho, impulsadas por los avances en DL. Luminoth es un kit de herramientas de código abierto Python para visión artificial, construido con TensorFlow y Sonnet. Actualmente, viene de fábrica y es compatible con la detección de objetos en forma de un modelo llamado Faster R-CNN.

Instalación de Python e IDEs Para instalar Python 3 en Debian GNU/Linux es necesario hacer:

```
# apt install ipython3 python3 idle3 python3-pip \
python3-matplotlib python3-rpy2 python3-numpy spyder3 \
python3-scipy bpython3 python3-pandas python-sklearn \
python-sklearn-docspe python-wxgtk3.0 jython xonsh \
python3-mpi4pypython3-pyqt5 python3-pyqtgraph mypy \
python-wxgtk3.0-dev python3-numba pyflakes3
# apt install flake8 black
```

Para instalar Jupyter (entorno de trabajo orientado a científicos que soporta los lenguajes R y Python):

```
# apt install jupyter-console jupyter-notebook
# pip3 install jupyter
# pip3 install matplotlib
# pip3 install ipywidgets
# jupyter nbextension enable –py –sys-prefix \
widgetsnbextension
```

y podemos instalar PYREPOT usando:

```
# pip install pyreport
```

además podemos instalar editores especializados en Python usando:

```
# apt install eric pyzo pyzo-doc thonny
```

otras opciones se pueden descargar de:

https://www.jetbrains.com/pycharm/
http://www.pydev.org/
https://wingware.com/

También, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Python, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text moreutils
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Compilar y Ejecutar Para compilar el archivo ejemplo.py en *python3* en línea de comandos:

- para compilarlo y ejecutarlo en *python3* en línea de comandos usamos:

  $ python3 ejemplo.py

- en caso de necesitar depurar de forma interactiva un código, podemos usar el módulo *pdb*, por ejemplo:

  $ pdb ejemplo.py

- en caso de necesitar hacer una compilación estática podemos usar:

  $ pyflakes ejemplo.py

Codificar Según Guía de Estilo PEP8 Python cuenta con herramientas que permiten al programador conocer en que discrepa su código de la guía de estilo de Python PEP8 y en caso de que lo requiera, se puede usar estas para que reformatee nuestro código siguiendo al PEP8.

- en caso querer revisar las discrepancias de nuestro código con respecto a la guía oficial de estilo PEP8 de Python usamos:

  $ flake8 ejemplo.py

- en caso de querer conocer qué cambios se sugiere hacer al código según la guía oficial de estilo PEP8 de Python usamos:

  $ black –check ejemplo.py
  $ black –diff ejemplo.py

- en caso de querer reformatear el código según la guía oficial de estilo PEP8 de Python usamos:

  $ black ejemplo.py
  $ black *.py

Anaconda Por otro lado existe Anaconda, una Suite de código abierto que abarca una serie de aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos con Python. En líneas generales Anaconda Distribu-tion es una distribución de Python que funciona como un gestor de entorno, un gestor de paquetes y que posee una colección de más de 720 paquetes de código abierto. Anaconda Distribution se agrupa en 4 sectores o soluciones tecnológicas; *Anaconda Navigator, Anaconda Project, las librerías de Ciencia de Datos y Conda*. Todas estas se instalan de manera automática y en un procedimiento muy sencillo.

Para más información ver: https://www.anaconda.com/.

También está SageMath, una Suite de código abierto bajo la licencia GPL de Software matemático como: N*umPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R*, entre otros. Además combina acceso a una poderosa combinación del lenguaje basada en Python o directamente vía interfaces o *Wrappers*. La misión del proyecto es crear una alternativa de Software libre a *Magma, Maple, Mathematica y Matlab*.

Para más información ver: http://www.sagemath.org/.

Instalación de Aplicaciones Usando Pip Pip es un sistema de administración de paquetes que se utiliza para instalar y administrar paquetes de Software escritos en Python. Pip se usa principalmente para instalar paquetes disponibles en Python Package Index (PyPI, página del proyecto: https://pypi.org). Los desarrolladores también pueden usar Pip para instalar módulos y paquetes desarrollados localmente.

Para instalar Pip en Python 2 hacemos:

```
# apt install python-pip
```

y para instalar alguna aplicación para todos los usuarios, por ejemplo *ratarmount*, usamos:

```
# pip2 install ratarmount
```

y para instalar alguna aplicación para el usuario, por ejemplo *ratarmount*, usamos:

```
$ pip2 install –user ratarmount
```

Para instalar Pip en Python 3 hacemos:

```
# apt install python3-venv python3-pip
```

y para instalar alguna aplicación para todos los usuarios, por ejemplo *ratarmount*, usamos:

```
# pip3 install ratarmount
```

y para instalar alguna aplicación para el usuario, por ejemplo *ratarmount*, usamos:

```
$ pip3 install –user ratarmount
```

En caso de instalación para el usuario, para usar la aplicación debemos agregar al PATH:

```
export PATH ="$PATH:/home/$USER/.local/bin"
```

Sin pérdida de generalidad (usando pip2 o pip3), podemos ver los detalles de algún paquete, usando:

```
# pip3 show nombre
```

Podemos instalar algún paquete, usando:

```
# pip3 install nombre
```

Podemos actualizar algún paquete, usando:

```
# pip3 install –upgrade nombre
```

Podemos desinstalar algún paquete, usando:

```
# pip3 uninstall nombre
```

Podemos listar los paquetes instalados, usando:

```
# pip3 list nombre
```

Podemos buscar algún paquete, usando:

```
# pip3 search nombre
```

Podemos listar los paquetes desactualizados, usando:

```
# pip3 list –outdated
```

Aprender a Programar en Python En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a programar cada uno de los distintos aspectos de Python, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Lenguaje Python

8.3 Julia

es un lenguaje de programación homoicónico, multiplataforma y multiparadigma (véase [21]) -soporta programación orientada a objetos, por procedimientos, funcional y meta además de multietapas- de tipado dinámico de alto nivel y alto desempeño para la computación genérica, técnica y científica, con una sintaxis similar a la de otros entornos de computación similares, con licencia MIT (véase apéndice 12.2).

Dispone de un compilador avanzado (JIT), mecanismos para la ejecución en paralelo y distribuida, además de una extensa biblioteca de funciones matemáticas. La biblioteca, desarrollada fundamentalmente en Julia, también contiene código desarrollado en C o Fortran, para el álgebra lineal, generación de números aleatorios, procesamiento de señales, y procesamiento de cadenas. Adicionalmente, la comunidad de desarrolladores de Julia contribuye con la creación y distribución de paquetes externos a través del gestor de paquetes integrado de Julia a un paso acelerado. IJulia es el resultado de la colaboración entre las comunidades de IPython y Julia, provee de una poderosa interfaz gráfica basada en el navegador para Julia.

Algunas características básicas son:

- El despacho múltiple: permite definir el comportamiento de las funciones a través de diversas combinaciones de tipos de argumentos
- Sistema de tipado dinámico: tipos para la documentación, la optimización y el despacho de funciones
- Buen desempeño, acercándose al de lenguajes estáticamente compilados como C
- Gestor de paquetes integrado
- Macros tipo Lisp y otras herramientas para la meta-programación

- Llamar funciones de Python: mediante el paquete PyCall
- Llamar funciones de C directamente: sin necesidad de usar envoltorios u APIs especiales
- Poderosas características de línea de comandos para gestionar otros procesos
- Diseñado para la computación paralela y distribuida
- Corutinas: hilos ligeros
- Los tipos definidos por el usuario son tan rápidos y compactos como los tipos estándar integrados.
- Generación automática de código eficiente y especializado para diferentes tipos de argumentos
- Conversiones y promociones para tipos numéricos y de otros tipos, elegantes y extensibles
- Soporte eficiente para Unicode, incluyendo UTF-8 pero sin limitarse solo a este
- Es de uso general, pero tiene todo lo necesario para hacer ciencia de datos, aprendizaje automático, ecuaciones diferenciales, resolvedores de sistemas lineales, etc.

Julia incluye una terminal interactiva, llamada REPL en donde se puede visualizar automáticamente los resultados de la ejecución del programa o segmento de código. Tanto el compilador justo a tiempo (JIT) basado en LLVM así como el diseño del lenguaje le permiten a Julia acercarse e incluso igualar a menudo el desempeño de C. Julia no le impone al usuario ningún estilo de paralelismo en particular. En vez de esto, le provee con bloques de construcción clave para la computación distribuida, logrando hacer lo suficientemente flexible el soporte de varios estilos de paralelismo y permitiendo que los usuarios añadan más.

Instalación de Julia e IDEs Existen diversas versiones de Julia para Linux, para instalar en Debian GNU/Linux es necesario hacer:

```
# apt install julia julia-doc
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Julia, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install eclipse eclipse-cdt eclipse-pydev netbeans \
bluefish bluefish-plugins codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Aprender a Programar en Julia En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a programar cada uno de los distintos aspectos de Julia, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Lenguaje Julia

8.4 C y C++

C (véase [9]) es un lenguaje de programación originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell, como evolución del lenguaje anterior B, a su vez basado en BCPL. Es un lenguaje orientado a la implementación de Sistemas operativos, concretamente Unix, Linux y el Kernel de Linux. *C* es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear Software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código *C* o acceder directamente a memoria o dispositivos periféricos.

Filosofía Uno de los objetivos de diseño del lenguaje *C* es que sólo sean necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución. Es muy posible escribir *C* a bajo nivel de abstracción; de hecho, *C* se usó como intermediario entre diferentes lenguajes.

En parte, a causa de ser relativamente de bajo nivel y tener un modesto conjunto de características, se pueden desarrollar compiladores de *C* fácilmente. En consecuencia, el lenguaje *C* está disponible en un amplio abanico de plataformas (más que cualquier otro lenguaje). Además, a pesar de su naturaleza de bajo nivel, el lenguaje se desarrolló para incentivar la programación independiente de la máquina. Un programa escrito cumpliendo los estándares e intentando que sea portátil puede compilarse en muchos computadores.

C se desarrolló originalmente (conjuntamente con el sistema operativo Unix, con el que ha estado asociado mucho tiempo) por programadores para programadores. Sin embargo, ha alcanzado una popularidad enorme, y se ha usado en contextos muy alejados de la programación de Software de sistemas, para la que se diseñó originalmente.

Propiedades Núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas. Es un lenguaje muy flexible que permite programar

con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo", entre sus principales propiedades destacan:

- Un sistema de tipos que impide operaciones sin sentido
- Usa un lenguaje de preprocesado, el preprocesador de *C*, para tareas como definir macros e incluir múltiples archivos de código fuente
- Acceso a memoria de bajo nivel mediante el uso de punteros
- Interrupciones al procesador con uniones
- Un conjunto reducido de palabras clave
- Por defecto, el paso de parámetros a una función se realiza por valor. El paso por referencia se consigue pasando explícitamente a las funciones las direcciones de memoria de dichos parámetros
- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo
- Tipos de datos agregados (struct) que permiten que datos relacionados (como un empleado, que tiene un id, un nombre y un salario) se combinen y se manipulen como un todo (en una única variable "empleado")

Carencias Aunque la lista de las características útiles de las que carece *C* es larga, éstos factores han sido importantes para su aceptación, porque escribir rápidamente nuevos compiladores para nuevas plataformas, mantiene lo que realmente hace el programa bajo el control directo del programador, y permite implementar la solución más natural para cada plataforma. Esta es la causa de que a menudo *C* sea más eficiente que otros lenguajes. Típicamente, sólo la programación cuidadosa en lenguaje ensamblador produce un código más rápido, pues da control total sobre la máquina, aunque los avances en los compiladores de *C* y la complejidad creciente de los microprocesadores modernos han reducido gradualmente esta diferencia, Algunas carencias son:

- Recolección de basura nativa, sin embargo se encuentran a tal efecto bibliotecas como la "*libgc*" desarrollada por Sun Microsystems, o el Recolector de basura de Boehm

- Soporte para programación orientada a objetos, aunque la implementación original de *C++* fue un preprocesador que traducía código fuente de *C++* a *C*. Véase también la librería *GObjec*t
- Funciones anidadas, aunque *GCC* tiene esta característica como extensión
- Soporte nativo para programación multihilo. Disponible usando librerías como libpthread

Ventajas estas se pueden resumir en:

- Lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas
- A pesar de su bajo nivel es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas de cómputo conocidos
- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes

Inconvenientes El mayor problema que presenta el lenguaje *C* frente a los lenguajes de tipo de dato dinámico es la gran diferencia en velocidad de desarrollo: es más lento programar en *C*, sobre todo para el principiante. La razón estriba en que el compilador de *C* se limita a traducir código sin apenas añadir nada. La gestión de la memoria es un ejemplo clásico: en *C* el programador ha de reservar y liberar la memoria explícitamente. En otros lenguajes (como *BASIC*, *MATLAB* o *C#*) la memoria es gestionada de forma transparente para el programador. Esto alivia la carga de trabajo humano y en muchas ocasiones evita errores, aunque también supone mayor carga de trabajo para el procesador.

El mantenimiento en algunos casos puede ser más difícil y costoso que con ciertos lenguajes de más alto nivel. El código en *C* se presta a sentencias cortas y enrevesadas de difícil interpretación.

Cabe destacar el contexto y época en la que fue desarrollado *C*. En aquellos tiempos existían muy pocos programadores, los cuales, a su vez, eran prácticamente todos expertos en el área. De esta manera, se asumía que los programadores eran conscientes de sus trabajos y capaces de manejar

perfectamente el lenguaje. Por esta razón es muy importante que los recién iniciados adopten buenas prácticas a la hora de escribir en *C* y manejar la memoria, como por ejemplo un uso intensivo de indentación y conocer a fondo todo lo que implica el manejo de punteros y direcciones de memoria.

Aplicabilidad Hecho principalmente para la fluidez de programación en sistemas UNIX. Se usa también para el desarrollo de otros sistemas operativos como Windows o GNU/Linux. Igualmente para aplicaciones de escritorio como GIMP, cuyo principal lenguaje de programación es *C*.

De la misma forma, es muy usado en aplicaciones científicas (para experimentos informáticos, físicos, químicos, matemáticos, entre otros, conocidos como modelos y simuladores), industriales (industria robótica, cibernética, sistemas de información y base de datos para la industria petrolera y petroquímica). Predominan también todo lo que se refiere a simulación de máquinas de manufactura, simulaciones de vuelo (es la más delicada, ya que se tienen que usar demasiados recursos tanto de Hardware como de Software para desarrollar aplicaciones que permitan simular el vuelo real de una aeronave). Se aplica por tanto, en diversas áreas desconocidas por gran parte de los usuarios noveles.

Los equipos de cómputo de finales de los 90 son varios órdenes de magnitud más potentes que las máquinas en que C se desarrolló originalmente. Programas escritos en lenguajes de tipo dinámico y fácil codificación (Ruby, Python, Perl, etc.) que antaño hubieran resultado demasiado lentos, son lo bastante rápidos como para desplazar en uso a *C*. Aun así, se puede seguir encontrando código *C* en grandes desarrollos de animaciones, modelados y escenas en 3D en películas y otras aplicaciones multimedia.

Actualmente, los grandes proyectos de Software se dividen en partes, dentro de un equipo de desarrollo. Aquellas partes que son más "burocráticas" o "de gestión" con los recursos del sistema, se suelen realizar en lenguajes de tipo dinámico o de guión (script), mientras que aquellas partes "críticas", por su necesidad de rapidez de ejecución, se realizan en un lenguaje de tipo compilado, como *C* o *C++*. Si después de hacer la división, las partes críticas no superan un cierto porcentaje del total (aproximadamente el 10%) entonces todo el desarrollo se realiza con lenguajes dinámicos. Si la parte crítica no llega a cumplir las expectativas del proyecto, se comparan las alternativas de una inversión en nuevo Hardware frente a invertir en el coste de un programador para que reescriba dicha parte crítica.

Ya que muchos programas han sido escritos en el lenguaje *C* existe una gran variedad de bibliotecas disponibles. Muchas bibliotecas son escritas en *C* debido a que *C* genera código objeto rápido; los programadores luego generan interfaces a la biblioteca para que las rutinas puedan ser utilizadas desde lenguajes de mayor nivel, tales como Java, Perl y Python.

C++ (véase [10]) es un lenguaje de programación diseñado a mediados de 1979 por Bjarne Stroustrup. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, *C++* es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos. Por esto se suele decir que el *C++* es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO *C++*, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad de *C++* es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre "*C++*" fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "*C* con clases". En *C++*, la expresión "*C++*" significa "incremento de *C*" y se refiere a que *C++* es una extensión de *C*.

El concepto de clase Los objetos en C++ son abstraídos mediante una clase. Según el paradigma de la programación orientada a objetos un objeto consta de:

- Identidad, que lo diferencia de otros objetos (Nombre que llevará la clase a la que pertenece dicho objeto).
- Métodos o funciones miembro
- Atributos o variables miembro

Diferencias de tipos respecto a C En C++, cualquier tipo de datos que sea declarado completo (fully qualified, en inglés) se convierte en un tipo de datos único. Las condiciones para que un tipo de datos T sea declarado completo son a grandes rasgos las siguientes:

- Es posible al momento de compilación conocer el espacio asociado al tipo de datos (es decir, el compilador debe conocer el resultado de sizeof(T))
- T Tiene al menos un constructor, y un destructor, bien declarados
- Si T es un tipo compuesto, o es una clase derivada, o es la especificación de una plantilla, o cualquier combinación de las anteriores, entonces las dos condiciones establecidas previamente deben aplicar para cada tipo de dato constituyente
- En general, esto significa que cualquier tipo de datos definido haciendo uso de las cabeceras completas, es un tipo de datos completo
- En particular, y a diferencia de lo que ocurría en *C*, los tipos definidos por medio de struct o enum son tipos completos. Como tales, ahora son sujetos a sobrecarga, conversiones implícitas, etcétera

Los tipos enumerados, entonces, ya no son simplemente alias para tipos enteros, sino que son tipos de datos únicos en *C++*. El tipo de datos bool, igualmente, también pasa a ser un tipo de datos único, mientras que en *C* funcionaba en algunos casos como un alias para alguna clase de dato de tipo entero.

Compiladores Uno de los compiladores libres de C++ es el de GNU, el compilador G++ (parte del proyecto GCC, que engloba varios compiladores para distintos lenguajes). Otros compiladores comunes son Intel C++ Compiler, el compilador de Xcode, el compilador de Borland C++, el compilador de CodeWarrior C++, el compilador g++ de Cygwin, el compilador g++ de MinGW, el compilador de Visual C++, Carbide.c++, entre otros.

Instalación de C y C++ e IDEs Existen diversas versiones de C y C++ para Linux, para instalarlos en Debian GNU/Linux es necesario hacer:

```
# apt install build-essential manpages-dev glibc-doc \
glibc-doc-reference gcc-doc-base gcc-doc splint \
c++-annotations-pdf g++ jam ohcount ctags \
cppcheck cccc autoconf automake make cmake scons
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en C y C++, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install eclipse eclipse-cdt eclipse-pydev netbeans \
bluefish bluefish-plugins codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text indent
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Compilar y Ejecutar Para compilar el archivo ejemplo.c en *C* en línea de comandos usamos:

```
$ gcc ejemplo.c -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

Para compilar el archivo ejemplo.cpp en *C++* en línea de comandos usamos:

```
$ g++ ejemplo.cpp -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

Aprender a Programar en C y C++ En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a programar cada uno de los distintos aspectos de C y C++, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Lenguajes C y C++

8.5 Fortran

Fortran (véase [**?**]) contracción del inglés The IBM Mathematical Formula Translating System, es un lenguaje de programación de alto nivel de propósito general, procedimental e imperativo, que está especialmente adaptado al cálculo numérico y a la computación científica. Desarrollado originalmente por IBM en 1957 para el equipo IBM 704, y usado para aplicaciones científicas y de ingeniería, el Fortran vino a dominar esta área de la programación desde el principio y ha estado en uso continuo por más de medio siglo en áreas de cómputo intensivo tales como la predicción numérica del tiempo, análisis de elementos finitos, dinámica de fluidos computacional, física computacional y química computacional. Es uno de los lenguajes más populares en el área de la computación de alto rendimiento y es el lenguaje usado para programas que evalúan el desempeño (benchmark) y el ranking de los supercomputadores más rápidos del mundo.

El Fortran abarca un linaje de versiones, cada una de las cuales evolucionó para añadir extensiones al lenguaje mientras que usualmente retenía compatibilidad con las versiones previas. Versiones sucesivas han añadido soporte para procesamiento de datos basados en caracteres (Fortran 77), programación de arreglos, programación modular y programación orientada a objetos (Fortran 90/95), y programación genérica (Fortran 2003/2008).

Ventajas e inconvenientes de su sintaxis como fue una primera tentativa de creación de un lenguaje de programación de alto nivel, tiene una sintaxis considerada arcaica por muchos programadores que aprenden

lenguajes más modernos. Es difícil escribir un bucle "for", y errores en la escritura de un solo carácter pueden llevar a errores durante el tiempo de ejecución en vez de errores de compilación, en el caso de que no se usen las construcciones más frecuentes. Algunas de las primeras versiones no poseían facilidades que son consideradas muy útiles, tal como la asignación dinámica de memoria.

Se debe tener en cuenta que la sintaxis de Fortran fue orientada para el uso en trabajos numéricos y científicos. Muchas de sus deficiencias han sido abordadas en revisiones recientes del lenguaje. Por ejemplo, en 1990 se presentó un tercer estándar ANSI conocido como FORTRAN 90, que contenía muchas nuevas características y permitía una programación más estructurada. Una serie de cambios menores se presentaron en 1995 conocido como FORTRAN 95 posee comandos mucho más breves para efectuar operaciones matemáticas con matrices y dispone de tipos y soporta OpenMP. Esto no solo mejora mucho la lectura del programa sino que además aporta información útil al compilador. Actualmente se tienen estándares para las versiones 2003 y 2008 y se continúa trabajando en mejoras al lenguaje.

Por estas razones Fortran prácticamente no se usa fuera de los campos científicos y del análisis numérico, pero permanece como el lenguaje preferido para desarrollar aplicaciones de computación numérica de alto rendimiento.

Características

Tipos de datos soportados:

- Numéricos (enteros, reales, complejos y de doble precisión).
- Boleanos (logical).
- Arreglos.
- Cadenas de caracteres.
- Archivos.

FORTRAN 90/95 ya es estructurado, y no requiere sentencias GOTO. Sólo admite dos ámbitos para las variables: local y global.

Variables y constantes

- Fortran no es sensible a mayúsculas y minúsculas. Los nombres de variables tienen de 6 a 31 caracteres máximo y deben comenzar por una letra. Los blancos son significativos.
- Declaración explícita de variables.
- Enteras (I-N), el resto reales. (se modifica con IMPLICIT).
- Punteros: en los primeros FORTRAN no hay punteros y todas las variables se almacenan en memoria estática. En FORTRAN 90 se declaran INTEGER, POINTER::P.
- Para memoria dinámica ALLOCATE y DEALLOCATE

Tipos de datos

- Arrays, pueden tener hasta 7 dimensiones y se guardan por columnas.
- REAL M(20),N(-5:5)
- DIMENSION I(20,20) (tipo por nomenclatura implícita)
- Cadenas de caracteres, el primer carácter es el 1, el operador // permite concatenar cadenas.
- CHARACTER S*10, T*25
- Almacenamiento de datos. Se usa COMMON para datos compartidos y EQUIVALENCE cuando almacenamos una variable con dos posibles tipos en la misma posición de memoria (como union en C). Se usa DATA para inicializar datos estáticos.
- DATA X/1.0/,Y/3.1416/,K/20/
- Tipos definidos por el usuario, con TYPE <nombre>... END TYPE <nombre>

Control de secuencia el conjunto de estructuras de control es limitado:

- Expresiones, prioridad de operadores
- Enunciados
  - ◦ Asignación, cuando se hace entre cadenas hay ajuste de tamaño con blancos o truncamiento.
  - ◦ Condicional. Permite IF ELSE IF... Para selección múltiple SELECT CASE CASE.....CASE DEFAULT.... END SELECT
  - ◦ Iteración. DO....END DO
  - ◦ Nulo, se usa solo para la etiqueta. CONTINUE.
- Control de subprogramas. CALL invoca al subprograma y RETURN devuelve un valor al programa llamante.
- Construcciones propensas a error: GOTO.

Entrada y salida

- Tipos de archivos:
  - ◦ Secuenciales
  - ◦ De acceso directo
- Comandos: READ, WRITE, PRINT, OPEN , CLASE, INQUIRE (propiedades o estado del archivo) REWIND y ENDFILE (para ubicar el puntero del fichero).
- Para el tratamiento de excepciones en las sentencias READ/WRITE se puede introducir la posición de la rutina de dicho tratamiento (ERR=90).

Subprogramas

- Hay tres tipos de subprogramas:

◦ Function, devuelven un solo valor de tipo numérico, lógico o cadena de caracteres.
◦ Subroutine, devuelve valores a través de variables no locales COMMON.
◦ Función de enunciado, permite calcular una sola expresión aritmética o lógica.
◦ FN(X,Y)=SIN(X)**2-COS(Y)**2

- Gestión de almacenamiento.

  ◦ Las variables son locales o globales (COMMON)

  ◦ Recursividad: RECURSIVE FUNCTION FACTORIAL(X)

  ◦ Parámetros de subprograma. Paso por referencia.

Abstracción y encapsulación. Evaluación del lenguaje

- La abstracción es posible mediante los subprogramas y el uso de variables COMMON, aunque su uso es propenso a errores.

- FORTRAN sigue siendo utilizado en el ámbito científico y es muy eficiente realizando cálculos.

  ◦ La estructura del programa suele ser difícil de entender.

  ◦ En FORTRAN 90/95 se incluye la recursividad y la memoria dinámica.

  ◦ Las etiquetas de las sentencias ya no son necesarias, ni el GOTO, pues se ha transformado en un lenguaje estructurado.

  ◦ El aspecto de los programas sigue siendo de procesamiento por lotes

Instalación de Fortran e IDEs Existen diversas versiones de Fortran para Linux, para instalarlos en Debian GNU/Linux es necesario hacer:

```
# apt install gfortran gfortran-doc fortran77-compiler \
fortran95-compiler fortran-compiler cfortran ftc fort77
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Fortran, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Compilar y Ejecutar Para compilar el archivo ejemplo.f77 en *Fortran 77* en línea de comandos usamos:

```
$ g77 ejemplo.f77 -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

Para compilar el archivo ejemplo.f90 en *Fortran 90/95/2003* en línea de comandos usamos:

```
$ gfortran ejemplo.f90 -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

8.6 R

El paquete R (véase [29]) es un lenguaje y entorno de programación para análisis estadístico y gráfico. Se trata de un proyecto de Software libre, resultado de la implementación GNU del premiado lenguaje S. SPSS, R y S-Plus —versión comercial de S— son, probablemente, los tres lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy populares en el campo de la investigación biomédica, la bioinformática y las matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes bibliotecas o paquetes con finalidades específicas de cálculo o gráfico.

Además, R puede integrarse con distintas bases de datos y existen bibliotecas que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python. R soporta hacer interfase con lenguajes de programación como C, C++ y Fortran.

Otra de las características de R es su capacidad gráfica, que permite generar gráficos con alta calidad. R posee su propio formato para la documentación basado en LaTeX (véase [18]). R también puede usarse como herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas específicas tales como FreeMat, GNU Octave y su equivalente comercial, MATLAB. Se ha desarrollado una interfaz RWeka para interactuar con Weka (véase [44]) que permite leer y escribir ficheros en el formato arff y enriquecer R con los algoritmos de minería de datos de dicha plataforma.

Los ambientes de desarrollo integrado para R existen como proyectos externos, como pueden ser editores —que sólo soportan la sintaxis—, los IDEs (Integrate Development Environments) y los GUI (Graphical User Interfaces) —permiten editar, ejecutar y depurar código desarrollado para R—. Hay más de 20 proyectos activos, tres de los más conocidos son Tinn-R (véase [45]), RStudio (véase [46]) y Rkward de KDE.

Instalación de R Existen diversos paquetes para *R* en Debian GNU/Linux, para instalarlos es necesario hacer:

```
# apt install r-base
# apt install 'r-cran*'
# apt install 'r-bioc*'
```

Editores nativo en Debian GNU/Linux

```
# apt install rkward rkward-data
```

y podemos generar gráficos de datos usando:

```
# apt install rlplot
```

para conocer todos los paquetes asociados, usar:

```
$ apt search ^r-cran
$ apt search ^r-bioc
```

Paquetes internos

```
R2WinBugs mcmcplots devtools Rattle e1071 FactiMineR \
Ellipse xlsx hsaur shiny
```

8.7 Herramientas de Programación

En Linux existe una gran variedad de herramientas para programación, ya que este sistema operativo fue hecho por programadores y para programadores, por ello entre las miles de herramientas, tenemos algunas que son ampliamente usadas, entre las que destacan:

Editores de Terminal

- Diakonos
- Jet
- Joe
- LE
- Mined
- Nano
- Nice Editor (NE)
- Pico
- Setedit
- Vim
- Fte

Editores Sencillos con Interfaz Gráfica

- Gedit
- SciTE
- JEdit
- NEdit
- MEdit
- KScope
- Editra
- Kate
- KWrite
- Leafpad
- Mousepad
- Anjunta
- TEA
- Pluma
- GVim
- Emacs

Editores Avanzados con Interfaz Gráfica

- Atom
- Bluefish
- BlueGriffon
- Brackets

- Geany
- Glade
- Google Web Designer
- KompoZer
- Light Table
- Notepadqq
- Scribes
- Sublime Text

Entornos de Programación Integrado (IDEs)

- Aptana
- Arduino IDE
- Android Studio
- CodeLite
- Code::Blocks
- Eclipse
- Gambas
- JetBrains Suite
- NetBeans
- Ninja-IDE
- Python IDLE
- PyDev
- Postman

- Qt Creator
- Simply Fortran
- Visual Studio Code
- Wing Python IDE
- Spyder
- PyCharm
- Jupyter
- Eric

Kit de Desarrollo de Software

- .Net Core SDK
- Android SDK
- Java JDK

Comparadores de texto y fuentes

- KDiff3
- Meld
- Diffuse
- DirDiff
- kompare
- Numdiff
- colordiff
- wdiff
- xxdiff
- tkdiff
- Ndiff

Otras Herramientas

- Alleyoop
- C2HTML
- Java2HTML
- Code2HTML
- c2html
- AutoDia
- txt2html
- html2text

Programas para control de versiones que permite desarrollo colaborativo de Software:

- Git https://git-scm.com/
- Mercurial https://www.mercurial-scm.org/
- Subversion https://subversion.apache.org/
- Perforce
- Bazaar
- CVS
- LibreSource
- Monotone
- SmartGit
- GitKraken
- Git Cola

Generadores automaticos de documentación que generan salida en PDF, HTML y XML para lenguajes como C++ y Java:

- Doxygen http://www.doxygen.org/
- JavaDoc

Formateador de código fuente para C, C++, Java y C#

- Astyle http://astyle.sourceforge.net

Lenguaje Unificado de Modelado (Unified Modeling Language)[89] forja un lenguaje de modelado visual común semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de Software complejos tanto en estructura como en comportamiento:

- UML https://www.uml.org/

Depuradores de programas

- ddd https://www.gnu.org/Software/ddd/
- gdb https://www.gnu.org/Software/gdb/
- kdbg http://www.kdbg.org/

Programas para rastrear errores en la manipulación de memoria y punteros desbordados

- Valgrind http://valgrind.org/
- DUMA http://duma.sourceforge.net/

Programas para hacer análisis de rendimiento[90]

[89] Otras opciones son: UML Diagram Generation, Code Generation, Document Generation and Reporting, Scaling, Database Schema Generation, Entity Relationship Diagrams, Data Flow Datagrams, StarUML BOUML, EclipseUML, UML Modeller, Papyrus, Nclass, PlantUML, UMLet, NetBeansIDE, Open ModelSphere, gModeler, RISE, Oracle jdeveloper, Oracle SQL Developer, Dia, Kivio, ArgoUML, Xfig, etc.

[90] Otras opciones son: Splint, cppcheck, Rough Auditing Tool for Security, C y C++ Code Counter, CppNcss, Gnocchi, CUnit, CppUnit, OProfile, Intel VTune, Nemiver, Mudflap, etc.

- gprof https://sourceware.org/binutils/docs/gprof/
- Callgrind http://valgrind.org/docs/manual/cl-manual.html
- kCachegrind http://kCachegrind.sourceforge.net/html/Home.html
- time https://www.cyberciti.biz/faq/unix-Linux-time-command-examples-usage-syntax/

En este apartado, solo tocaremos las más usadas, pero abunda la documentación de estas y otras importantes herramientas en línea de comandos (véase **??**). Iniciaremos por las de compilar[91] y depurar[92] programas compilables en C, C++, Fortran, entre otros.

8.7.1 ¿Qué es eso de ASCII, ISO-8859-1 y UTF-8?

Los tres estándares representan el esfuerzo informático por brindar un sistema de codificación que permita representar los caracteres que se usan en todos los idiomas. El primer esfuerzo lo hizo *ASCII* y fue para el idioma inglés (128 caracteres), luego ante su insuficiencia para representar otros caracteres como los latinos por ejemplo, nace *ISO-8859-1* (también llamado *LATIN-1* ó *ASCII* extendido) pero debido a que no podía representar caracteres de otros idiomas aparece el estándar Unicode (del cual es parte *UTF-8*).

Un buen detalle a saber es que mientras *ISO-8859-1* usa un byte para representar un carácter, no pasa lo mismo con *UTF-8* que puede usar hasta 4 bytes ya que es de longitud variable. Esto hace que una base de datos en *UTF-8* sea un poco más grande que una en ISO-8859-1. Esto sucede porque -por ejemplo- mientras *ISO-8859-1* usa un byte para representar la letra ñ, *UTF-8* usa dos bytes. Hay un tema más y es que muchas veces cuando

[91] Un compilador es un programa informático que traduce un programa que ha sido escrito en un lenguaje de programación a un lenguaje común, usualmente lenguaje de máquina, aunque también puede ser traducido a un código intermedio (bytecode) o a texto y que reúne diversos elementos o fragmentos en una misma unidad, este proceso de traducción se conoce como compilación.

[92] Un depurador (en inglés, debugger), es un programa usado para probar y depurar (eliminar) los errores del programa "objetivo". El código a ser examinado puede alternativamente estar corriendo en un simulador de conjunto de instrucciones (ISS), una técnica que permite gran potencia en su capacidad de detenerse cuando son encontradas condiciones específicas pero será típicamente más lento que ejecutando el código directamente en el apropiado (o el mismo) procesador.

vamos a migrar información nos encontramos con caracteres *ISO-8859-1* (los correspondientes a los números 147, 148, 149, 150, 151 y 133) que no pueden verse en un editor *UNIX/LINUX* pero si en un navegador *HTML*.

Unicode es un set de caracteres universal, es decir, un estándar en el que se definen todos los caracteres necesarios para la escritura de la mayoría de los idiomas hablados en la actualidad que se usan en la computadora. Su objetivo es ser, y, en gran medida, ya lo ha logrado, un superconjunto de todos los sets de caracteres que se hayan codificado. El texto que aparece en la computadora o en la Web se compone de caracteres. Los caracteres representan letras del abecedario, signos de puntuación y otros símbolos.

En el pasado, distintas organizaciones han recopilado diferentes sets de caracteres y han creado codificaciones específicas para ellos. Un set puede abarcar tan sólo los idiomas de Europa occidental con base en el latín (sin incluir países de la Unión Europea como Bulgaria o Grecia), otro set puede contemplar un idioma específico del Lejano Oriente (como el japonés), y otros pueden ser parte de distintos sets diseñados especialmente para representar otro idioma de algún lugar del mundo.

Lamentablemente, no es posible garantizar que su aplicación particular pueda soportar todas las codificaciones, ni que una determinada codificación pueda soportar todos sus requerimientos para la representación de un cierto idioma. Además, generalmente resulta imposible combinar distintas codificaciones en la misma página Web o en una base de datos, por lo que siempre es muy difícil soportar páginas plurilingües si se aplican enfoques "antiguos" cuando se trata de tareas de codificación.

El Consorcio Unicode proporciona un único y extenso set de caracteres que pretende incluir todos los caracteres necesarios para cualquier sistema de escritura del mundo, incluyendo sistemas ancestrales (como el cuneiforme, el gótico y los jeroglíficos egipcios). Hoy resulta fundamental para la arquitectura de la Web y de los sistemas operativos, y las principales aplicaciones y navegadores Web incluyen soporte para este elemento. En el Estándar Unicode también se describen las propiedades y algoritmos necesarios para trabajar con caracteres. Este enfoque facilita mucho el trabajo con sistemas o páginas plurilingües y responde mucho mejor a las necesidades del usuario que la mayoría de los sistemas de codificación tradicionales.

Sets de caracteres, sets de caracteres codificados y codificaciones un set de caracteres o repertorio comprende el grupo de caracteres que se utilizarían para una finalidad específica, ya sea los necesarios para el soporte de los idiomas de Europa Occidental en la computadora.

Un set de caracteres codificados es un grupo de caracteres en el que se ha asignado un número exclusivo a cada carácter. Las unidades de un set de caracteres codificados se conocen como puntos de código. El valor de un punto de código representa la ubicación de un carácter en el set de caracteres codificados. Por ejemplo, el punto de código para la letra *á* en el set de caracteres codificados Unicode es *225* en notación decimal, o *E1* en notación hexadecimal.

La codificación de caracteres refleja la manera en la que el set de caracteres codificados se convierte a bytes para su procesamiento en la computadora. Además, en Unicode existen distintas formas de codificar el mismo carácter. Por ejemplo, la letra *á* se puede representar mediante dos bytes en una codificación y con cuatro bytes, en otra. Los formatos de codificación que se pueden usar con Unicode se denominan *UTF-8, UTF-16 y UTF-32*.

Por todo lo anterior, al programar es necesario tener en cuenta la codificación usada por el editor o IDE que se use para ello y que no todos los editores soportan las mismas codificaciones[93], además puede haber problemas de portabilidad en los archivos entre distintos sistemas operativos. En el código fuente (las instrucciones del programa) no se suele usar caracteres distintos al *ASCII*, pero en las cadenas de visualización o en la documentación es común el uso de caracteres acentuados, es aquí donde hay que tomar una decisión sobre el usar o no dichos caracteres, siempre y cuando el compilador los soporte.

Si siempre se usa el mismo editor y la misma plataforma de desarrollo, no hay razón para no usar caracteres extendidos como los acentos. Pero si se usarán múltiples sistemas operativos y no hay garantía de usar editores que soporten dichos caracteres, entonces existe la posibilidad de perder dichos caracteres o bien pueden generar errores al compilar los archivos por no ser soportados. Por ello una opción para evitar problemas es sólo usar caracteres *ASCII* o tener el cuidado de usar editores que no pierdan dichos caracteres.

En Linux, para verificar la codificación de un archivo se utiliza el comando

[93] Dado que los archivos fuente se intercambian entre usuarios y es común el uso de diferentes sistemas operativos, la conversiones de los caracteres entre diferentes formatos puede ser causa de problemas de codificación, perdiéndose dichos caracteres en la conversión.

file –i o *–mime*, este comando permite mostrar en pantalla el tipo de archivo y su codificación, usando:

```
$ file -i Car.java
```

El comando *iconv* es utilizado para realizar esta tarea de convertir el código de un texto a otro. La lógica para aplicar correctamente el commando *iconv* es la siguiente:

```
$ iconv options -f from-encoding -t to-encoding inputfile(s) -o
outputfile
```

así, *-f* o *–from-code* significa la entrada de la codificación, y *–t* o *–to-encoding* especifica la salida de la misma. Para enumerar todos los juegos de caracteres podemos usar *-l* o *–list*:

```
$ iconv -l
```

Con todo esto en mente podemos proceder a explicar la codificación de *UTF-8* a *ASCII*. Primero hay que comenzar con conocer las codificaciones de los caracteres en el archivo y luego poder ver el contenido del mismo. Así, se podrán convertir todos los archivos a la codificación *ASCII*. Todo después de haber utilizado el comando *iconv*, para poder verificar lo que contiene la salida del archivo. Para eso hay que hacer lo siguiente:

```
$ file -i input.file
$ cat input.file
$ iconv -f ISO-8859-1 -t UTF-8//TRANSLIT input.file -o
out.file
$ cat out.file
$ file -i out.file
```

Cabe destacar que, si el comando *//IGNORE* se añade a *to-encoding*, los caracteres no pueden ser convertidos y un error se mostrará luego de la conversión. También, si el comando *//TRANSLIT* es añadido a *to-encoding* como en el ejemplo dado *(ASCII//TRANSLIT)*, los caracteres convertidos son transliterados, si es posible, como necesarios.

Esto implicaría que en este evento los caracteres no pueden ser representados como se desea, aunque pueden haber aproximaciones del mismo,

inclusive dos. Por lo que, si un carácter no puede ser transliterado, no será reconocido como un objetivo para reemplazo y se mostrará la marca (?) en la salida del archivo.

Algunas veces es necesario convertir el archivo de UTF-8 a ASCII y lo hacemos mediante:

```
$ iconv -f UTF-8 -t ISO-8859-1 prog.c -o progMod.c
```

o mediante:

```
$ iconv -f UTF-8 -t ASCII//TRASLIT prog.c -o progMod.c
```

8.7.2 Uso de Espacios o Tabuladores en Fuentes

En muchos lenguajes de programación la identación es una forma opcional de hacer legible el código para el programador, en otros (como Python) es imprescindible (ya que de ella dependerá su estructura).

¿Qué es la identación? En un lenguaje informático, la identación es lo que a la sangría al lenguaje humano escrito (a nivel formal). Así como para el lenguaje formal, cuando uno redacta una carta, debe respetar ciertas sangrías, los lenguajes informáticos, pueden requerir una identación.

No todos los lenguajes de programación necesitan una identación, aunque sí se estila implementarla, a fin de otorgar mayor legibilidad al código fuente. Una identación depende del lenguaje y estilo de codificación usado, por ejemplo en C, C++ y Java se acostumbra usar tres espacios en blanco. En el caso de Python se suele usar 4 espacios en blanco o un tabulador, que indicará que las instrucciones identadas forman parte de una misma estructura de control.

Los programadores siempre han debatido entre el uso de espacios y tabulaciones para estructurar su código. Los espacios y las tabulaciones son utilizados por los programadores para estructurar el código de una forma determinada. La primera línea de código (sin espacio o tabulación) inicia un "bloque" de contenido. Si las sucesivas líneas de código forman parte de ese mismo bloque (encerrado entre corchetes) o forman nuevos subbloques, estas se van desplazando hacia la derecha para indicar esa subordinación. En caso de formar un bloque completamente nuevo, se mantiene en la misma posición que la línea inmediatamente anterior.

A nivel funcional, la diferencia entre el uso de espacios o tabulaciones es nula. Cuando el código pasa por el compilador antes de ser ejecutado, la máquina interpreta de igual forma ambos formatos. No obstante, sí existen diferencias técnicas que marcan la diferencia entre el uso de tabulaciones y espacios:

- Precisión. Una tabulación no es más que un conjunto de espacios agrupados. Por norma general, este conjunto suele ser de 8 caracteres, pero puede variar. ¿Qué quiere decir esto? Que cuando un mismo fichero de código se abre en dos máquinas diferentes, la apariencia del código puede ser diferente. En cambio, el uso de espacios no conlleva este problema: un espacio siempre ocupa el mismo "espacio" —valga la redundancia— y asegura que el código se visualiza de la misma forma en todas las máquinas.

- Comodidad. En el caso de las tabulaciones, basta con pulsar la tecla de tabulación una única vez para estructurar correctamente el código. En el caso de los espacios, es necesario pulsar varias veces la misma tecla para lograr la estructura deseada.

- Almacenamiento. El uso de tabulaciones también reduce el tamaño del fichero final, mientras que el uso de espacios lo aumenta. Lo mismo sucedería con el uso de espacios en lugar de saltos de línea.

Entonces, ¿cuál es la más correcta? La realidad es que todo depende de las preferencias personales. Si necesitas optimizar el tamaño de los ficheros al máximo, el uso de espacios se convierte en un sacrilegio. Si, en cambio, tu código debe lucir exactamente igual en múltiples máquinas, el uso de espacios puede ser más conveniente para lograr esa homogeneidad.

Por suerte, existen múltiples editores en la actualidad que trabajan y facilitan la transición entre ambos sistemas. Asimismo, los equipos de desarrollo de software establecen en sus guidelines el uso de espacios o tabulaciones. De esta forma, se evitan conflictos entre los programadores de un mismo proyecto y se alcanza esa homogeneidad tan deseada.

El comando *expand* y *unexpand* (que vienen instalados en los paquetes *GNU Core*) permite convertir tabuladores en espacios y viceversa, según nuestras necesidades o gustos. Estos comandos sacan el resultado de stdin o de los archivos nombrados en la línea de comando. Utilizando la opción *-t* se pueden establecer una o más posiciones de tabulador.

Para ver si se usan espacios o tabuladores en un archivo fuente podemos usar el comando *cat* con la opción *-T* que nos mostrará los caracteres tabulador como *^I*, ejemplo:

```
$ cat -T archivo
```

Para convertir los espacios en tabuladores (un tabulador igual a 8 espacios) usamos:

```
$ unexpand progEsp.c
```

o redireccionando la salida usando:

```
$ unexpand progEsp.c > progTab.c
```

Para convertir los tabuladores en espacios (1 tabulador por ejemplo 4 caracteres) usamos:

```
$ expand -t 4 progTab.c
```

o redireccionando la salida usando:

```
$ expand -t 4 progTab.c > progEsp.c
```

También es posible buscar todos los archivos (digamos *.cpp) y cambiar los tabuladores por 4 espacios (para ello usamos el comando *sponge* que está contenido en el paquete *moreutils*), mediante:

```
$ find . -name '*.cpp' -type f -exec bash -c \
'expand -t 4 "$0" | sponge "$0"' {} \;
```

8.7.3 Comparar Contenido de Fuentes

Cuando se programa es común generar distintas versiones de un mismo archivo, en GNU/Linux se tiene varias herramientas para comparar y combinar cambios. En la línea de comandos el comando *diff* permite ver los cambios entre dos versiones de un archivo y el comando *merge* sirve para combinar cambios. Por otro lado *sdiff* nos permite ver las diferencias entre dos archivos y de forma interactiva combinar cambios.

Pese a que son poderosos estos comandos, en forma gráfica se puede obtener todo su potencial. Algunas de estas opciones son:

```
# apt install kdiff3 meld diffuse dirdiff kompare wdiff  \
numdiff colordiff xxdiff tkdiff ndiff
```

Estos permiten comparar dos o tres versiones de un archivo simultáneamente, y hacerlo con el contenido de una o más carpetas. Cada uno tiene la capacidad de mostrar los cambios y si se requiere hacer la combinación de ellos.

meld nos muestra gráficamente las diferencias entre dos archivos o también, entre todos los archivos de dos directorios utilizando distintos colores, y nos permite editar estos archivos desde el propio programa, actualizando dinámicamente las diferencias. El programa incluye filtros y distintas ayudas para hacer la edición más sencilla, como flechas al lado de los cambios para aplicar cambio en cualquiera de los archivos con un simple clic. Este programa se puede utilizar como un sencillo cliente de control de cambios para Git, CVS, Subversion, etc.

kdiff3 nos muestra gráficamente las diferencias entre tres archivos utilizando distintos colores, y nos permite editar estos archivos desde el propio programa, actualizando dinámicamente las diferencias. El programa incluye filtros y distintas ayudas para hacer la edición más sencilla, como flechas al lado de los cambios para aplicar cambio en cualquiera de los archivos con un simple clic.

8.7.4 Astyle

Para dar uniformidad a la codificación de los programas fuente, se puede usar un formateador automático de código, *Astyle* soporta una gran variedad de lenguajes y opciones, para instalar en Debian GNU/Linux usar:

```
# apt install astyle
```

para formatear los archivos de C, C++, C# usar:

```
$ astyle -s3 -p –style=allman –lineend=Linux *.?pp
```

para Java, una opción es

```
$ astyle -s3 -p –style=java –lineend=Linux *.java
```

Algunos estilos disponibles son:

style=allman style=java style=kr style=stroustrup
style=whitesmith style=vtk style=ratliff style=gnu
style=Linux style=horstmann style=1tbs style=google
style=mozilla style=pico style=lisp

más opciones en:

- http://astyle.sourceforge.net/astyle.html
- https://en.wikipedia.org/wiki/Programming_style
- https://en.wikipedia.org/wiki/Indent_style

Aprender a Usar ASTYLE en la red existen múltiples sitios especializados y una amplia bibliografía para formatear código nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Sistema Operativo Linux

8.7.5 Compilación y la Optimización del Ejecutable

Al programar es necesario revisar nuestro código por un compilador y los errores son inherentes al proceso de programación. Los errores de programación responden a diferentes tipos y pueden clasificarse dependiendo de la fase en que se presenten. Algunos tipos de errores son más difíciles de detectar y reparar que otros, veamos entonces:

- Errores de sintaxis
- Advertencias
- Errores de enlazado
- Errores de ejecución
- Errores de diseño

Errores de sintaxis son errores en el código fuente. Pueden deberse a palabras reservadas mal escritas, expresiones erróneas o incompletas, variables que no han sido declaradas, etc. Los errores de sintaxis se detectan en la fase de compilación. El compilador, además de generar el código objeto, nos dará una lista de errores de sintaxis. De hecho nos dará sólo una cosa o la otra, ya que si hay errores no es posible generar un código objeto.

Advertencias además de errores, el compilador puede dar también advertencias (Warnings). Las advertencias son errores, pero no lo suficientemente graves como para impedir la generación del código objeto. No obstante, es importante corregir estos errores la mayoría de las veces, ya que ante un aviso el compilador tiene que tomar decisiones, y estas no tienen porqué coincidir con lo que nosotros pretendemos hacer, ya se basan en las directivas que los creadores del compilador decidieron durante la creación del compilador. Por lo tanto, en ocasiones, ignorar las advertencias puede ocasionar que nuestro programa arroje resultados inesperados o erróneos.

Errores de enlazado el programa enlazador también puede encontrar errores. Normalmente se refieren a funciones que no están definidas en ninguno de los ficheros objetos ni en las bibliotecas. Puede que hayamos olvidado incluir alguna biblioteca, o algún fichero objeto, o puede que hayamos olvidado definir alguna función o variable, o lo hayamos hecho mal.

Errores de ejecución incluso después de obtener un fichero ejecutable, es posible que se produzcan errores durante la ejecución del código. En el caso de los errores de ejecución normalmente no obtendremos mensajes específicos de error o incluso puede que no obtengamos ningún error, sino que simplemente el programa terminará inesperadamente. Estos errores son más difíciles de detectar y corregir (pues se trata de la lógica como tal de nuestra aplicación). Existen herramientas auxiliares para buscar estos errores, son los llamados depuradores (Debuggers). Estos programas permiten detener la ejecución de nuestros programas, inspeccionar variables y ejecutar nuestro programa paso a paso (instrucción a instrucción). Esto resulta útil para detectar excepciones, errores sutiles, y fallos que se presentan dependiendo de circunstancias distintas. Generalmente los errores en tiempo de ejecución se dan por situaciones no consideradas en la aplicación, por ejemplo, que el usuario ingrese una letra en vez de un número y ésto no es controlado.

Errores de diseño finalmente los errores más difíciles de corregir y prevenir. Si nos hemos equivocado al diseñar nuestro algoritmo, no habrá ningún programa que nos pueda ayudar a corregirlos, pues es imposible que un programa pueda determinar qué es lo que tratamos de conseguir o un programa que realice aplicaciones cualquiera por nosotros. Contra estos errores sólo cabe practicar y pensar, realizar pruebas de escritorio, hacerle seguimiento y depuración a la aplicación hasta dar con el problema (una mala asignación, un valor inesperado, olvidar actualizar una variable, etc.), también es útil buscar un poco de ayuda de libros o en sitios y foros especializados.

Compilación y la Optimización del Ejecutable Para usar muchas de estas herramientas (en línea de comandos), primero debemos conocer cómo compilar fuentes[94], sin pérdida de generalidad trabajaremos en C++ solicitando que el archivo ejecutable[95] tenga el nombre *ejemp*:

```
$ g++ *.cpp -o ejemp
```

Para ejecutar el programa ya compilado:

```
$ ./ejemp
```

Para compilar y ver todos los avisos usar:

```
$ g++ -pedantic -Wall -Wextra -O *.cpp
```

o de forma alternativa:

```
$ g++ -Weffc++ *.cpp
```

Por otro lado, también podemos hacer una revisión estática del código, por ejemplo en C++ usamos:

```
$ cppcheck –enable=all *.?pp
```

[94] Compilador para C es gcc, para C++ es g++, para Fortran es f77 o f95, etc.

[95] Un archivo ejecutable es tradicionalmente un archivo binario con instrucciones en código de máquina cuyo contenido se interpreta por el ordenador como un programa. Además, suele contener llamadas a funciones específicas de un sistema operativo.

mostrará los avisos de análisis estático del código indicado.

Para conocer el tiempo de ejecución[96] de un programa, podemos usar el comando básico *time*, mediante:

```
$ time ejecutable
```

que entregará información del tipo:

```
$ time ls

real    0m0.004s
user    0m0.001s
sys     0m0.004s
```

Pero podemos instalar una versión optimizada de este comando que proporciona información adicional, para ello instalar:

```
# apt install time
```

y su ejecución mediante:

```
$ /usr/bin/time ejecutable
```

por ejemplo para el comando *ls*, entrega una salida del tipo:

```
$ /usr/bin/time -v ls
   Command being timed: "ls"
   User time (seconds): 0.00
   System time (seconds): 0.00
   Percent of CPU this job got: 66%
   Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.00
   Average shared text size (kBytes): 0
   Average unshared data size (kBytes): 0
   Average stack size (kBytes): 0
```

[96] El tiempo total de ejecución de un programa (tiempo real) es la suma del tiempo de ejecución del programa del usuario (tiempo de usuario) más el tiempo de ejecución del sistema necesario para soportar la ejecución (tiempo de sistema).

Average total size (kBytes): 0
Maximum resident set size (kBytes): 2360
Average resident set size (kBytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 110
Voluntary context switches: 1
Involuntary context switches: 1
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (Bytes): 4096
Exit status: 0

Además, podemos compilar usando diversas optimizaciones[97] disponibles en todos los compiladores GNU de Linux, estas opciones de optimización están ordenadas, desde no optimizar, a la mejor optimización posible, estas son: –O0, O1, -O2, -O3, -O3 -march=native, -O3 -march=native -fito, -Ofast -march=native.

Para compilar y solicitar la optimización usamos:

```
$ g++ -O1 *.cpp
```

y para ejecutar el programa usamos:

```
$ ./a.out
```

El resultado de las optimizaciones dependen del programa y se puede ver que el rendimiento (tiempo de ejecución) mejora en varios órdenes de magnitud.

Por ejemplo en los siguientes test[98] se obtienen estos rendimientos:
Cryto++ v8.2:

[97] La optimización de código es el conjunto de fases de un compilador que transforma un fragmento de código en otro fragmento con un comportamiento equivalente y se ejecuta de forma más eficiente, es decir, usando menos recursos de cálculo como memoria o tiempo de ejecución.

[98] https://www.phoronix.com/scan.php?page=news_item&px=GCC-10.1-Compiler-Optimizations

O0 (95), -O2 (660.46), -O3 (712.01), -O3 -march=native (751.56), -O3 -march=native -fito (699.80), -Ofast -march=native (751.01)

LeelaChessZero:

O0 (18,300), -O2 (157,289), -O3 (142,198), -O3 -march=native (136,608), -O3 -march=native -fito (163,773), -Ofast -march=native (157,629)

Himeno Benchmark v3.0:

O0 (597.53), -O2 (4,150.11), -O3 (4,015.86), -O3 -march=native (4,771.42), -O3 -march=native -fito (4,774.03), -Ofast -march=native (5,065.07)

C-Ray v1.1:

O0 (113.58), -O2 (69.70), -O3 (38.00), -O3 -march=native (30.46), -O3 -march=native -fito (30.24), -Ofast -march=native (27.13)

Geometric Mean Of All Test Results:

O0 (222.36), -O2 (681.88), -O3 (709.76), -O3 -march=native (735.14), -O3 -march=native -fito (755.97), -Ofast -march=native (758.30)

8.7.6 Análisis de Rendimiento y Depuración

El comando *gprof* produce un perfil de ejecución de programas en C, C++, Pascal, FORTRAN77, etc. El efecto de las rutinas llamadas se incorpora en el perfil de cada llamador. Los datos del perfil se toman del fichero de perfil de grafos de llamada ('gmon.out' por omisión) que es creado por programas que se han compilado con la opción -pg de cc(1), pc(1), y f77(1). La opción -pg también enlaza al programa versiones de las rutinas de biblioteca que están compiladas para la perfilación. *Gprof* lee el fichero objeto dado (el predeterminado es 'a.out') y establece la relación entre su tabla de símbolos y el perfil de grafo de llamadas de 'gmon.out'. Si se especifica más de un fichero de perfil, la salida de gprof muestra la suma de la información de perfilado en los ficheros de perfil dados.

Gprof calcula la cantidad de tiempo empleado en cada rutina. Después, estos tiempos se propagan a lo largo de los vértices del grafo de llamadas. Se

descubren los ciclos, y se hace que las llamadas dentro de un ciclo compartan el tiempo del ciclo. El primer listado muestra las funciones clasificadas de acuerdo al tiempo que representan incluyendo el tiempo de sus descendientes en su grafo de llamadas. Debajo de cada entrada de función se muestran sus hijos (directos) del grafo de llamadas, y cómo sus tiempos se propagan a esta función. Un despliegue similar sobre la función muestra cómo el tiempo de esta función y el de sus descendientes se propagan a sus padres (directos) del grafo de llamadas.

También se muestran los ciclos, con una entrada para el ciclo completo y un listado de los miembros del ciclo y sus contribuciones al tiempo y número de llamadas del ciclo. En segundo lugar, se da un perfil plano, similar al producido por *prof*. Este listado de los tiempos de ejecución totales, los números de llamadas, el tiempo en milisegundos que la llamada empleó en la propia rutina, y el tiempo en ms que la llamada empleó en la propia rutina pero incluyendo sus descendientes. Finalmente, se proporciona un índice a los nombres de función.

Para obtener el análisis de rendimiento, hacemos:

```
$ g++ -g -pg -O0 *.cpp
$ ./a.out
$ gprof -c -z a.out gmon.out > sal.txt
```

el archivo *sal.txt* contiene el análisis de rendimiento detallado. Un ejemplo de esta salida es:

```
Flat profile:
Each sample counts as 0.01 seconds.
% cumulative self self total
time seconds seconds calls s/call s/call name
23.25    0.60    0.60  40656734     0.00     0.00 retorna(int, int)
14.85    0.98    0.38  27627674     0.00     0.00 retoaNumColu(int, int)
12.89    1.31    0.33  91126931     0.00     0.00 Vector::retorna(int)
10.94    1.59    0.28        31     0.01     0.03 ResJacobi::resuelve()
...
```

que permite conocer en que parte del código se consume más tiempo de ejecución.

Aprender a Usar GPROF en la red existen múltiples sitios especializados y una amplia bibliografía para optimizar código, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

GPROF

Depuración con ddd un depurador (en inglés: Debugger) es un programa usado para probar y depurar (eliminar) los errores de otros programas (el programa "objetivo"). El código a ser examinado puede alternativamente estar corriendo en un simulador de conjunto de instrucciones (ISS), una técnica que permite gran potencia en su capacidad de detenerse cuando son encontradas condiciones específicas pero será típicamente algo más lento que ejecutando el código directamente en el apropiado (o el mismo) procesador. Algunos depuradores ofrecen dos modos de operación - la simulación parcial o completa, para limitar este impacto.

Si es un depurador de nivel de fuente o depurador simbólico, comúnmente ahora visto en entornos de desarrollo integrados, cuando el programa "se estrella" o alcanza una condición predefinida, la depuración típicamente muestra la posición en el código original. Si es un depurador de bajo nivel o un depurador de lenguaje de máquina, muestra la línea en el fuente desensamblado (a menos que también tenga acceso en línea al código fuente original y pueda exhibir la sección apropiada del código del ensamblador o del compilador). Un "estrellamiento" sucede cuando el programa no puede continuar normalmente debido a un error de programación. Por ejemplo, el programa pudo haber intentado usar una instrucción no disponible en la versión actual del CPU o haber intentado tener acceso a memoria protegida o no disponible.

Típicamente, los depuradores también ofrecen funciones más sofisticadas tales como correr un programa paso a paso (un paso o animación del programa), parar el programa (Breaking), es decir, pausar el programa para examinar el estado actual en cierto evento o instrucción especificada por medio de un Breakpoint, y el seguimiento de valores de algunas variables. Algunos depuradores tienen la capacidad de modificar el estado del programa mientras que está corriendo, en vez de simplemente observarlo. También es posible continuar la ejecución en una posición diferente en el programa pasando un estrellamiento o error lógico.

La importancia de un buen depurador no puede ser exagerada. De hecho, la existencia y la calidad de tal herramienta para un lenguaje y una

plataforma dadas a menudo puede ser el factor de decisión en su uso, incluso si otro lenguaje/plataforma es más adecuado para la tarea.

Para hacer depuración del código mediante el depurador gráfico *ddd* usar:

```
$ g++ -g -O0 *.cpp
$ ddd ./a.out
```

Puede usarse también los depuradores *xxgdb, gdb, kdbg, nevimer, lldb* cada uno tiene sus pros y contras, depende del usuario cual es el más adecuado para usar.

Depuración con valgrind es un conjunto de herramientas libres que ayuda en la depuración de problemas de memoria y rendimiento de programas.

La herramienta más usada es *Memcheck*. *Memcheck* introduce código de instrumentación en el programa a depurar, lo que le permite realizar un seguimiento del uso de la memoria y detectar los siguientes problemas:

- Uso de memoria no inicializada.
- Lectura/escritura de memoria que ha sido previamente liberada.
- Lectura/escritura fuera de los límites de bloques de memoria dinámica.
- Fugas de memoria.
- Otros.

El precio a pagar es una notable pérdida de rendimiento; los programas se ejecutan entre cinco y veinte veces más lento al usar Valgrind, y su consumo de memoria es mucho mayor. Por ello normalmente no siempre se ejecuta un programa en desarrollo usando Valgrind, sino que se usa en situaciones concretas cuando se está buscando un error determinado se trata de verificar que no haya errores ocultos como los que Memcheck puede detectar.

Valgrind incluye además otras herramientas:

- Addrcheck, versión ligera de Memcheck que se ejecuta más rápido y requiere menos memoria pero que detecta menos tipos de errores.
- Massif, mide el rendimiento del montículo (heap).

- Helgrind, herramienta de detección de condiciones de carrera (race conditions) en código multihilo.
- Cachegrind, mide el rendimiento de la Caché durante la ejecución, de acuerdo a sus características (capacidad, tamaño del bloque de datos, grado de asociatividad, etc.).

Para el rastreo de problemas con la manipulación de memoria y punteros desbordados usamos:

```
$ g++ -g -O0 *.cpp
$ valgrind –tool=memcheck –leak-check=yes –show-reachable=yes ./a.out
```

o analizar la salida usando *kCachegrind*:

```
$ valgrind –tool=callgrind ./a.out
$ kCachegrind profile.callgrind
```

Aprender a Usar VALGRIND en la red existen múltiples sitios especializados y una amplia bibliografía para depurar código, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

VALGRIND

Existen varios paquetes de modo gráfico para *valgrind*, uno de ellos es *alleyoop* y se usa:

```
$ alleyoop ./a.out -v –arg1=foo
```

otro es *kCachegrind*, podemos ver más opciones en:

- http://valgrind.org/
- http://alleyoop.sourceforge.net/usage.html
- http://kCachegrind.sourceforge.net/html/Home.html

8.7.7 Mejora del Rendimiento en Python

Python es un lenguaje interpretado, pero es posible compilar el código para generar Byte Code para el intérprete (no aumenta la velocidad de ejecución). Si se necesita crear un archivo *.pyc* para un módulo que no se importa, se puede usar los módulos *py_ compile* y *compile_ all* desde el intérprete de Python.

El módulo py_compile puede compilar manualmente cualquier módulo. Una forma de usar la función *py_ compile.compile* en ese módulo de forma interactiva es:

```
>>> import py_compile
>>> py_compile.compile('modulo.py')
```

esto escribirá el archivo *modulo*.pyc.

En la línea de comando de Linux es posible compilar todos los archivos en un directorio utilizando el módulo compileall, para ello usar:

```
$ python -m compileall *.py
```

y ejecutar mediante:

```
$ python modulo.pyc
```

También es posible hacer ligera optimización del código generado mediante:

```
$ python –O -m compileall *.py
```

esto generará código Bytecode con extensión *.pyo*, y ejecutar mediante:

```
$ python modulo.pyo
```

Python es un lenguaje razonablemente rápido, pero no es tan rápido como los programas compilados de C o Fortran. Eso es porque se interpreta *CPython*, la implementación estándar. Para ser más precisos, su código de Python se compila en un código de *Bytes* que luego se interpreta. Eso es bueno para aprender, ya que puede ejecutar el código en el *REPL* de Python

y ver los resultados de inmediato en lugar de tener que compilar y ejecutar. Pero como los programas de Python no son tan rápidos, los desarrolladores han creado varios compiladores de Python[99] a lo largo de los años, incluidos[100] Numba, *Nuitka, PyPy, Cython, cx_FreezeIron, Pythran, Jython entre otros.*

Comparando Compiladores de Python Alguien ya ha hecho el trabajo de crear un punto de referencia de Python. Opté por *PyStone*, una traducción de un programa en C de Guido van Rossum, el creador de Python (el programa en C era en sí mismo una traducción de un programa *Ada*). Encontré una versión convertida por el desarrollador Christopher Arndt en *Github* que era capaz de probar Python 3. Para dar un sentido de perspectiva, aquí está el rendimiento de *CPython* (es decir, Python estándar) con *Pystone*:

> Python 2.7.15Rc1 2: 272.647 pystones / second.
> Python 3.6.5: 175,817

Como puede ver, hay una gran diferencia entre Python 2 y 3 (cuanto más Pystones por segundo, mejor). En los siguientes desgloses, todos los compiladores de Python se compararon con Python 3.

Nuitka Aunque puede seguir las instrucciones en la página de descarga, lo siguiente en Debian GNU/Linux funcionó bien para mí:

```
$ apt install nuitka
```

adicionalmente Nuitka también puede usar otro compilador de C (además del gcc), así que descargué clang. Puedes instalarlo con esto:

```
$ apt install clang
```

De forma predeterminada, *Nuitka* usa *gcc*, pero un parámetro te permite usar el *clang*, así que lo probé con ambos. El compilador clang es parte de la familia llvm, y está pensado como un reemplazo moderno para *gcc*.

[99] El rendimiento rápido no es la única razón para compilar; Posiblemente la mayor desventaja de los lenguajes de *Scripting* como Python es que se proporciona de manera implícita su código fuente a los usuarios finales.

[100] Si está interesado en los compiladores de Python en general, tenga en cuenta que hay mucho debate y controversia sobre los "mejores" compiladores y la rapidez general del lenguaje.

Compilar pystone.py con *gcc* fue tan simple como esto (primera línea), o con *clang* (segunda línea), y con la optimización del tiempo de enlace para *gcc* (tercera línea):

```
$ nuitka pystone.py
$ nuitka pystone.py –clang
$ nuitka pystone.py –lto
```

Después de compilar, lo que tomó aproximadamente 10 segundos, ejecuté el pystone.exe desde la terminal con:

```
$ ./pystone.exe 500000
```

Hice 500,000 pases:

Tamaño Ejecución pystones / seg.

1. 223.176 Kb 597,000
2. 195,424 Kb 610,000
3. 194.2 kb 600,000

Estos fueron los promedios de más de 5 corridas. Había cerrado tantos procesos como pude, pero tómo los tiempos con un poco de variación porque había un +/- 5% en los valores de tiempo.

PyPy Guido van Rossum dijo una vez: "Si quieres que tu código se ejecute más rápido, probablemente debas usar PyPy". Para instalarlo en Debian GNU/Linux usar:

```
$ apt install pypy
```

Entonces lo corrí así:

```
$ pypy pystone.py
```

El resultado fue una asombrosa cantidad de 1,776,001 pystones por segundo, casi tres veces más rápido que *Nuitka*.

PyPy usa un compilador justo a tiempo y hace algunas cosas muy inteligentes para alcanzar su velocidad. De acuerdo con los puntos de referencia reportados, es 7.6 veces más rápido que el *CPython* en promedio. Puedo creer eso fácilmente. La única (leve) desventaja es que siempre está un poco por detrás de las versiones de Python (es decir, hasta 2.7.13 (no 2.7.15) y 3.5.3 (no 3.6.5)). Producir un exe requiere un poco de trabajo. Tienes que escribir tu Python en un subconjunto llamado *RPython*.

Cython no es solo un compilador para Python; es para un superconjunto de Python que admite la interoperabilidad con C / C++. CPython está escrito en C, por lo que es un lenguaje que generalmente se combina bien con Python.

Configurar las cosas con *Cython* es un poco complicado. No es como *Nuitka*, que acaba de salir de la caja. Primero, debes comenzar con un archivo de Python con una extensión .pyx; ejecuta *Cython* para crear un archivo pystone.c a partir de eso:

$ cython pystone.pyx –embed

No omita el parámetro *-embed*. Se agrega en *main* y eso es necesario. A continuación, compila pystone.c con esta hermosa línea:

$ gcc $ (python3-config –includes) pystone.c -lpython3.6m -o pystone.exe

Si recibe algún error, como "no se puede encontrar la versión -lpython", podría ser el resultado de su versión de Python. Para ver qué versión está instalada, ejecute este comando:

$ pkg-config –cflags python3

Después de todo eso, *Cython* solo dio 228,527 pystones / sec. Sin embargo, *Cython* necesita que hagas un poco de trabajo especificando los tipos de variables. Python es un lenguaje dinámico, por lo que no se especifican los tipos; *Cython* utiliza la compilación estática y el uso de variables de tipo C le permite producir un código mucho mejor optimizado. (La documentación es bastante extensa y requiere lectura).

Tamaño Ejecución pystones / seg.

1. 219.552 Kb 228.527

cx_freeze es un conjunto de Scripts y módulos para "congelar" Scripts de Python en ejecutables, y se pueden encontrar en *Github*.

Lo instalé y creé una carpeta congelada para administrar cosas en:

$ pip3 install cx_Freeze –upgrade

Un problema que encontré con el Script de instalación fue un error que falta "lz". Necesitas tener instalado zlib; ejecuta esto para instalarlo:

$ apt install zlib1g-dev

Después de eso, el comando *cx_Freeze* tomó el Script *pystone.py* y creó una carpeta dist que contenía una carpeta *lib*, un archivo *lib* de 5MB y el archivo de aplicación pystone:

$ cxfreeze pystone.py –target-dir dist

Tamaño Ejecución pystones / seg.

1. 10,216 174,822

No es el rendimiento más rápido, porque es la misma velocidad que *CPython*. (La congelación de Python implica enviar su aplicación en un solo archivo (o carpeta) con los elementos Python necesarios, en lugar de compilar; significa que el destino no requiere Python).

Numba Este es un compilador "justo a tiempo" para Python que optimiza el código que se usa en algoritmos numéricos como son en las matrices, bucles y funciones de NumPy (también da soporte a *Threading*, vectorización *SIMD* y aceleración por *GPUs*: *Nvidia CUDA, AMD ROC*). La forma más común de usar Numba es a través de su colección de decoradores que se pueden aplicar a sus funciones para indicar a Numba que las compile usando el estándar *LLVM*. Cuando se realiza una llamada a una función decorada de Numba, se compila en el código de máquina "justo a tiempo" para su ejecución y todo o parte de su código puede ejecutarse posteriormente a la velocidad de código de máquina nativo. Numba también trabaja bien con Jupiter notebook para computación interactiva y con ejecución distribuida como *Dask* y *Spark*.

Se puede instalar en Debian GNU/Linux mediante:

$ apt install python3-numba

y se puede descargar mediante *CONDA* paquete de *Anaconda*, usando:

$ conda install numba

o mediante *PIP* usando:

```
$ pip install numba
```

Dando mejores resultados en la ejecución de múltiples pruebas que *PyPy*, pero no en todos los casos. Por ello, la recomendación es evaluar el rendimiento mediante pruebas en cada caso particular.

Conclusión Una buena opción es *PyPy* por el rendimiento obtenido en código general (y dependiendo del código en cuestión Numba puede ser mejor que *PyPy* en aplicaciones de cómputo científico), la compilación fue muy rápida y produjo los resultados en menos de un segundo después de presionar la tecla *RETURN*. Si requieres un ejecutable, sin embargo, te recomiendo Nuitka; fue una compilación sin complicaciones y se ejecuta más rápido que CPython. Experimenta con estos compiladores de Python y vea cuál funciona mejor para tus necesidades particulares.

8.7.8 Git

Git es un programa de control de versiones que sirve para la gestión de los diversos cambios que se realizan sobre los elementos de algún proyecto de Software y sus respectivos programas fuente o configuración del mismo guardando instantáneas (Snapshots) del código en un estado determinado, que viene dado por el autor y la fecha. Fue diseñado por Linus Torvalds y es usado para controlar los cambios de diversos proyectos como los fuentes del Kernel de Linux que tiene decenas de millones de líneas de código (en la versión 4.12 cuenta con 24,170,860 líneas de código repartidos en 59,806 archivos) y es trabajado por miles de programadores alrededor del mundo.

¿Qué es control de versiones? se define como control de versiones a la gestión de los diversos cambios (Commit es un conjunto de cambios guardados en el repositorio de Git y tiene un identificador SHA1 único) que se realizan sobre los elementos de algún producto o una configuración del mismo, es decir, es lo que se hace al momento de estar desarrollando un Software o una página Web. Exactamente es eso que haces cuando subes y actualizas tu código en la nube, o le añades alguna parte o simplemente editas cosas que no funcionan como deberían o al menos no como tú esperarías.

¿A que le llamamos sistema de control de versiones? son todas las herramientas que nos permiten hacer todas esas modificaciones antes mencionadas en nuestro código y hacen que sea más fácil la administración de las distintas versiones de cada producto desarrollado; es decir *Git*.

Antes de seguir avanzando, conviene definir algunos términos comunes que encontraremos más adelante:

- Un repositorio es una carpeta que contiene los archivos y subdirectorios de un proyecto. Puede ser público o privado, dependiendo de quién deba tener acceso.

- Una rama es una ruta de desarrollo separada en el mismo repositorio. En un mismo proyecto suelen utilizarse ramas separadas para trabajar en nuevas características sin interferir con la versión de producción. Una vez que el código es revisado y probado, un administrador del repositorio puede fusionar los cambios en la rama principal.

- Un commit es una instantánea de un repositorio en un momento dado. Permite a los programadores incluir comentarios y pedir la opinión de otras personas. Usando el Hash que lo identifica puedes volver a un estado anterior del proyecto si es necesario. Antes de que los archivos y directorios puedan ser comiteados, necesitamos decirle a Git que los rastree. Normalmente nos referimos a este paso simplemente como agregar los archivos al área de Staging.

- Un Pull Request (o simplemente PR) es un método para informar a otros desarrolladores y discutir los cambios recientes antes de incorporarlos a la ruta de desarrollo principal.

- Un Fork es un proyecto independiente que se basa en un repositorio determinado. A diferencia de las ramas, no es local a este último. Sin embargo, también puede fusionarse con él a través de un Pull Request adecuado.

- Se puede utilizar un archivo .gitignore para indicar qué contenido local no queremos incluir en el repositorio. Esto nos ayuda a evitar enviar archivos temporales o exponer información sensible en una solución basada en la Web.

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente, es decir *Git* nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún Software o página que implique código el cual necesitemos hacerlo con un grupo de personas.

Algunas de las características más importantes de *Git* son:

- Rapidez en la gestión de ramas (Branche es como una línea de tiempo a partir de los Commit, siempre hay siempre como mínimo una rama principal predefinida llamada Master), debido a que *Git* nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
- Gestión distribuida: Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
- Gestión eficiente de proyectos grandes.
- Realmacenamiento periódico en paquetes.

Con esto en mente, vamos a hacer una introducción a Git desde cero.

Instalación de Git para instalar *Git* completo en el servidor o en la máquina de trabajo, usamos:

```
# apt install git-all
```

Para instalar lo básico de *Git*, si no está instalado:

```
# apt install git
```

otras opciones para trabajar con *Git* son:

```
# apt install git git-all gitk gitg git-cola git-gui qgit tig lighttpd
vim-fugitive
```

También existen otros proyectos parecidos a Git, algunos de ellos son:

```
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs
```

Configuración de Git si se quiere especificar la identidad del que controla el repositorio local en el equipo, debemos usar (por omisión toma la información de la cuenta del usuario y máquina):

```
$ git config –global user.name "Antonio Carrillo"
$ git config –global user.email antoniocarrillo@ciencias.unam.mx
```

si se desea configurar el editor de texto a usar por *Git*, usamos (por omisión es *vim*):

```
$ git config –global Core.editor scite
```

si se desea configurar la herramienta de control de diferencias, usamos (por omisión *vimdiff*):

```
$ git config –global merge.tool meld
```

para comprobar podemos usar:

```
git config –global -list
```

el cual creará un archivo de texto llamado .gitconfig, y podemos ver su contenido, usando:

```
cat ~/.gitconfig
```

Clonar un repositorio Git podemos iniciar clonando algún repositorio remoto de la red, para ello debemos conocer su dirección y usar:

```
$ git clone <dirección>
```

esto permitirá bajar todo el contenido del proyecto a clonar.

Crear un repositorio Git local si lo que requiero es un control personal sin necesidad de compartir los archivos con ningún otro usuario, puedo usar *Git* de forma local en cualquier directorio mediante:

```
$ git init
```

Si se desea agregar la identidad del que controla el repositorio en este directorio, se debe usar:

```
$ git config user.name "Antonio Carrillo"
$ git config user.email antoniocarrillo@ciencias.unam.mx
```

Ahora para agregar los archivos (todos los de este directorio), usar:

```
$ git add .
```

así podemos hacer la confirmación de los cambios, mediante:

```
$ git commit -m "Primer lanzamiento"
```

ahora cada que lo requiera al hacer modificaciones, puedo checar los cambios:

```
$ git status
```

o en forma gráfica con *gitk*, mediante:

```
$ gitk
```

para actualizar los cambios, usar:

```
$ git commit -a -m 'Actualizacion'
```

en caso necesario, podemos mover uno o más archivos de una trayectoria a otra, usando:

```
$ git mv archivo(s) /trayectoria
```

o podemos borrar uno o más archivos, usando:

```
$ git rm archivo(s)
```

además podemos editar el archivo *.gitignore*, para indicar los archivos a ignorar usando líneas independientes para cada tipo.

Podemos ver el histórico de las operaciones que hemos hecho, usando:

```
$ git log
```

y podemos comprobar el estado del repositorio, usando:

```
$ git status
```

La otra alternativa es preparar un directorio para el repositorio ya sea en el servidor o de forma local, mediante:

```
$ mkdir example.git
$ cd example.git
```

Para inicializar el repositorio:

```
$ git –bare init
```

Es buena opción limitar el acceso a la cuenta vía *ssh*, por ello es mejor cambiar en */etc/passwd*, la línea del usuario predeterminada:

```
tlahuiz:x:1005:1005:Tlahuizcalpan„„:/home/tlahuiz:/bin/bash
```

a esta otra:

```
tlahuiz:x:1005:1005:Tlahuizcalpan„„:/home/tlahuiz:/usr/bin/git-
Shell
```

En la máquina de trabajo o en el servidor en cualquier carpeta se genera la estructura del repositorio en un directorio temporal de trabajo para el repositorio:

```
$ mkdir tmp
$ cd tmp
$ git init
```

Para generar la estructura de trabajo para el repositorio y los archivos necesarios:

```
$ mkdir branches release trunk
$ mkdir ...
```

Para adicionar todos y cada uno de los archivos y carpetas:

```
$ git add .
```

Para subir los cambios:

```
$ git commit -m "Texto"
```

Después debemos mandarlo al servidor:

```
$ git remote add origin ssh://usr@máquina/~/trayectoria
```

o mandarlo a un directorio local:

```
$ git remote add origin ~/trayectoria
$ git push origin +master:refs/heads/master
```

Para usar el repositorio en cualquier otra máquina hay que bajar el repositorio por primera vez del servidor:

```
$ git clone ssh://usr@máquina/~/trayectoria
```

o de una carpeta local:

```
$ git clone ~/trayectoria
```

Ahora, podemos configurar algunos datos usados en el control de cambios:

```
$ git config –global usr.name "Antonio Carrillo"
$ git config –global usr.email antoniocarrillo@ciencias.unam.mx
```

cuando se requiera actualizar del repositorio los cambios:

```
$ git pull
```

para subir los cambios al repositorio:

```
$ git commit -a -m "mensaje"
$ git push
```

Comando usados para el trabajo cotidiano en *Git* para ver el estado de los archivos locales, usamos:

```
$ git status
```

para generar una nueva rama y trabajar en ella:

```
$ git branch MiIdea
$ git checkout MiIdea
```

o en un solo paso:

```
$ git checkout -b MiIdea
```

Para unificar las ramas generadas en el punto anterior:

```
$ git checkout master
$ git merge MiIdea
```

Para borrar una rama:

```
$ git branch -d MiIdea
```

Para listar ramas:

```
$ git branch
```

Para listar ramas fusionadas:

```
$ git branch –merged
```

Para listar ramas sin fusionar:

```
$ branch –no-merged
```

Para ver los cambios en el repositorio:

```
$ git log
```

o verlos en forma acortada:

$ git log –pretty=oneline

Para recuperar un archivo de una actualización anterior:

$ git show a30ab2ca64d81876c939e16e9dac57c8db6fb103:ruta/al/archivo > ruta/al/archivo.bak

Para volver a una versión anterior:

$ git reset –hard 56f8fb550282f8dfaa75cd204d22413fa6081a11:

para regresar a la versión presente (cuidado con subir cambios en ramas anteriores):

$ git pull

Si en algún momento borramos algo o realizamos cambios en nuestra máquina y necesitamos regresar los archivos como estaban en nuestra última actualización, podemos usar:

$ git reset –hard HEAD

este trabaja con la información de nuestra copia local y no necesita conexión de red para la restitución. Eventualmente es necesario optimizar la copia local de los archivos en *Git*, para ello podemos usar:

$ git gc

Visualizador gráfico para *Git*:

apt install gitk

Git es un proyecto pujante, amplio y bien documentado, ejemplos y documentación puede ser consultada en:

- https://git-scm.com/book/es/v1
- http://git-scm.com/documentation
- https://coderwall.com/p/kucyaw/protect-secret-data-in-git-repo

Git en Google Drive:

- http://www.iexplain.org/using-git-with-google-drive-a-tutorial/
- https://techstreams.github.io/2016/09/07/google-drive-as-simple-git-Host/

Aprender a Usar *Git* En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a usar *Git*, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Git

Un Resumen de los Comando de Git más Usados

`CONFIGURACION`

Configurar Nombre que salen en los commits

git config –global user.name "antonio"

Configurar Email

git config –global user.email antonio@gmail.com

Marco de colores para los comando

git config –global color.ui true

`INICIANDO REPOSITORIO`

Iniciamos GIT en la carpeta donde está el proyecto

git init

Clonamos el repositorio de github o bitbucket

git clone <url>

Añadimos todos los archivos para el commit

git add .

Hacemos el primer commit

git commit -m "Texto que identifique por que se hizo el commit"

Subimos al repositorio

git push origin master

`GIT CLONE`

Clonamos el repositorio de github, gitlab o bitbucket

git clone <url>

Clonamos el repositorio de github, gitlab o bitbucket

git clone <url> git-demo

`GIT ADD`

Añadimos todos los archivos para el commit

git add .

Añadimos el archivo para el commit

git add <archivo>

Añadimos todos los archivos para el commit omitiendo los nuevos

git add –all

Añadimos todos los archivos con la extensión especificada

git add *.txt

Añadimos todos los archivos dentro de un directorio y de una extensión especifica

git add docs/*.txt

Añadimos todos los archivos dentro de un directorios

git add docs/

`GIT COMMIT`

Cargar en el HEAD los cambios realizados

git commit -m "Texto que identifique por que se hizo el commit"

Agregar y Cargar en el HEAD los cambios realizados

git commit -a -m "Texto que identifique por que se hizo el commit"

De haber conflictos los muestra

git commit -a

Agregar al último commit, este no se muestra como un nuevo commit en los logs. Se puede especificar un nuevo mensaje

git commit –amend -m "Texto que identifique por que se hizo el commit"

`GIT PUSH`

Subimos al repositorio

git push <origien> <branch>

Subimos un tag

git push –tags

`GIT LOG`

Muestra los logs de los commits

git log

Muestras los cambios en los commits

git log –oneline –stat

Muestra gráficos de los commits

```
git log –oneline –graph
```

`GIT DIFF`

Muestra los cambios realizados a un archivo

```
git diff
git diff –staged
```

`GIT HEAD`

Saca un archivo del commit

```
git reset HEAD <archivo>
```

Devuelve el último commit que se hizo y pone los cambios en staging

```
git reset –soft HEAD^
```

Devuelve el último commit y todos los cambios

```
git reset –hard HEAD^
```

Devuelve los 2 últimos commit y todos los cambios

```
git reset –hard HEAD^^
```

Rollback merge/commit

```
git log
git reset –hard <commit_sha>
```

`GIT REMOTE`

Agregar repositorio remoto

```
git remote add origin <url>
```

Cambiar de remote

git remote set-url origin <url>

Remover repositorio

git remote rm <name/origin>

Muestra lista repositorios

git remote -v

Muestra los branches remotos

git remote show origin

Limpiar todos los branches eliminados

git remote prune origin

`GIT BRANCH`

Crea un branch

git branch <nameBranch>

Lista los branches

git branch

Comando -d elimina el branch y lo une al master

git branch -d <nameBranch>

Elimina sin preguntar

git branch -D <nameBranch>

`GIT TAG`

Muestra una lista de todos los tags

git tag

Crea un nuevo tags

git tag -a <verison> - m "esta es la versión x"

`GIT REBASE`

Los rebase se usan cuando trabajamos con branches esto hace que los branches se pongan al día con el master sin afectar al mismo
Une el branch actual con el mastar, esto no se puede ver como un merge

git rebase

Cuando se produce un conflicto no das las siguientes opciones:
cuando resolvemos los conflictos *–continue* continua la secuencia del rebase donde se pauso

git rebase –continue

Omite el conflicto y sigue su camino

git rebase –skip

Devuelve todo al principio del rebase

git reabse –abort

Para hacer un rebase a un branch en específico

git rebase <nameBranch>

`OTROS COMANDOS`

Lista un estado actual del repositorio con lista de archivos modificados o agregados

git status

Quita del HEAD un archivo y le pone el estado de no trabajado

git checkout – <file>

Crea un branch en base a uno Online

```
git checkout -b newlocalbranchname origin/branch-name
```

Busca los cambios nuevos y actualiza el repositorio

```
git pull origin <nameBranch>
```

Cambiar de branch

```
git checkout <nameBranch/tagname>
```

Une el branch actual con el especificado

```
git merge <nameBranch>
```

Verifica cambios en el repositorio Online con el local

```
git fetch
```

Borrar un archivo del repositorio

```
git rm <archivo>
```

`Fork`

Descargar remote de un fork

```
git remote add upstream <url>
```

Merge con master de un fork

```
git fetch upstream
git merge upstream/master
```

Git-crypt El paquete *git-crypt* es una solución que usa *GPG* por debajo de *Git* que permite cifrado y descifrado transparente de archivos en un repositorio *git*.

Los archivos que se requieran proteger serán cifrados al hacer commit y descifrados al hacer *checkout* y permite compartir libremente un repo-sitorio que contenga contenido tanto público como privado. De esta forma, permite trabajar de manera transparente con el contenido descifrado, de forma que desarrolladores que no tengan la clave secreta podrán clonar y hacer commit en un repositorio con archivos cifrados.

Esto te permite almacenar tu material secreto (como pueden ser claves) en el mismo repositorio que tu código sin tener que bloquearlo. Solo un usuario autorizado puede dar permisos a otros usuarios.

Para instalar el paquete git-crypt usamos:

```
# apt install git-crypt
```

Ya instalado debemos preparar el repositorio git, para crear la llave, entonces usar:

```
$ git-crypt keygen ~/crypt-key
```

Ahora podemos crear el repositorio:

```
$ cd repo
$ git-crypt init
```

Especifica que carpetas/archivos deben ser cifrados, como git-filters:

```
$ cat .gitattributes
```

```
keys filter=git-crypt diff=git-crypt
```

crear la lista de los archivos a cifrar

```
$ vi .gitattributes
```

Indicamos que se cifren, por ejemplo, los archivos *.html, .org*, directorio:secretdir/**secreto y archivo, con cualquier extensión o palabra que le preceda.

*.html filter=git-crypt diff=git-crypt‘

*.org filter=git-crypt diff=git-crypt

directorio_secreto/** filter=git-crypt diff=git-crypt

archivo filter=git-crypt diff=git-crypt

ahora cada vez que hagamos un commit, los archivos *.html y .org*, subirán cifrados.

Ya podemos usar la llave para cifrar los archivos indicados por .gitattributes mediante:

$ git-crypt unlock ~/crypt-key

y agregar los archivos que deseamos cifrar, usando *git add*, revisando el estado de los archivos cifrados mediante:

$ git-crypt status -f

y podemos hacer los commits necesarios.

Al clonar el repositorio, los archivos cifrados se mostrarán como tal, hasta hacer en el repositorio:

$ git-crypt unlock ~/crypt-key

mostrando los archivos descifrados a partir de ese momento

Si se desea respaldar el repositorio en un solo archivo se puede usar:

$ git bundle create /tmp/Respaldo –all

y para restaurar usar algo como:

$ git clone /tmp/Respaldo newFolder

También podemos añadir usuarios autorizados (identificados por su clave *GPG*), mediante:

$ git-crypt add-gpg-user USER_ID

Flujos de trabajo comunes

- En la máquina del desarrollador: Crea el vault, añádete como usuario fiable. Pide las claves públicas a los miembros de tu equipo y añádelas al vault.
- En el entorno de Integración Continua (CI): Añade una clave *GPG* común para los ejecutores jenkins/CI. Autorízarla en el repositorio.

Seguridad

- Git-crypt usa *GPG* internamente, así que el nivel de seguridad debería ser el dado por *GPG*, a excepción de posibles errores en el propio programa *git-crypt*.
- Git-crypt es más seguro que otros sistemas git de cifrado transparente, *git-crypt* cifra archivos usando *AES-256* en modo *CTR* con un synthetic IV derivado del *SHA-1 HMAC* del archivo. Este modo de operar proporciona seguridad semántica ante *CPAs* (chosen-plain attacks) determinísticos. Esto significa que pese a que el cifrado es determinística (lo cual es requerido para que git pueda distinguir cuando un archivo ha cambiado y cuando no), no filtra información más allá de mostrar si dos archivos son idénticos o no.

Limitaciones y Trucos

- Cualquier usuario no autorizado puede ver que estamos usando *git-crypt* basándose en la evidencia dejada en el archivo .gitattributes.
- Git-crypt no cifra nombres de archivo, mensajes de *commit, symlink targets, gitlinks*, u otros metadatos.
- Git-crypt se apoya en git filters, los cuales no fueron diseñados con el cifrado en mente. Así pues, *git-crypt* no es la mejor herramienta para cifrar la mayoría o totalidad de los archivos de un repositorio. Donde *git-crypt* destaca es en aquellos casos en que la mayoría del repositorio es público pero unos pocos archivos deben ser cifrados (por ejemplo, claves privadas o archivos con credenciales API). Para cifrar un repositorio entero, es mejor considerar usar un sistema como **git-remote-gcrypt**.

- Git-crypt no esconde cuando un archivo cambia o no, cuanto ocupa o el hecho de que dos archivos sean idénticos.
- Los archivos cifrados con *git-crypt* no se pueden comprimir. Incluso el más pequeño de los cambios en un archivo cifrado requiere que git archive el archivo modificado en su totalidad y no solo un delta.
- A pesar de que *git-crypt* protege el contenido de los archivos individuales con *SHA-1 HMAC*, *git-crypt* no puede ser usado de forma segura a menos que el repositorio entero esté protegido contra la alteración de datos (un atacante que pueda mutar tu repositorio podrá alterar tu archivo *.gitattributes* para deshabilitar el cifrado). Si fuera necesario, usa características de *git* cómo signed tags en vez de contar únicamente con *git-crypt* para la integridad.
- El *diff* del *commit* varía cuando el vault está abierto vs cuando está cerrado. Cuando está abierto, los contenidos del archivo están en formato plano, es decir, descifrados. En consecuencia puedes ver el *diff*. Cuando el vault está cerrado, no se puede apreciar un *diff* efectivo ya que el texto cifrado cambia, pero el ojo humano no puede distinguir los contenidos.

Además de Git usado de forma local, existen diversos servicios en la nube[101] que permiten dar soporte a proyectos mediante *Git,* en los cuales es necesario crear una cuenta y subir los datos usando *Git,* algunos de estos servicios son:

8.7.9 GitLab vs GitHub

Aunque GitHub y GitLab tienen similitudes, incluso en el propio nombre que comienza por Git debido a que ambas se basa en la famosa herramienta de control de versiones escrita por Linus Torvalds, pero ni una ni otra son exactamente iguales. Por ello, el ganador de la batalla GitHub vs GitLab no está tan claro, tienen algunas diferencias que hacen que tengan sus ventajas y desventajas para los usuarios y desarrolladores que las suelen usar.

[101] Algunos de estos proyectos gratuitos son: Gitlab, Github, Bitbucket, Beanstalk, Launchpad, SourceForge, Phabricator, GitBucket, Gogs, Gitea, Apache Allura, entre otros.

¿Qué es GitHub? es una plataforma de desarrollo colaborativo, también llamado forja. Es decir, una plataforma enfocada hacia la cooperación entre desarrolladores para la difusión y soporte de su software (aunque poco a poco se ha ido usando para otros proyectos más allá del Software).

Como su propio nombre indica, se apoya sobre el sistema de control de versiones *Git*. Así, se puede operar sobre el código fuente de los programas y llevar un desarrollo ordenado. Además, esta plataforma está escrita en Ruby on Rails.

Tiene una enorme cantidad de proyectos de código abierto almacenada en su plataforma y accesibles de forma pública. Tal es su valor que Microsoft optó por comprar esta plataforma en 2018, aportando una cifra de nada menos que de 7,500 millones de dólares.

Pese a las dudas sobre esa compra, la plataforma continuó operando como de costumbre, y continúa siendo una de las más usadas. En ella se alojan proyectos tan importantes como el propio Kernel Linux.

¿Qué es GitLab? es otra alternativa a GitHub, otro sitio de forja con un servicio Web y un sistema de control de versiones también basado en Git. Por supuesto, se ideó para el alojamiento de proyectos de código abierto y para facilitar la vida de los desarrolladores, pero existen algunas diferencias con el anterior.

Esta Web, además de la gestión de repositorios y control de versiones, también ofrece alojamiento para Wikis, y sistema de seguimiento de errores. Una completa suite para crear y gestionar los proyectos de todo tipo, ya que, al igual que GitHub, actualmente se alojan proyectos que van más allá del código fuente.

Fue escrito por unos desarrolladores ucranianos, Dmitry Zaporozhets y Valery Sizov, usando lenguaje de programación Ruby y algunas partes en Go. Después se mejoró su arquitectura con Go, Vue.js, y Ruby on Rails, como en el caso de GitHub.

A pesar de ser muy conocida y ser la gran alternativa a GitHub, no cuenta con tantos proyectos. Eso no quiere decir que la cantidad de código alojado sea muy grande, con organizaciones que confían en ella de la talla del CERN, la NASA, IBM, Sony, etc.

Personalmente, te diría que no existe un claro ganador en la batalla GitHub vs GitLab. No es tan sencillo elegir una plataforma que sea infinitamente superior a la otra, de hecho, cada una tiene sus puntos fuertes y

sus puntos débiles. Y todo dependerá de lo que realmente busques para que te tengas que decantar por una u otra.

Diferencias GitHub vs GitLab A pesar de todas las similitudes, una de las claves a la hora de decantarse en la comparativa GitHub vs GitLab pueden ser las diferencias entre ambas:

Niveles de autenticación: GitLab puede establecer y modificar permisos a los diferentes colaboradores según su función. En el caso de GitHub, puede decidir quién tiene derechos de lectura y escritura en un repositorio, pero es más limitado en ese sentido.

- Alojamiento: aunque ambas plataformas permiten alojar el contenido de los proyectos en las propias plataformas, en el caso de GitLab también te puede permitir autoalojar tus repositorios, lo que puede ser una ventaja en algunos casos. GitHub ha agregado esa característica también, pero solo con ciertos planes de pago. GitHub ofrece máximo 3 colaboradores gratis y hasta 500 MB por repositorio, en cambio en GitLab no hay límite al número de colaboradores y hasta 10 GB por repositorio.

- Importación y exportación: GitLab contiene información muy detallada de cómo poder importar proyectos para moverlos de una plataforma a otra, como GitHub, Bitbucket, o traerlos a GitLab. Además, a la hora de exportar, GitLab ofrece un trabajo muy sólido. En el caso de GitHub no se ofrece documentación detallada, aunque se puede usar GitHub Importer como herramienta, aunque puede ser algo más restrictivo cuando se trata de exportar.

- Comunidad: ambos tienen una buena comunidad tras de sí, aunque GitHub parece haber ganado la batalla en popularidad. Actualmente aglutina millones de desarrolladores. Por eso, será más sencillo encontrar ayuda al respecto.

- Versiones Enterprise: ambos las ofrecen si pagas la cuota, por lo que se podría pensar que la comparativa GitHub vs GitLab no tiene sentido en este punto, pero lo cierto es que GitLab ofrece unas prestaciones muy interesantes, y se ha hecho popular entre los equipos de desarrollo muy grandes.

Ventajas y Desventajas de GitLab una vez conocidas las diferencias y semejanzas entre GitHub vs GitLab, las ventajas y desventajas de estas plataformas te pueden ayudar a decidirte.

Ventajas

- Plan gratuito y sin limitaciones, aunque tiene planes de pago.
- Es de licencia de código abierto.
- Permite el autohospedaje en cualquier plan.
- Está muy bien integrado con Git.

Desventajas

- Su interfaz puede ser algo más lenta con respecto a la competencia.
- Existen algunos problemas habituales con los repositorios.

Ventajas y Desventajas de GitHub Por otro lado, GitHub también tiene sus pros y contras, entre los que destacan los siguientes:

Ventajas

- Servicio gratuito, aunque también tiene servicios de pago.
- Búsqueda muy rápida en las estructura de los repos.
- Amplia comunidad y fácil encontrar ayuda.
- Ofrece prácticas herramientas de cooperación y buena integración con Git.
- Fácil integración con otros servicios de terceros.
- Trabaja también con TFS, HG y SVN.

Desventajas

- No es absolutamente abierto.
- Tiene limitaciones de espacio, ya que no puedes exceder de 100MB en un solo archivo, mientras que los repositorios están limitados a 1GB en la versión gratis.

Como ves, no existe un claro ganador. La elección no es fácil y, como comenté, deberías vigilar muy bien las ventajas, desventajas y diferencias de cada uno para poder identificar cuál se adapta mejor a tus necesidades.

Personalmente te diría que si quieres disponer de un entorno totalmente abierto, mejor usa GitLab. En cambio, si prefieres más facilidades y usar el servicio Web con más presencia, entonces ve a por GitHub. Incluso incluiría un tercero en discordia y te diría que si buscas trabajar con servicios Atlassian deberías mirar del lado de Bitbucket.

Uso GitLab (https://about.gitlab.com/)

Para configurar:

```
git config –global user.name "Antonio Carrillo Ledesma"
git config –global user.email "antoniocarrillo@ciencias.unam.mx"
```

Para crear nuevo repositorio:

```
git clone https://gitlab.com/antoniocarrillo69/MDF.git
cd MDF
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Para usar una carpeta existente:

```
cd existing_folder
git init
git remote add origin https://gitlab.com/antoniocarrillo69/MDF.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Para usar un repositorio existente:

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/antoniocarrillo69/MDF.git
git push -u origin –all
git push -u origin –tags
```

Uso Github (https://github.com/)

Para configurar:

```
git config –global user.name "Antonio Carrillo Ledesma"
git config –global user.email "antoniocarrillo@ciencias.unam.mx"
```

Para configurar un nuevo repositorio:

```
$ touch README.md
$ git init
$ git add .
$ git commit -m "mi primer commit"
$ git remote add origin https://github.com/antoniocarrillo69/ejemploPruebas.git
$ git pull origin master
$ git push origin master
```

8.7.10 Otras opciones

Herramientas para convertir código fuentes en HTML, usando:

```
$ code2html Fuente Salida.html
```

o

```
$ c2html Fuente
```

Para java, usamos:

```
$ java2html Fuentes
```

También podemos convertir código fuente en PDF, usando:

```
$ nl test.cpp | a2ps -1 -l100 -otest.ps ; ps2pdf test.ps
```

el primer comando numera las líneas del fuente, el segundo comando generá del fuente numerado un .PS y el último comando convierte .PS a .PDF

Si se tiene que ejecutar múltiples programas que son independientes uno de otro se puede usar el programa *parallel* para correr N (número de cores del equipo) de ellos al mismo tiempo, por ejemplo si tenemos un archivo Bash con el nombre *mi-Bash* y cuyo contenido es:

./a.out 4 5 4 > a1.txt

./a.out 4 5 3 > a2.txt

./a.out 4 5 6 > a3.txt

./a.out 4 5 4 > a4.txt

./a.out 3 5 4 > a5.txt

./a.out 4 6 4 > a6.txt

entonces podemos ejecutarlo usando *parallel*, el programa usará el número máximo de cores disponibles:

$ parallel -v < mi-Bash

si solo se desea usar una determinada cantidad de cores (por ejemplo 3) entonces usamos:

$ parallel -v -j 3 < mi-Bash

8.8 Programando Desde la Nube

Existen diferentes servicios Web[102] que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Algunos ejemplos de estos servicios son:

- https://www.jdoodle.com/
- https://repl.it/
- http://browxy.com
- https://jupyter.org/try
- https://tio.run/
- https://www.compilejava.net/
- http://codepad.org/
- https://code.hackerearth.com/
- https://www.remoteinterview.io/online-c-compiler
- https://ideone.com/
- https://hackide.herokuapp.com/
- https://www.codechef.com/ide
- http://cpp.sh/

[102] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de Internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

- https://codebunk.com/
- https://rextester.com/
- https://www.tutorialspoint.com/codingground.htm
- https://www.compileonline.com
- http://pythonfiddle.com/
- https://trinket.io/python
- https://www.pythonanywhere.com/try-ipython/
- https://www.rollapp.com/
- https://godbolt.org/
- https://www.codiva.io/
- https://paiza.io/en
- https://wandbox.org/
- http://coliru.stacked-crooked.com/
- http://quick-bench.com/
- https://cppinsights.io/
- https://ideone.com/
- http://cpp.sh/
- https://ide.geeksforgeeks.org/
- https://www.codechef.com/ide
- https://visualstudio.microsoft.com/services/visual-studio-online/

`Usando Editores Colaborativos`

La escritura colaborativa es una escritura de códigos de programación en la Web hecha por más de una persona simultáneamente.

Algunos ejemplos de estos servicios son:

- http://collabedit.com (edita código, tiene chat, no compila)
- https://gitduck.com/
- https://codeshare.io/
- https://www.tutorialspoint.com/codingground.htm
- http://ideone.com
- https://codebunk.com
- https://visualstudio.microsoft.com/services/visual-studio-online/
- https://ace.c9.io/build/kitchen-sink.html
- https://coderpad.io/
- https://peerpad.net/
- https://aws.amazon.com/cloud9/
- https://codeanywhere.com/
- https://stekpad.com/home/

Algunas de las terminales soportados son:

CentOS, IPython, Lua, MemCached, Mongo DB, MySQL, Node.js, Numpy, Oracle, Octave, PowerShell, PHP, R Programming, Redis, Ruby, SciPy, SymPy, etc.

Algunos de los IDEs soportados son:

Ada (GNAT), Algol68, Angular JS, Assembly, AsciiDoc, AWK, Bash Shell, Befunge, Bootstrap, Brainf**k, C, CSS3, Chipmunk BASIC, Clojure, Cobol, CoffeeScript, ColdFusion, C99 Strict, C++, C++ 0x, C++ 11, C#, Dart, D Programming Language, Embedded C, Erlang, Elixir, Factor, Fantom, Falcon, Fortran-95, Forth,F#, Free Basic, Groovy, GO, Haxe, Haskell, HTML, ilasm, Intercal, Icon, Java, Java 8, Java MySQL, JavaScript, JSP, JQuery, Julia, Korn Shell (ksh), Latex, Lisp, LOLCODE, Lua, Matlab/Octave, Malbolge, Markdown, MathML, Mozart-Oz, Nimrod, Node.JS, Objective-C, OCaml, Pascal, PARI/GP, Pawn, Perl, Perl MySQL, PHP, PHP MySQL, WebView, Pike, Processing.js, p5.js, Prolog, Python-2, Python-3, Python MySQL, Jupyter Notebook, Rexx, reStructure, Ruby, Rust, Scala, R Programming, Scheme, Smalltalk,SML/NJ, Simula, SQLite SQL, Tcl, TeX, Unlambda, VB.NET, Verilog, Whitespace, Ya Basic, etc.

Google Colaboratory Integrante de la G Suite for Education de Google permite a los usuarios que pertenezcan a esta Suite (como gran parte de los estudiantes de la UNAM) tener acceso desde el navegador para escribir y ejecutar código de Python (Jupyter), es posible elegir correr nuestro Notebook en una CPU, GPU o en una TPU de forma gratuita. Tiene algunas restricciones, como por ejemplo que una sesión dura 12 hrs, pasado ese tiempo se limpia nuestro ambiente y perdemos las variables y archivos que tengamos almacenados allí.

Es conveniente para principiantes que requieran experimentar con Machine Learning y Deep Learning pero sin recurrir en costos de procesamiento Cloud. Además el ambiente de trabajo ya viene con muchas librerías instaladas y listas para utilizar (como por ejemplo *Tensorflow, Sclkit-learn, Pytorch, Keras* y *OpenCV*), ahorrándonos el trabajo de configurar nuestro ambiente de trabajo. Podemos importar nuestros archivos y datos desde *Google Drive, GitHub*, etc.

Más información sobre Google Colaboratory en:

https://colab.research.google.com/notebooks/intro.ipynb

9 Bases de Datos

¿Qué son las bases de datos? una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior (véase [**?**] y [**?**]). En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente la mayoría de las bases de datos están en formato digital, se han desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

9.1 Clasificación

Las bases de datos se pueden clasificar de diferentes formas: acuerdo a la ubicación, de acuerdo a la forma de establecer relaciones entre los datos, según la orientación, entre otros.

De acuerdo a la ubicación

- Base de datos centralizada: Se ubica, almacena y mantiene en una sola ubicación. Eso no significa que el usuario tenga que estar en el mismo lugar para acceder.
- Base de datos distribuida: Se trata en realidad de diferentes bases de datos en distintas locaciones físicas unidas por un gestor que las hace funcionar como si fueran una sola.

De acuerdo a la forma de establecer relaciones entre los datos

- Relacionales: En este tipo de base de datos estos se organizan en forma de filas y columnas.
- Orientadas a objetos: Aquí los datos se almacenan en forma de objetos. Estos objetos se guardan en la base de datos asignándoles atributos y métodos que son los que definen qué hacer con los datos.
- Orientadas a grafos: Utiliza la teoría de grafos para almacenar, mapear y consultar las relaciones entre datos.
- NoSQL: Permite almacenar datos no estructurados o semiestructurados.

- Orientada a documentos: Es un subtipo de la anterior. En lugar de almacenar los datos en filas y columnas utiliza documentos para almacenar y recuperar los datos. Estos documentos organizan los datos utilizando estándares como JSON o XML.

Según la orientación

- OLTP: Son bases de datos orientadas al procesamiento de transacciones e incluye funciones de introducción, modificación y borrado de datos.
- OLAP: Estas bases de datos están orientadas al análisis de los datos para permitir extraer conclusiones.

Otros tipos

- Autónomas: Están basadas en la nube y utilizan el aprendizaje automático para automatizar el trabajo la base de datos, la seguridad, las copias de seguridad, las actualizaciones y otras tareas de gestión rutinarias que en las bases de datos tradicionales realiza un administrador.
- Almacén de datos: Es una base de datos enfocada en el sector corporativo que integra y depura información de varias fuentes distintas para procesarla y analizarla desde diferentes puntos de vista a gran velocidad.

Hay programas denominados sistemas gestores de bases de datos SGBD (Database Management System o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Actualmente se tienen dos importantes tipos de bases de datos:

- Structured Query Language SQL (lenguaje de consulta estructurada) es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
- NoSQL "no solo SQL" es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas.

9.2 ¿Qué son las bases de datos SQL?

el lenguaje de consulta estructurada SQL (Structured Query Language), es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

Originalmente basado en el álgebra relacional y en el cálculo relacional, SQL consiste en un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de control de datos. El alcance de SQL incluye la inserción de datos, consultas, actualizaciones y borrado, la creación y modificación de esquemas y el control de acceso a los datos. También en SQL a veces se describe como un lenguaje declarativo, también incluye elementos procesales.

Características generales de SQL SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros –y no a registros individuales– permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros. SQL también tiene las siguientes características:

- Lenguaje de definición de datos: El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.
- Lenguaje interactivo de manipulación de datos: El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.
- Integridad: El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.

- Definición de vistas: El LDD incluye comandos para definir las vistas.
- Control de transacciones: SQL tiene comandos para especificar el comienzo y el final de una transacción.
- SQL incorporado y dinámico: Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, Python, PHP, COBOL, Pascal y Fortran.
- Autorización: El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

Tipos de datos Algunos de los tipos de datos básicos de SQL son:

- BINARY (1 Byte), para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
- BIT (1 Byte), valores Si/No ó True/False
- BYTE (1 Byte), un valor entero entre 0 y 255.
- COUNTER (4 Bytes), un número incrementado automáticamente (de tipo Long)
- CURRENCY (8 Bytes), un entero escalable entre -922,337,203,685,477.5808 y 922,337,203,685,477.5807.
- DATETIME (8 Bytes), un valor de fecha u hora entre los años 100 y 9999.
- SINGLE (4 Bytes), un valor en punto flotante de precisión simple con un rango de $-3.402823*10^{38}$ a $-1.401298*10^{-45}$ para valores negativos, $1.401298 * 10^{-45}$ a $3.402823 * 10^{38}$ para valores positivos, y 0.
- DOUBLE (8 Bytes), un valor en punto flotante de doble precisión con un rango de $-1.79769313486232 * 10^{308}$ a $-4.94065645841247 * 10^{-324}$ para valores negativos, $4.94065645841247 * 10^{-324}$ a $1.79769313486232 * 10^{308}$ para valores positivos, y 0.
- SHORT (2 Bytes), un entero corto entre -32,768 y 32,767.
- LONG (4 Bytes), un entero largo entre -2,147,483,648 y 2,147,483,647.

- LONGTEXT (1 Byte por carácter), de cero a un máximo de 1.2 gigabytes.
- LONGBINARY (según se necesite), de cero 1 gigabyte. Utilizado para objetos OLE.
- TEXT (1 Byte por carácter), de cero a 255 caracteres.

También se tienen definidos múltiples sinónimos de los tipos de datos definidos:

- BINARY: VARBINARY
- BIT: BOOLEAN, LOGICAL, LOGICAL1, YESNO
- BYTE: INTEGER1
- COUNTER: AUTOINCREMENT
- CURRENCY: MONEY
- DATETIME: DATE, TIME, TIMESTAMP
- SINGLE: FLOAT4, IEEESINGLE, REAL
- DOUBLE: FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC
- SHORT: INTEGER2, SMALLINT
- LONG: INT, INTEGER, INTEGER4
- LONGBINARY: GENERAL, OLEOBJECT
- LONGTEXT: LONGCHAR, MEMO, NOTE
- TEXT: ALPHANUMERIC, CHAR - CHARACTER, STRING - VARCHAR

SQL no requiere que escribamos todas las palabras clave en mayúscula, pero por convención ayuda a las personas a distinguir las palabras clave de SQL de los nombres de las columnas y las tablas.

Optimización Como ya se dijo antes, y suele ser común en los lenguajes de acceso a bases de datos de alto nivel, SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.

El orden de ejecución interno de una sentencia puede afectar seriamente a la eficiencia del SGBD, por lo que se hace necesario que este lleve a cabo una optimización antes de su ejecución. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos. Dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.

Los sistemas de bases de datos modernos poseen un componente llamado optimizador de consultas. Este realiza un análisis detallado de los posibles planes de ejecución de una consulta SQL y elige aquel que sea más eficiente para llevar adelante la misma.

Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa.

9.2.1 Lenguaje de definición de datos (DDL)

El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

CREATE (Crear) este comando permite crear objetos de datos, como nuevas bases de datos, tablas, vistas y procedimientos almacenados.

Ejemplos (crear una tabla):

```
CREATE TABLE clientes;
CREATE TABLE personas(
    IDPersonas INT NOT NULL,
    Nombre VARCHAR(50) NOT NULL,
    Apellido VARCHAR(60) NOT NULL,
UNIQUE(IDPersonas)
);
CREATE TABLE pedidos(
```

```
    OrdenID INT NOT NULL,
    NumeroOrden INT NOT NULL,
    IDPersona INT,
    PRIMARY KEY(OrdenID),
    FOREING KEY(IDPersona) REFERENCES personas(IDPersonas)
);
```

ALTER (Alterar) este comando permite modificar la estructura de una tabla u objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un *trigger*, etc.

Ejemplo (agregar columna a una tabla):

```
ALTER TABLE personas ADD edad INT UNSIGNED;
```

DROP (Eliminar) este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo (elimina la tabla personas):

```
DROP TABLE personas;
```

TRUNCATE (Truncar) este comando solo aplica a tablas y su función es borrar el contenido completo de la tabla especificada. La ventaja sobre el comando DELETE, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande. La desventaja es que TRUNCATE solo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Mani-pulación de Datos), es en realidad una DDL, ya que internamente, el comando TRUNCATE borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

Ejemplo (borra el contenido de la tabla personas):

```
TRUNCATE TABLE personas;
```

CREATE INDEX permite crear índices (ya sea únicos o permite valores duplicados) en una o más columnas.

Ejemplo (crea un índice admitiendo valores duplicados en la columna)

```
CREATE INDEX indicePersonas ON personas(personas);
```

Ejemplo (crear un índice sin que puedan existir valores duplicados)

```
CREATE UNIQUE INDEX indicePersonas ON personas(personas);
```

9.2.2 Lenguaje de manipulación de datos DML

Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

El lenguaje de manipulación de datos más popular hoy en día es SQL, es usado para recuperar y manipular datos en una base de datos relacional.

SELECT (Seleccionar) La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.

Forma básica:

```
SELECT [{ALL|DISTINCT}]
<nombre_campo>[, <nombre_campo>...]
FROM {<nombre_tabla>|<nombre_vista>}[,
{<nombre_tabla>|<nombre_vista>}...]
[WHERE <condición> [{AND|OR} <condición>...]]
[GROUP BY <nombre_campo>[, <nombre_campo>...]]
[HAVING <condición> [{AND|OR} <condición>...]]
[ORDER BY {<nombre_campo>|<indice_campo>}
[{ASC|DESC}][,{<nombre_campo>|<indice_campo>}
[{ASC|DESC}]]];
```

SELECT palabra clave que indica que la sentencia de SQL que queremos ejecutar es de selección.

ALL indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca.

DISTINCT indica que queremos seleccionar solo los valores distintos.

FROM indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o "join". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR.

GROUP BY especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

HAVING especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

ORDER BY presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

Ejemplo:

Para formular una consulta a la tabla coches, usamos la palabra clave FROM y con SELECT le indicamos que deseamos recuperar los campos *matrícula, marca, modelo, color, número_kilómetros, num_plazas*, para ello debemos ejecutar la siguiente consulta:

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY
    marca, modelo;
```

los datos serán devueltos ordenados por marca y por modelo en orden ascendente, de menor a mayor.

Ejemplo de consulta simplificada a través de un comodín de campos (*): El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla y los datos serán devueltos ordenados por marca y por modelo.

```
SELECT *
FROM
    coches
ORDER BY
    marca, modelo;
```

Ejemplo de consulta simplificada a través de un comodín de campos (*): El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla y los datos serán devueltos ordenados por marca en forma descendente y por modelo en forma ascendente.

```
SELECT *
FROM
    coches
ORDER BY
    marca DESC, modelo ASC;
```

Cláusula WHERE (Donde) la cláusula WHERE es la instrucción que nos permite filtrar el resultado de una sentencia SELECT. Habitualmente no deseamos obtener toda la información existente en la tabla, sino que queremos obtener sólo la información que nos resulte útil en ese momento. La cláusula WHERE filtra los datos antes de ser devueltos por la consulta. Cuando en la cláusula WHERE queremos incluir un tipo texto, debemos incluir el valor entre comillas simples.

Ejemplos:

En nuestro ejemplo, se desea consultar un coche en concreto, para esto se agregó una cláusula WHERE. Esta cláusula especifica una o varias condiciones que deben cumplirse para que la sentencia SELECT devuelva los datos. En este caso la consulta devolverá sólo los datos del coche con matrícula para que la consulta devuelva sólo los datos del coche con matrícula MF-234-ZD o bien la matrícula FK-938-ZL . Se puede utilizar la cláusula WHERE solamente, ó en combinación con tantas condiciones como queramos.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
WHERE
matricula = 'MF-234-ZD' OR matricula = 'FK-938-ZL';
```

Una condición WHERE puede ser negada a través del Operador Lógico NOT. La siguiente consulta devolverá todos los datos de la tabla coches, menos el que tenga la matrícula MF-234-ZD.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
WHERE
    NOT matricula = 'MF-234-ZD';
```

La siguiente consulta utiliza la condicional DISTINCT, la cual nos devolverá todos los valores distintos formados por los campos *marca* y *modelo* de la tabla coches.

```
SELECT DISTINCT marca, modelo FROM coches;
```

Cláusula ORDER BY (Ordernar Por) la cláusula ORDER BY es la instrucción que nos permite especificar el orden en el que serán devueltos los datos. Podemos especificar el orden de forma ascendente o descendente a través de las palabras clave ASC y DESC. El orden depende del tipo de datos que estén definidos en la columna, de forma que un campo numérico será ordenado como tal, y un alfanumérico se ordenará de la A a la Z, aunque su contenido sea numérico. El valor predeterminado es ASC si no se especifica al hacer la consulta.

Ejemplos:

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY
    marca ASC, modelo DESC;
```

Este ejemplo, selecciona todos los campos *matricula, marca, modelo, color, numero_kilometros y num_plazas* de la tabla coches, ordenándolos por los campos *marca* y *modelo*, *marca* en forma ascendente y *modelo* en forma descendente.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY 2;
```

Este ejemplo, selecciona todos los campos *matrícula, marca, modelo, color, numero_kilometros y num_plazas* de la tabla coches, ordenándolos por el campo *marca*, ya que aparece en segundo lugar dentro de la lista de campos que componen la cláusula SELECT.

Subconsultas una subconsulta es una sentencia SELECT que está embebida en una cláusula de otra sentencia SQL. También pueden utilizarse subconsultas en los comandos INSERT, UPDATE, DELETE y en la cláusula FROM.

Las subconsultas pueden resultar útiles si necesitas seleccionar filas de una tabla con una condición que depende de los datos de la propia tabla o de otra tabla.

La subconsulta (consulta interna), se ejecuta antes de la consulta principal; el resultado de la subconsulta es utilizado por la consulta principal (consulta externa).

```
SELECT
    c.matricula, c.modelo
FROM
    coches AS c
WHERE c.matricula IN
(
    SELECT m.matricula
    FROM multas AS m
    WHERE m.importe > 100
);
```

En este ejemplo, se seleccionan las matrículas y los modelos de los coches cuyas multas superan los USD 100.

INSERT (Insertar) una sentencia INSERT de SQL agrega uno o más registros a una (y solo una) tabla en una base de datos relacional.

Forma básica:

```
INSERT INTO
tablatura(columnaA, [columnaB, ...])
VALUES
('valor1', ['valor2', ...]);
```

o también se puede utilizar como:

```
INSERT INTO tablatura VALUES ('valor1', 'valor2');
```

Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo:

```
INSERT INTO agenda_telefonica (nombre, numero)
VALUES ('Roberto Jeldrez', 4886850);
```

Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada:

```
INSERT INTO nombre_tabla VALUES ('valor1', ['valor2',
...]);
```

Ejemplo (asumiendo que 'nombre' y 'número' son las únicas columnas de la tabla 'agenda_telefonica'):

```
INSERT INTO agenda_telefonica
VALUES ('Jhonny Aguilar', 080473968);
```

Formas avanzadas una característica de SQL (desde SQL-92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

```
INSERT INTO
tabla(columna1[, columna2, ...])
VALUES
('valor1A', ['valor1B', ...]),
('value2A', ['value2B', ...]), ...;
```

Esta característica es soportada por DB2, PostgreSQL (desde la versión 8.2), MySQL, y H2.

Ejemplo (asumiendo que nombre y número son las únicas columnas en la tabla agenda_telefonica):

```
INSERT INTO
    agenda_telefonica
VALUES
    ('Roberto Fernández', '4886850'),
    ('Alejandro Sosa', '4556550');
```

Que podía haber sido realizado por las sentencias:

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850');
INSERT INTO agenda_telefonica VALUES ('Alejandro Sosa', '4556550');
```

Notar que las sentencias separadas pueden tener semántica diferente (especialmente con respecto a los triggers), y puede tener diferente rendimiento que la sentencia de inserción múltiple.

Para insertar varias filas en MS SQL puede utilizar esta construcción:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212'
UNION ALL
SELECT 'Peter Doe', '555-2323';
```

Tenga en cuenta que no se trata de una sentencia SQL válida de acuerdo con el estándar SQL (SQL: 2003), debido a la cláusula subselect incompleta.

Para hacer lo mismo en Oracle se usa la Tabla DUAL, siempre que se trate de solo una fila simple:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM DUAL
UNION ALL
SELECT 'Peter Doe','555-2323' FROM DUAL
```

Una implementación conforme al estándar de esta lógica se muestra en el siguiente ejemplo, o como se muestra arriba (no aplica en Oracle):

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM LATERAL ( VALUES (1) ) AS t(c)
UNION ALL
SELECT 'Peter Doe','555-2323' FROM LATERAL ( VALUES (1) ) AS t(c)
```

Copia de filas de otras tablas un INSERT también puede utilizarse para recuperar datos de otros, modificarla si es necesario e insertarla directamente en la tabla. Todo esto se hace en una sola sentencia SQL que no implica ningún procesamiento intermedio en la aplicación cliente. Un SUBSELECT se utiliza en lugar de la cláusula VALUES. El SUBSELECT puede contener la sentencia JOIN, llamadas a funciones, y puede incluso consultar en la misma TABLA los datos que se insertan. Lógicamente, el SELECT se evalúa antes de que la operación INSERT esté iniciada. Un ejemplo se da a continuación:

```
INSERT INTO phone_book2
SELECT *
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe');
```

Una variación es necesaria cuando algunos de los datos de la tabla fuente se está insertando en la nueva tabla, pero no todo el registro. (O cuando los esquemas de las tablas no son iguales.)

```
INSERT INTO phone_book2 ([name], [phoneNumber])
SELECT [name], [phoneNumber]
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe');
```

El SELECT produce una tabla (temporal), y el esquema de la tabla temporal debe coincidir con el esquema de la tabla donde los datos son insertados.

UPDATE (Actualizar) una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.
Ejemplo:

```
UPDATE My_table SET field1 = 'updated value' WHERE
field2 = 'N';
```

DELETE (Borrar) una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.
Forma básica:
DELETE FROM tabla WHERE columna1 = 'valor1';
Ejemplo:

```
DELETE FROM mi_tabla WHERE columna2 = 'N';
```

Funciones Disponemos de varias funciones que podemos usar, por ejemplo la función MIN(), devuelve el valor más pequeño de la columna seleccionada:

```
SELECT MIN(columna)
FROM tabla
WHERE condición;
```

La función MAX() devuelve el valor más grande de la columna seleccionada:

```
SELECT MAX(columna)
FROM tabla
WHERE condición;
```

La función COUNT() devuelve el número de filas de la consulta, es decir, el número de registros que cumplen una determinada condición:

```
SELECT COUNT(columna)
FROM tabla
WHERE condición;
```

La función AVG() devuelve el valor promedio de una columna especifica (numérica):

```
SELECT AVG(columna)
FROM tabla
WHERE condición;
```

La función SUM() devuelve la suma total de una columna especifica (numérica):

```
SELECT SUM(columna)
FROM tabla
WHERE condición;
```

Caracteres Comodín Los comodines son caracteres especiales que se utilizan con las palabras clave LIKE y NOT LIKE. Esto permite buscar datos con patrones sofisticados de manera bastante eficiente.

- % Equivale a cero o más caracteres

```
SELECT * FROM clientes
WHERE nombre LIKE '%ory';
```

- _ Equivale a cualquier carácter individual

```
SELECT * FROM clientes
WHERE ciudad LIKE '___vale';
```

- [listacaracteres] Equivale a cualquier carácter en la lista

```
SELECT * clientes
WHERE nombre LIKE '[JKT]%';
```

Claves En las bases de datos relacionales, existe un concepto de claves primarias y externas . En las tablas SQL, estos se incluyen como restricciones, donde una tabla puede tener una clave principal, una clave externa o ambas.

Claves Primarias una clave primaria permite que cada registro de una tabla se identifique de forma única. Solo puede tener una clave principal por tabla y puede asignar esta restricción a cualquier columna única o combinación. Sin embargo, esto significa que cada valor dentro de esta (s) columna (s) debe ser único.

Por lo general, en una tabla, la columna de ID es una clave principal y, por lo general, está emparejada con la palabra clave AUTO_INCREMENT. Esto significa que el valor aumenta automáticamente a medida que se crean nuevos registros.

Ejemplo: Cree una nueva tabla y establezca la clave principal en la columna ID.

```
CREATE TABLE clientes (
    id int NOT NULL AUTO_INCREMENT,
    Nombre varchar(255),
    Apellido varchar(255) NOT NULL,
    Direccion varchar(255),
    Email varchar(255),
    PRIMARY KEY (id)
);
```

Claves Externas se puede aplicar una clave externa a una columna o a muchas. Se usa para vincular 2 tablas juntas en una base de datos relacional.

- La tabla que contiene la clave externa se llama clave secundaria ,
- La tabla que contiene la clave a la que se hace referencia (o candidata) se denomina tabla principal .

Básicamente, esto significa que los datos de la columna se comparten entre 2 tablas, porque una clave externa también evita que se inserten datos no válidos que no están presentes en la tabla principal.

Ejemplo: Cree una nueva tabla y convierta cualquier columna que haga referencia a ID en otras tablas en claves externas.

```
CREATE TABLE ordenes (
id int NOT NULL,
usuario_id int,
```

```
producto_id int,
PRIMARY KEY (id),
FOREIGN KEY (usuario_id) REFERENCES users(id),
FOREIGN KEY (producto_id) REFERENCES products(id)
);
```

Índices Los índices son atributos que se pueden asignar a columnas en las que se buscan con frecuencia para que la recuperación de datos sea un proceso más rápido y eficiente.

```
CREATE INDEX idx_test
ON users (first_name, surname);
CREATE UNIQUE INDEX idx_test
ON users (first_name, surname);
DROP INDEX
ALTER TABLE users
DROP INDEX idx_test;
```

Uniones En SQL, JOINse utiliza una cláusula para devolver un resultado que combina datos de varias tablas, basándose en una columna común que aparece en ambas.

Hay varias combinaciones diferentes disponibles para su uso:

- Inner Join (predeterminado): Devuelve cualquier registro que tenga valores coincidentes en ambas tablas.
- Left Join: Devuelve todos los registros de la primera tabla, junto con los registros coincidentes de la segunda tabla.
- Right Join: Devuelve todos los registros de la segunda tabla, junto con los registros coincidentes de la primera.
- Full Join: Devuelve todos los registros de ambas tablas cuando hay una coincidencia.

```
SELECT orders.id, users.FirstName, users.Surname, products.name
as 'product name'
FROM orders
INNER JOIN users on orders.user_id = users.id
INNER JOIN products on orders.product_id = products.id;
```

Vistas en SQL Una vista es esencialmente un conjunto de resultados de SQL que se almacena en la base de datos bajo una etiqueta, por lo que puede volver a él más tarde sin tener que volver a ejecutar la consulta. Estos son especialmente útiles cuando tiene una consulta SQL costosa que puede necesitar varias veces. Entonces, en lugar de ejecutarlo una y otra vez para generar el mismo conjunto de resultados, puede hacerlo una vez y guardarlo como una vista.

Cómo crear vistas para crear una vista, puede hacerlo así:

```
CREATE VIEW priority_users AS
SELECT * FROM users
WHERE country = 'United Kingdom';
```

Luego, en el futuro, si necesita acceder al conjunto de resultados almacenado, puede hacerlo así:

```
SELECT * FROM [priority_users];
```

Cómo reemplazar vistas con el CREATE OR REPLACE comando, podemos actualizar una vista como esta:

```
CREATE OR REPLACE VIEW [priority_users] AS
SELECT * FROM users
WHERE country = 'United Kingdom' OR country='USA';
```

Cómo eliminar vistas para eliminar una vista, simplemente usamos DROP VIEW comando.

```
DROP VIEW priority_users;
```

Recuperación de Clave Los diseñadores de base de datos que usan una clave suplente como la clave principal para cada tabla, se ejecutará en el escenario ocasional en el que es necesario recuperar automáticamente la base de datos, generando una clave primaria de una sentencia SQL INSERT para su uso en otras sentencias SQL. La mayoría de los sistemas no permiten sentencias SQL INSERT para retornar la fila de datos. Por lo tanto, se hace necesario aplicar una solución en tales escenarios.

Implementaciones comunes incluyen:

- Utilizando un procedimiento almacenado específico de base de datos que genera la clave suplente, realice la operación INSERT y finalmente devuelva la clave generada.

- Utilizando una sentencia SELECT específica de base de datos, sobre una tabla temporal que contiene la última fila insertada. DB2 implementa esta característica de la siguiente manera:

```
SELECT *
FROM NEW TABLE (
INSERT INTO phone_book
VALUES ('Cristobal Jeldrez','0426.817.10.30')
) AS t
```

- Utilizando una sentencia SELECT después de la sentencia INSERT con función específica de base de datos, que devuelve la clave primaria generada por el registro insertado más recientemente.

- Utilizando una combinación única de elementos del original SQL INSERT en una sentencia posterior SELECT.

- Utilizando un GUID en la sentencia SQL INSERT y la recupera en una sentencia SELECT.

- Utilizando la función de PHP mysql_insert_id() de MySQL después de la sentencia INSERT.

- Utilizando un INSERT con la cláusula RETURNING para Oracle, que solo se puede utilizar dentro de un bloque PL/SQL, en el caso de PostgreSQL se puede usar tanto con SQL como con PL/SQL.

```
INSERT INTO phone_book VALUES ('Cristobal Jeldrez',
'0426.817.10.30')
RETURNING phone_book_id INTO v_pb_id
```

- En el caso de MS SQL se puede utilizar la siguiente instrucción:

```
Set NoCount On;
INSERT INTO phone_book VALUES ('Cristobal Jeldrez',
'0426.817.10.30');
Select @@Identity as id
```

9.3 ¿Qué son las bases de datos NoSQL?

A veces llamado "no solo SQL" es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no solo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

Por lo general, los investigadores académicos se refieren a este tipo de bases de datos como almacenamiento estructurado, término que abarca también las bases de datos relacionales clásicas. A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, las implementaciones de BigTable, bases de datos documentales, y bases de datos orientadas a grafos.

Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala. Bastantes sistemas NoSQL emplean una arquitectura distribuida, manteniendo los datos de forma redundante en varios servidores, usando frecuentemente una tabla Hash distribuida. De esta forma, el sistema puede realmente escalar añadiendo más servidores, y el fallo en un servidor puede ser tolerado.

¿Cómo funciona una base de datos NoSQL (no relacionales)? Las bases de datos NoSQL utilizan una variedad de modelos de datos para acceder y administrar datos. Estos tipos de bases de datos están optimizados específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

Considere el ejemplo de modelado del esquema para una base de datos simple de libros:

- En una base de datos relacional, un registro de libros a menudo se enmascara (o "normaliza") y se almacena en tablas separadas, y las relaciones se definen mediante restricciones de claves primarias y externas. En este ejemplo, la tabla *Libros* tiene las columnas *ISBN, Título*

del libro y Número de edición, la tabla *Autores* tiene las columnas *IDAutor y Nombre de autor* y, finalmente, la tabla *Autor-ISBN* tiene las columnas *IDAutor e ISBN*. El modelo relacional está diseñado para permitir que la base de datos aplique la integridad referencial entre tablas en la base de datos, normalizada para reducir la redundancia y, generalmente, está optimizada para el almacenamiento.

- En una base de datos NoSQL, el registro de un libro generalmente se almacena como un documento JSON. Para cada libro, el elemento *ISBN, Título del libro, Número de edición, Nombre autor y IDAutor* se almacenan como atributos en un solo documento. En este modelo, los datos están optimizados para un desarrollo intuitivo y escalabilidad horizontal.

¿Por qué debería usar una base de datos NoSQL? Las bases de datos NoSQL se adaptan perfectamente a muchas aplicaciones modernas, como dispositivos móviles, Web y juegos, que requieren bases de datos flexibles, escalables, de alto rendimiento y altamente funcionales para proporcionar excelentes experiencias de usuario.

- Flexibilidad: las bases de datos NoSQL generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para datos semiestructurados y no estructurados.

- Escalabilidad: las bases de datos NoSQL generalmente están diseñadas para escalar usando clústeres distribuidos de Hardware en lugar de escalar añadiendo servidores caros y sólidos. Algunos proveedores de la nube manejan estas operaciones en segundo plano, como un servicio completamente administrado.

- Alto rendimiento: la base de datos NoSQL está optimizada para modelos de datos específicos y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.

- Altamente funcional: las bases de datos NoSQL proporcionan APIs altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

Algunos ejemplos de bases de datos NoSQL son: DynamoDB, Cassandra, Redis, CouchDB, RethinkDB, RavenDB, Elasticsearch y MongoDB.

Tipos de bases de datos NoSQL

- Clave-valor: las bases de datos clave-valor son altamente divisibles y permiten escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar. Los casos de uso como juegos, tecnología publicitaria e IoT se prestan particularmente bien con el modelo de datos clave-valor. Amazon DynamoDB está diseñado para proporcionar una latencia de milisegundos constante de un solo dígito para cualquier escala de cargas de trabajo. Este rendimiento sistemático es uno de los principales elementos que explican por qué la característica de historias de Snapchat, que incluye la carga de trabajo de escritura de almacenamiento más grande de Snapchat, se trasladó a DynamoDB.

- Documentos: en el código de aplicación, los datos se representan a menudo como un objeto o un documento de tipo JSON porque es un modelo de datos eficiente e intuitivo para los desarrolladores. Las bases de datos de documentos facilitan a los desarrolladores el almacenamiento y la consulta de datos en una base de datos mediante el uso del mismo formato de modelo de documento que emplean en el código de aplicación. La naturaleza flexible, semiestructurada y jerárquica de los documentos y las bases de datos de documentos permiten que evolucionen según las necesidades de las aplicaciones. El modelo de documentos funciona bien con catálogos, perfiles de usuario y sistemas de administración de contenido en los que cada documento es único y evoluciona con el tiempo. Amazon DocumentDB (con compatibilidad para MongoDB) y MongoDB son bases de datos de documentos conocidas que proporcionan APIs poderosas e intuitivas para un desarrollo flexible e iterativo.

- Gráficos: el propósito de una base de datos de gráficos es facilitar la creación y la ejecución de aplicaciones que funcionan con conjuntos de datos altamente conectados. Los casos de uso típicos para una base de datos de gráficos incluyen redes sociales, motores de recomendaciones, detección de fraude y gráficos de conocimiento. Amazon Neptune es un servicio de base de datos de gráficos completamente administrado. Neptune admite tanto el modelo de Property Graph como el Resource

Description Framework (RDF), que ofrece la opción de dos API de gráficos: TinkerPop y RDF/SPARQL. Las bases de datos de gráficos populares incluyen Neo4j y Giraph.

- En memoria: las aplicaciones de juegos y tecnología publicitaria tienen casos de uso como tablas de clasificación, tiendas de sesión y análisis en tiempo real que requieren tiempos de respuesta de microsegundos y pueden tener grandes picos de tráfico en cualquier momento. Amazon ElastiCache ofrece MemCachéd y Redis, para servir cargas de trabajo de baja latencia y alto rendimiento, como McDonald's, en las que no se pueden servirse con almacenes de datos basados en disco. Amazon DynamoDB Accelerator (DAX) es otro ejemplo de un almacén de datos especialmente diseñado. DAX hace que DynamoDB lea una orden de magnitud más rápida.

- Buscar: muchas aplicaciones generan registros para ayudar a los desarrolladores a solucionar problemas. Amazon Elasticsearch Service (Amazon ES) está diseñado para proporcionar visualizaciones en tiempo real y análisis de datos generados por máquinas al indexar, agregar y buscar registros y métricas semiestructuradas. Amazon ES también es un poderoso motor de búsqueda de alto rendimiento para casos de uso de búsqueda de texto completo. Expedia está utilizando más de 150 dominios de Amazon ES, 30 TB de datos y 30 mil millones de documentos para una variedad de casos de uso críticos, que van desde el monitoreo operativo y la resolución de problemas, hasta el seguimiento de la pila de aplicaciones distribuidas y la optimización de precios.

9.3.1 SQL en comparación con NoSQL

Durante décadas, el modelo de datos predominante utilizado para el desarrollo de aplicaciones era el modelo de datos relacional empleado por bases de datos relacionales como Oracle, DB2, SQL Server, MySQL y PostgreSQL. No fue sino hasta mediados y finales de la década del 2000 que otros modelos de datos comenzaron a adoptarse y aumentó su uso significativamente. Para diferenciar y categorizar estas nuevas clases de bases de datos y modelos de datos, se acuñó el término "NoSQL". Con frecuencia, los términos "NoSQL" y "no relacional" se usan indistintamente.

Cargas de trabajo óptimas las bases de datos relacionales están diseñadas para aplicaciones de procesamiento de transacciones online (OLTP) altamente coherentes y transaccionales, y son buenas para el procesamiento analítico online (OLAP), en cambio las bases de datos NoSQL están diseñadas para varios patrones de acceso a datos que incluyen aplicaciones de baja latencia. Las bases de datos de búsqueda NoSQL están diseñadas para hacer análisis sobre datos semiestructurados.

Modelo de datos el modelo relacional normaliza los datos en tablas conformadas por filas y columnas. Un esquema define estrictamente las tablas, las filas, las columnas, los índices, las relaciones entre las tablas y otros elementos de las bases de datos. La base de datos impone la integridad referencial en las relaciones entre tablas, en cambio las bases de datos NoSQL proporcionan una variedad de modelos de datos, como clave-valor, documentos y gráficos, que están optimizados para el rendimiento y la escala.

Propiedades ACID las bases de datos relacionales ofrecen propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID):

- La atomicidad requiere que una transacción se ejecute por completo o no se ejecute en absoluto.
- La coherencia requiere que una vez confirmada una transacción, los datos deben acoplarse al esquema de la base de datos.
- El aislamiento requiere que las transacciones simultáneas se ejecuten por separado.
- La durabilidad requiere la capacidad de recuperarse de un error inesperado del sistema o de un corte de energía y volver al último estado conocido.

Las bases de datos NoSQL a menudo hacen concesiones al flexibilizar algunas de las propiedades ACID de las bases de datos relacionales para un modelo de datos más flexible que puede escalar horizontalmente. Esto hace que las bases de datos NoSQL sean una excelente opción para casos de uso de baja latencia y alto rendimiento que necesitan escalar horizontalmente más allá de las limitaciones de una sola instancia.

Rendimiento en las bases de datos relacionales normalmente, el rendimiento depende del subsistema de disco. Se necesita la optimización de consultas, índices y estructura de tabla para lograr el máximo rendimiento, en cambio en las bases de datos NoSQL el rendimiento depende del tamaño del clúster de Hardware subyacente, la latencia de red y la aplicación que efectúa la llamada.

Escalado las bases de datos relacionales generalmente escalan en forma ascendente las capacidades de computación del Hardware o la ampliación mediante la adición de réplicas para cargas de trabajo de solo lectura, en cambio en las bases de datos NoSQL normalmente se pueden particionar porque los patrones de acceso son escalables mediante el uso de arquitectura distribuida para aumentar el rendimiento que proporciona un rendimiento constante a una escala casi ilimitada.

API en las bases de datos relacionales se solicita almacenar y recuperar datos que están comunicados mediante consultas que se ajustan a un lenguaje de consulta estructurado (SQL). Estas consultas son analizadas y ejecutadas por la base de datos relacional, en cambio en las bases de datos NoSQL las APIs basadas en objetos permiten a los desarrolladores almacenar y recuperar fácilmente estructuras de datos. Las claves de partición permiten que las aplicaciones busquen pares de clave-valor, conjuntos de columnas o documentos semiestructurados que contengan atributos y objetos de aplicación serializados.

En los últimos años, las bases de datos SQL y NoSQL incluso han comenzado a fusionarse. Por ejemplo, los sistemas de bases de datos, como PostgreSQL, MySQL y Microsoft SQL Server ahora admiten el almacenamiento y la consulta de datos JSON, al igual que las bases de datos NoSQL. Con esto, ahora puede lograr muchos de los mismos resultados con ambas tecnologías. Pero aún no obtiene muchas de las funciones NoSQL, como el escalado horizontal y la interfaz fácil de usar.

9.3.2 SQL en comparación con Terminología NoSQL

La siguiente tabla compara la terminología utilizada por las bases de datos NoSQL seleccionadas con la terminología utilizada por las bases de datos SQL.

SQL	MongoDB	DynamoDB	Cassandra	Couchbase
Tabla	Conjunto	Tabla	Tabla	Bucketdedatos
Fila	Documento	Elemento	Fila	Documento
Columna	Campo	Atributo	Columna	Campo
Claveprincipal	ObjectId	Clave principal	Clave principal	ID del documento
Índice	Índice	Índice secundario	Índice	Índice
Ver	Ver	Índice secundario global	Vista materializada	Ver
Tabla u objeto anidado	Documento incrustado	Mapa	Mapa	Mapa
Matriz	Matriz	Lista	Lista	Lista

Un conjunto de *Campos* formarán un *Documento*, que en caso de asociarse con otros formará una *Colección*. Las *Bases de Datos* estarán formadas por *Colecciones*, y a su vez, cada *Servidor* puede tener tantas bases de datos como el equipo lo permita.

Los tipos de datos soportados son: Numéricos:

- Double: punto flotante de 64 bits.
- Decimal: punto flotante de 128 bits.
- Int: enteros de hasta 32 bits.
- Long: enteros de hasta 64 bits.

Texto:

- String: Cadenas UTF-8.
- Regex: Almacena expresiones regulares.

Fecha:

- Date: entero de 64 bits representa en número de milisegundos desde el primero de enero de 1970.
- Timestamp: entero de 64 bits, en el que los primeros 32 bits representan los segundos pasados desde el primero de enero de 1970 y los otros 32 bits son ordinales incrementales.

Especiales:

- Array: almacena un conjunto de elementos de cualquier tipo.
- Objectld: tipo de dato único, principalmente utilizados para dar valor al campo _id de los documentos.
- Javascript: código de Javascript
- Null: valor nulo
- Boolean: valor booleano.

Consulta y modificación de datos Los datos pueden ser accedidos mediante operaciones CRUD (Create, Read, Update, Delete), la ejecución de cualquiera de estas instrucciones sigue el siguiente esquema:

db.<nombre_colección>.<nombre_Operación>(<condiciones_de_la_operación>)

Operaciones:

- Find: lectura (Read) de datos
- Insert: escritura (Create) de datos
- Update: actualización de valores de datos
- Remove: borrado (Delete) de datos

A modo de ejemplo utilizaremos un JSON como el que se usa en mongoDB. Supongamos que vamos a registrar diferentes personas en una colección perteneciente a una BBDD NoSQL con algunos campos especiales. Estos no necesariamente tienen que seguir un patrón específico como veremos a continuación:

```
item1 = {
    Nombre: "Luís",
    Apellidos: "Martinez",
    Edad: 18,
    Aficiones: ["fútbol","senderismo","tenis"],
    Amigos: [
        {
```

```
            Nombre:"Monica",
            Edad:20
        },
        {
            Nombre:"Andrés",
            Edad:24
        }
    ]
}
db.hi.insert(item1)
db.amigos.find().pretty()
```

Ahora bien, si queremos añadir otros datos con algunas características diferentes la podemos hacer sin mayor problema introduciendo lo siguiente:

```
item2 = {
    Nombre: "Luís",
    Apellidos: "Martinez",
    Edad: 18,
}
db.hi.insert(item2)
db.amigos.find().pretty()
```

En un modelo relacional o SQL clásico esto sería imposible de hacer. Esta es una de las tantas ventajas que hemos comentado

Operación Find ejemplo del comando Find, para encontrar a alguien de nombre Luis:

```
db.amigos.find({Nombre:"Luís"})
```

ejemplo del comando Find, para encontrar a alguien de nombre Luis y apellidos Martinez:

```
db.amigos.find({$and:[{Nombre:"Luís"},{Apellidos:"Martinez"}]})
```

ejemplo del comando Find, para encontrar a alguien de nombre Luis y apellidos Martinez o edad mayor a 18:

```
db.amigos.find({
    $or:[
        {$and:[{Nombre:"Luís"},{Apellidos:"Martinez"}]},
        {Edad:{gte:18}}
    ]
})
```

ejemplo en el que el campo Amigos exista:

```
db.amigos.find({Amigos:{$exists:true}})
```

Lista de comando ejecutables contra la base de datos:

db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
db.auth(username, password)
db.cloneDatabase(fromhost)
db.commandHelp(name) returns the help for the command
db.copyDatabase(fromdb, todb, fromhost)
db.createCollection(name, { size : ..., capped : ..., max : ... })
db.createUser(userDocument)
db.currentOp() displays currently executing operations in the db
db.dropDatabase()
db.fsyncLock() flush data to disk and lock server for backups
db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
db.getCollectionNames()
db.getLastError() - just returns the err msg string
db.getLastErrorObj() - return full status object
db.getLogComponents()
db.getMongo() get the server connection object
db.getMongo().setSlaveOk() allow queries on a replication slave server
db.getName()

db.getPrevError()
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSlaveReplicationInfo()
db.dropUser(username)
db.repairDatabase()
db.resetError()
db.runCommand(cmdObj) run a database command. if cmdObj is a string, turns it into { cmdObj : 1 }
db.serverStatus()
db.setLogLevel(level,<component>)
db.setProfilingLevel(level,<slowms>) 0=off 1=slow 2=all
db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
db.unsetWriteConcern(<write concern doc>) - unsets the write concern for writes to the db
db.setVerboseShell(flag) display extra information in shell output
db.shutdownServer()
db.stats()
db.version() current version of the server

9.4 ¿Qué son las bases de datos de Series de Tiempo?

Una base de datos de series de tiempo[103] (TSDB) es una base de datos que está optimizada para manejar datos con marcas de tiempo o series temporales, que son datos indexados por tiempo. Los TSDB están diseñados para manejar grandes volúmenes de datos de series temporales de manera eficiente y rápida, con funciones como compresión, indexación y agregación. Por lo general, utilizan un formato de almacenamiento en columnas que permite consultas y análisis rápidos de datos de series temporales.

Las bases de datos de series temporales están completamente personalizadas con datos con marca de tiempo, que se indexan y se escriben de manera eficiente de tal manera que puede insertar datos de series temporales. Puede consultar esos datos de series temporales mucho más rápido de lo que lo haría en un entorno relacional o Base de datos NoSQL.

Los ejemplos de bases de datos de series temporales incluyen InfluxDB, Prometheus, TimescaleDB y OpenTSDB. Estas bases de datos se utilizan en una amplia gama de aplicaciones, que incluyen Internet de las cosas, monitoreo de sistemas y análisis de datos financieros.

Influjo DB es una de las bases de datos de series temporales más populares entre DevOps, que está escrita en Go. InfluxDB fue diseñado desde cero para proporcionar un motor de almacenamiento e ingesta de datos altamente escalable. Es muy eficiente para recopilar, almacenar, consultar, visualizar y tomar medidas en flujos de datos. datos de series de tiempo, eventos y métricas en tiempo real.

Proporciona políticas de reducción de resolución y retención de datos para respaldar el mantenimiento de datos de alto valor y alta precisión en la memoria y datos de menor valor en el disco. Está construido de manera nativa de la nube para proporcionar escalabilidad en múltiples topologías de implementación, incluidos los entornos de nube locales e híbridos.

InfluxDB es una solución de código abierto, utiliza InfluxQL, que es muy similar a un lenguaje de consulta de estructura, para interactuar con los datos. Ofrece agentes, paneles, consultas y tareas en un conjunto de he-

[103] Por serie de tiempo nos referimos a datos estadísticos que se recopilan, observan o registran en intervalos de tiempo regulares (diario, semanal, semestral, anual, entre otros). El término serie de tiempo se aplica por ejemplo a datos registrados en forma periódica que muestran, por ejemplo, las ventas anuales totales de almacenes, el valor trimestral total de contratos de construcción otorgados, el valor trimestral del PIB

rramientas. Es una herramienta todo en uno para generar paneles, visualizar y alertar.

Características:

- Alto rendimiento para datos de series de tiempo con alto nivel de ingesta y consultas en tiempo real
- InfluxQL para interactuar con datos que es un SQL como un lenguaje de consulta
- Componente principal de la pila TICK (Telegraf, InfluxDB, Chronograf y Kapacitor).
- Soporte de complementos para protocolos como collectd, Graphite, OpenTSDB para la ingestión de datos
- Puede manejar millones de puntos de datos en solo 1 segundo
- Políticas de retención para automáticos.ally eliminando los datos obsoletos

Como es de código abierto, puedes descargarlo e iniciarlo en tu servidor. Sin embargo, ofrecen InfluxDB Cloud en AWS, Azurey GCP.

Prometeo es un código abierto Monitor-Solución innovadora utilizada para comprender información a partir de datos de métricas y enviar las alertas necesarias. Tiene una base de datos local de series temporales en disco que almacena datos en un formato personalizado en el disco.

El modelo de datos de Prometheus es multidimensional y se basa en series de tiempo; almacena todos los datos como secuencias de valores con marca de tiempo. Es muy útil cuando se trabaja con series de tiempo completamente numéricas. Recopilar datos de microservicios y consultarlos es una de las fortalezas de Prometheus. Se integra estrechamente con Grafana para visualización.

Características

- Tiene un modelo multidimensional que utiliza el nombre de las métricas y los pares clave-valor (etiquetas)
- PromQL para consultar datos de series temporales para generarate tablas, alertas y gráficos Adhoc

- Utiliza el modo de extracción HTTP para recopilar datos de series de tiempo
- Utiliza intermediario gateforma de impulsar series de tiempo

Prometeo tiene cientos de exportadores para exportar los datos de Windows, Linux, Java, bases de datos, API, sitios web, hardware de servidor, PHP, mensajería y más. A monitor Linux, mira esto Configuración de Prometheus + Grafana.

Escala de tiempoDB es una base de datos relacional de código abierto que hace que SQL sea escalable para datos de series temporales. Esta base de datos se basa en PostgreSQL. Ofrece dos productos: en la primera opción es una edición comunitaria, de uso gratuito, que puede instalar en su servidor. La opción es TimescaleDB Cloud, donde obtiene una infraestructura completamente alojada y administrada en la nube para sus necesidades de implementación.

Se puede utilizar para DevOps. monitoring, entender métricas de aplicaciones, seguimiento de datos de dispositivos IoT, comprensión landing datos financieros, etc. Puede medir registros, eventos de Kubernetes, métricas de Prometheus e incluso métricas personalizadas.

Características

- Ejecute consultas de 10 a 100 veces más rápido que PostgreSQL, MongoDB
- Puede escalar a petabytes y escribe millones de puntos de datos por segundo
- Muy similar a PostgreSQL, muy fácil de operar para desarrolladores y administradores.ate
- Combina funcionalidades de bases de datos relacionales y de series de tiempo para crear aplicaciones potentes.
- Algoritmos incorporados y función de rendimiento para ahorrar muchos costos.

Grafito es una solución todo en uno para almacenar y visualizar de manera eficiente datos de series de tiempo en tiempo real. Graphite puede hacer dos cosas, almacenar datos de series de tiempo y representar gráficos bajo demanda. Pero no recopila datos por usted; para eso, puede utilizar herramientas como collectd, Ganglia, Sensu, telegraf, etc.

Tiene tres componentes: Carbono, Susurro, y Web de grafito. Carbon recibe los datos de la serie temporal, agregados y lo persiste en el disco. Whisper es un almacenamiento de base de datos de series temporales que almacena los datos. Graphite-Web es la interfaz para crear paneles y visualizar los datos.

Características del grafito:

- El formato de métricas en el que se envían los datos es sencillo.
- API integral para renderizar los datos y crear cuadros, tableros de control, gráficos
- Proporciona un amplio conjunto de funciones de representación transformadora y biblioteca estadística
- Encadena múltiples funciones de renderizado para construir una consulta de destino.

QuestDB es una base de datos relacional orientada a columnas que puede realizar análisis en tiempo real de datos de series de tiempo. Funciona con SQL y algunas extensiones para crear un modelo relacional para datos de series de tiempo. QuestDB ha sido codificado desde cero y no tiene dependencias que mejoren su rendimiento.

QuestDB admite uniones relacionales y de series de tiempo, lo que ayuda a correlacionar los datos. La forma más sencilla de comenzar con QuestDB es implementarlo dentro de un contenedor Docker.

Características

- Consola interactiva para importar datos usando arrastrar y soltar y consultarlos
- Compatible con la nube nativa (AWS, Azure, GCP), local o integrada
- Proporciona integración empresarial con funciones como directorio activo, alta disponibilidad, enterprise seguridad, agrupación

- Proporciona información en tiempo real mediante análisis operativos y predictivos.

Flujo de tiempo de AWS es un servicio de base de datos de series temporales sin servidor que es rápido y escalable. Se utiliza principalmente para que las aplicaciones de IoT almacenen billones de eventos en un día y 1000 veces más rápido con una décima parte del costo de las bases de datos relacionales.

Utilizando su motor de consultas especialmente diseñado, puede consultar simultáneamente datos recientes y datos históricos almacenados.neously. Proporciona múltiples funciones integradas para analizar datos de series temporales y encontrar información útil.

Características

- No hay servidores que administrar ni instancias que aprovisionar; todo se maneja automaticoally.
- Rentable, pague solo por lo que ingiera, almacene y consulte.
- Capaz de ingerir billones de eventos al día sin que disminuya el rendimiento.
- Capacidad analítica incorporada con funciones estándar de SQL, interpolación y suavizado para identificar tendencias, patrones y anomalías
- Todos los datos se cifran mediante el sistema de administración de claves (KMS) de AWS con claves administradas por el cliente (CMK)

OpenTSDB es una base de datos de series temporales escalable que se ha escrito sobre HBase. Es capaz de almacenar billones de puntos de datos a millones de escrituras por segundo. Puede mantener los datos en OpenTSDB para ver con su marca de tiempo original y valor preciso, para que no pierda ningún dato.

Tiene un demonio de serie temporal (TSD) y utilidades de línea de comandos. El demonio de series de tiempo es responsable de almacenar datos en HBase o recuperarlos de él. Puede hablar con TSD mediante HTTP API, telnet o una sencilla GUI incorporada. Necesita herramientas como flume, collectd, vacuumetrix, etc., para recopilar datos de varias fuentes en OpenTSDB.

Características

- Puede agregar, filtrar y reducir la resolución de métricas a una velocidad vertiginosa
- Almacena y escribe datos con precisión de milisegundos
- Se ejecuta en Hadoop y HBase y se escala fácilmente agregando nodos al clúster.
- Utiliza GUI para generar gráficos

En resumen: dado que en la actualidad se utilizan cada vez más dispositivos IoT/inteligentes, se está generando un enorme tráfico en tiempo en sitios Web con millones de eventos en un día, las operaciones en el mercado están aumentando y ¡ha llegado la base de datos de series temporales!.

9.5 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias

Elegir la base de datos más adecuada para un proyecto supone enfrentarse a todo tipo de posibilidades de elección. Quizá la primera, y la más importante, es elegir una base de datos de código abierto o una propietaria. Es decir, una de código abierto, generalmente gratuita, o una propiedad de una empresa. A la vista de esto, la primera ventaja de las bases de datos de código abierto salta a la vista: evitan tener que hacer frente a un desembolso económico fuerte, puesto que no hace falta comprar su licencia a un fabricante. A no ser que tengas necesidades muy concretas y específicas, pero la proliferación de bases de datos de código abierto en los últimos años, así como de diversos complementos para ellas, hace que resulten cada vez más atractivas incluso para fines muy concretos.

Mientras tanto, la popularidad de los sistemas de bases de datos con licencia comercial, es decir, de las propietarias, ha ido cayendo a lo largo de los últimos años. Y según la lista de sistemas de gestión de bases de datos de DB-Engines, en muchos casos han sido superadas por las bases de datos de código abierto en lo que respecta a la adopción en empresas.

Ventajas de las Bases de Datos de Código Abierto Una base de datos de código abierto es un sistema de gestión de bases de datos abierto y de uso gratuito, y quizá una de las pocas desventajas que tiene frente a

las bases de datos propietarias es que no cuenta con un servicio de soporte técnico por parte de un proveedor, algo con lo que sí cuentan las bases de datos propietarias. Pero precisamente de esta desventaja nace una de sus ventajas.

Las bases de datos de código abierto cuentan con una nutrida comunidad de usuarios que, en muchos casos no solo comparten sus conocimientos en foros y grupos de usuarios especializados, sino que en muchos casos están dispuestos a ayudar a otros usuarios con problemas y necesidades concretas con ellas.

Además, la información que se guarda en una base de datos de código abierto es de quien la inserta en ella. Sin ningún tipo de restricción. También puedes modificarla para adaptarla a tus necesidades, aunque para ello necesitas contar con ciertos conocimientos de desarrollo. Puedes crear aplicaciones y complementos con base en el Software de código abierto que elijas, de una manera muy parecida a como sucede por ejemplo con el CMS WordPress. Por lo tanto, las bases de datos de código abierto te ofrecen un sinfín de posibilidades de personalización y ampliación.

Las Principales Bases de Datos de Código Abierto El número de bases de datos de código abierto crece cada vez más a medida que pasan los años. No obstante, entre todas las posibilidades, hay varias que, según TechTarget, o bien por su sencillez y robustez, o bien por sus posibilidades y facilidades de ampliación y personalización, son las que consiguen una mayor adopción y despliegue en las empresas. Entre ellas están las seis siguientes, quizá las más populares y utilizadas.

MySQL Sin duda esta es la más extendida de las bases de datos de código abierto, y durante muchos años, la opción por defecto para muchos desarrolladores a la hora de optar por una base de datos. Oracle se hizo con ella hace ya bastantes años cuando compró Sun Microsystems, y es un sistema de gestión de bases de datos relacional. Esto quiere decir que utiliza tablas para almacenar los datos, y los tipos de datos almacenados deben estar relacionados entre sí de alguna manera. Cuenta con una edición de código abierto y algunas versiones de pago, y debido a su popularidad hay muchas herramientas de terceros para ella, así como una cantidad importante de documentación centrada en facilitar su aprendizaje.

La base de MySQL se originó a partir de un sistema de base de datos

propietario antiguo, mantener la mayor parte de su compatibilidad y convertir los resultados en código abierto. Sus creadores se centraron en la velocidad, y en la actualidad sigue estando considerada entre las opciones más rápidas de bases de datos. Además, de la velocidad, cuenta con un Script que contribuye a la mejora de la seguridad de las bases de datos de los usuarios. Esta base de datos es compatible, entre otros, con los lenguajes de programación C, C++, Java, Python y Ruby.

No obstante, se trata de una base de datos con licencia dual. Esto implica que algunas de sus funciones y Plugins solo son accesibles en ediciones propietarias de pago. En cuanto a su curva de aprendizaje, no es muy elevada. No es necesario saber mucho SQL para utilizarla, y puedes trabajar con la base de datos desde la línea de comandos. MySQL tiene una compatibilidad elevada prácticamente con cualquier sistema operativo. Se trata de una base de datos sólida, rápida y versátil, por lo que es adecuada para multitud de casos de uso.

MariaDB Cuando Oracle compró Sun, y con ella MySQL, hubo quien no se lo tomó muy bien, porque aunque Oracle ha mantenido MySQL como código abierto, la compañía no tiene mucha reputación que digamos como defensora del Software abierto. Uno de los que no se creyó a Oracle fue uno de los propios fundadores de MySQL, que dejó la compañía para crear MariaDB, un fork de MySQL con una compatibilidad casi completa con ella.

Se puede utilizar como un sustituto prácticamente para todo de MySQL, aunque cuenta con varias funciones únicas que la distinguen de ella. Para empezar, MariaDB usa el motor de almacenamiento Aria para gestionar querys de SQL complejas. Esto hace que la base de datos sea todavía más veloz que MySQL. Además, puedes utilizar filas dinámicas como columnas de tabla, lo que la hace más flexible y adaptable a diversas situaciones.

Además, cuenta con motores de almacenamiento especializados para casos de uso concretos que no tiene MySQL. Por ejemplo, permite implementar el almacenamiento distribuido o las transacciones distribuidas. Como hemos mencionado, puedes utilizar MariaDB en los mismos supuestos y casos que MySQL, pero no hay compatibilidad entre las dos, por lo que tendrás que elegir cuál de las dos quieres utilizar.

PostgreSQL Esta base de datos relacional y de código abierto es muy utilizada en ciencia de datos y creación de gráficos, y también en el sector

de la Inteligencia Artificial. Esto último se debe a que PostgreSQL está especialmente pensada para aplicaciones de Python y Ruby. No obstante, también es compatible con PHP. Se ha desarrollado pensando en la elegancia, y cuenta con ciertas funciones bastante diferenciadas. Entre ellas están la posibilidad de implementar la replicación asíncrona, y el soporte nativo para el almacenamiento de documentos de estilo JSON o XML.

Además, PostgreSQL permite hacer búsquedas completas de texto en la base de datos, y cuenta con varios tipos de datos integrados de gran valor para algunas aplicaciones, como los rangos, los arrays y la geolocalización. Eso sí, no es excesivamente adecuada para aplicaciones que requieran operaciones intensivas de lectura, y la creación de informes de datos con regularidad puede hacer que el almacenamiento de documentos no esté muy fino si tiene que almacenar un conjunto de datos muy grande.

SQLite También relacional, SQLite es una base de datos de código abierto compuesta por una librería ligera y pequeña que ofrece un motor de base de datos. A pesar de su ligereza, permite generar bases de datos de cientos de Terabytes, y un tamaño máximo de filas de un Gigabyte. Incluso con archivos de este tamaño, SQLite sigue siendo rápida. Además, es compatible con Java, Python, C o C++, entre otros lenguajes.

Cuenta con decenas de casos de uso, y por ejemplo, la encontrarán muy útil los desarrolladores de aplicaciones sencillas. Está integrada en smartphones y en muchos ordenadores. También es muy valorada en aplicaciones de Internet de las Cosas y Webs de poco tráfico. Eso sí, por su estructura, no está indicada para sitios con mucho tráfico, ya que su rendimiento se resiente. Además, cuenta con ciertas limitaciones que en algunos casos pueden ser importantes.

Estas hacen que, entre otras cosas, no esté indicada en aplicaciones a las que tienen que acceder varios usuarios con permisos de acceso especiales. SQLite lee y escribe en un archivo de disco común y corriente, por lo que los únicos permisos de acceso aplicables para esta base de datos son los incluidos habitualmente en el sistema operativo.

MongoDB Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios

están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades. Si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Redis Esta base de datos de código abierto es distinta a las más utilizadas, ya que se utiliza sobre todo en situaciones completamente distintas a ellas. Se trata de una base de datos que permite estructurar datos como pares de valor clave. Por tanto, se trata de un sistema parecido a los arrays asociativos de PHP o los diccionarios de Python. En realidad, Redis es un sistema para enlazar datos para su referencia, más adelante, con rapidez. Está considerada como la mejor elección como base de datos para cacheado, y también para el trabajo con datos distribuidos.

Para empezar, es una solución de almacenamiento en memoria, enteramente en RAM. Esto quiere decir que sus velocidades de lectura y escritura son muy rápidas. Para aprender sus bases tampoco necesitas mucho tiempo, ya que puedes conocerlas en minutos y empezar con rapidez con el almacenamiento de objetos en ella. Eso sí, puede que no sea muy útil para aplicaciones complejas, pero basta con utilizarlas junto con otras bases de datos, como MariaDB, para el resto de la aplicación que se esté desarrollando.

9.6 Programando en Diferentes Manejadores de Bases de Datos

Existen diferentes paquetes para que permiten trabajar con bases de datos, en esta sección mostraremos cómo trabajar con algunos de ellos:

- SQLite
- MySQL/MariaDB/Microsoft SQL Server

- PostgreSQL
- MongoDB
- Redis

A continuación, mostraremos el mismo ejemplo en cada uno de estos manejadores:

9.6.1 SQLite

es una herramienta de Software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. Está presente en todos los dispositivos Android, iPhone, Mac y Windows 10, además en los navegadores Firefox, Chromium, Chrome y Safari entre otros.

SQLite implementa el estándar SQL92 y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente.

Lenguajes de Programación de SQLite gracias a que SQLite[104] es Software libre, es posible encontrar una gran cantidad de componentes, librerías y Drivers para interactuar con SQLite desde una gran diversidad de lenguajes y plataformas de programación. Ya sea que estemos utilizando lenguajes modernos como Java, Perl, Python, PHP, Ruby, C#, lenguajes más antiguos como Pascal, SmallTalk, Clipper, o lenguajes poco conocidos como Suneido, REXX, S-Lang, para todos podemos encontrar librerías y ejemplos de código para SQLite.

[104] También existe Rqlite que es una base de datos relacional distribuida y liviana de código abierto para configuraciones de cluster que está construido sobr SQLite y tiene como objetivo ser fácil de usar y tolerante a fallas.

http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers ofrece más información sobre "wrappers" para SQLite sobre diferentes plataformas y lenguajes.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install sqlite3 sqlite3-doc sqlitebrowser
```

y ya podemos iniciar a trabajar, usando

```
# sqlite3
```

SQLite y Python SQLite es probablemente la base de datos más sencilla para conectarse con una aplicación Python, ya que no necesita instalar ningún módulo SQL externo de Python para hacerlo. De forma predeterminada, su instalación de Python contiene una biblioteca SQL de Python llamada *sqlite3* que puede usar para interactuar con una base de datos SQLite.

Además, las bases de datos SQLite no tienen servidor y son autónomas, ya que leen y escriben datos en un archivo. Esto significa que, a diferencia de MySQL y PostgreSQL, ni siquiera necesita instalar y ejecutar un servidor SQLite para realizar operaciones de base de datos.

Así es como se usa *sqlite3* para conectarse a una base de datos SQLite en Python:

```
import sqlite3
from sqlite3 import Error

def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")
    return connection

connection = create_connection("/trayectoria/basedatos")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que cree o abra una base de datos:

```
sqlite3 prueba.db
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

9.6.2 MySQL/MariaDB

MySQL/MariaDB es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, estos manejadores de bases de datos, no son más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL/MariaDB, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL/MariaDB fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL/MariaDB, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

MariaDB este sistema de gestión de base de datos es una bifurcación de MySQL, aunque parecidos, estos dos sistemas de gestión de bases de datos siguen siendo bastante diferentes:

- MariaDB tiene licencia GPL mientras que MySQL tiene un enfoque de doble licencia
- Cada mango se acumula de una manera diferente
- MariaDB soporta muchos motores de almacenamiento diferente
- En muchos escenarios, MariaDB ofrece un mejor rendimiento

Instalación en Debian GNU/Linux En este caso tenemos dos opciones, instalar MariaDB que ya es parte integral de múltiples aplicaciones dentro de GNU/Linux o usar MySQL, en cada caso el uso es prácticamente el mismo.

Para instalar el paquete MariaDB usamos:

```
# apt install mariadb-server mariadb-client
```

y para configurarlo, usamos:

```
# mysql
> use mysql
> update user set plugin=” where user='root';
> alter user root@localhost identified by 'XXXXXXX';
> flush privilegies;
> exit
```

y para usarlo

```
$ mysql -u root -p
```

Para instalar el paquete MySQL usamos:

```
# apt install mysql-common mysql-client mysql-server mytop
mysql-admin
```

y para configurarlo (las bases de datos se guardan en /var/lib/mysql), usamos:

```
# mysql_secure_installation
```

o

```
# mysql
```

en caso de no crear contraseña, podemos ingresar así:

```
$ mysql -u root -p
```

una vez en la consola del cliente de MySQL creamos un nuevo usuario

```
> create user nombre identified by 'XXXXX';
```

y a continuación le damos privilegios totales

```
> gran all privileges on *.* to nombre with grant option;
> flush privileges;
> exit;
```

otras opciones a instalar son:

```
# apt install ferret mysql-workbench mysql-workbench-data
mycli
# snap install squirrelsql mysql-workbench-community data-
grip dbeaver-ce
```

Crear un Nuevo Usuario Para esto usamos la declaración CREATE USER. La sintaxis básica de la declaración es:

> CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';

En la sintaxis anterior, claramente debes de reemplazar el nombre de usuario y la contraseña con el nombre de usuario y contraseña deseados.

Establezca el hostname en localhost si desea que el usuario pueda conectarse a MySQL Server solo desde localhost. Si desea que el usuario pueda conectarse desde cualquier host, use el comodín % como hostname.

Por ejemplo, crearemos un usuario con el nombre Javier y la contraseña 1234 usando el siguiente comando:

> CREATE USER 'Javier'@'localhost' IDENTIFIED BY '1234';

Sin embargo, este usuario no podrá trabajar con ninguna de las bases de datos MySQL hasta que se le otorguen privilegios adicionales. Ahí el detalle.

Como otorgar privilegios adicionales al usuario inmediatamente después de crear con éxito el nuevo usuario, podemos otorgar privilegios a este nuevo usuario. En la mayoría de los casos, otorgará privilegios a los usuarios de MySQL en función de la base de datos particular a la que la cuenta debería tener acceso. Hay varios tipos de privilegios que se pueden otorgar a una cuenta de usuario, como:

- ALL PRIVILEGES: otorga todos los privilegios a una cuenta de usuario.
- ALTER: el usuario puede cambiar la estructura de una tabla o base de datos.
- CREATE: la cuenta de usuario puede crear bases de datos y tablas.
- DROP: la cuenta de usuario puede eliminar bases de datos y tablas.
- DELETE: la cuenta de usuario puede eliminar filas de una tabla específica.
- INSERT: la cuenta de usuario puede insertar filas en una tabla específica.

- SELECT: la cuenta de usuario puede leer una base de datos.
- UPDATE: la cuenta de usuario puede actualizar las filas de la tabla.

Para proporcionar acceso y otorgar permisos a un $user, generalmente se debe usar la siguiente declaración GRANT:

> GRANT permission_type ON privilege_level TO 'username'@'hostname';

Sí retomamos el ejemplo de arriba, al usuario Javier y su base de datos Javierdb, se le puede otorgar permisos, así:

> GRANT ALL PRIVILEGES ON Javierdb.* TO 'Javier'@'localhost';

Oh claro otorgarle permisos específicos, solo cambiando el ALL PRIVILEGES, por SELECT, CREATE, INSERT.... todos separados por una coma (,).

En algunos casos, es posible que desee crear otro "superusuario". Para otorgar a un usuario los mismos privilegios que el usuario raíz de MySQL, usamos el siguiente comando, que otorga privilegios globales al usuario james que se conecta a través de localhost:

> GRANT ALL ON *.* TO 'Javier'@'localhost' WITH GRANT OPTION;

Como revocar privilegios y eliminar un usuario si necesitas revocar privilegios del usuario Javier en una base de datos Javierdb, sería suficiente con la sintaxis siguiente, que es muy similar

> REVOKE ALL PRIVILEGES ON Javierdb.* TO 'Javier'@'localhost';

Mientras que para eliminar a ese usuario. Basta con:

> DROP USER 'Javier'@'localhost';

Para que cualquier cambio surta efecto, es necesario usar el comando:

> FLUSH PRIVILEGES;

Mantenimiento En caso de requerir mantenimiento a las tablas (analizar, verificar, optimizar y reparar las tablas de una base de datos) podemos usar *mysqlcheck*, por ejemplo para verificar y reparar todas las bases de datos:

```
$ mysqlcheck -A –auto-repair -u root -p
```

para forzar la optimización y autoreparación de todas las tablas:

```
$ mysqlcheck -A –auto-repair –f -o -u root -p
```

para verificar todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -c
```

para verificar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -a
```

para reparar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -r
```

para optimizar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -o
```

Exportación e Importación Para exportar una base de datos, debemos utilizar una utilidad de cliente *mysqldump* que crea la copia de seguridad lógica de las bases de datos en archivos de texto SQL, lo que facilita la transferencia de archivos de un servidor a otro, para ello hacemos:

```
$ mysqldump -u usuario -p baseDatos > archivo.sql
```

para restaurarla usamos:

```
$ mysqldump -u usuario -p baseDatos < archivo.sql
```

Exportar como CSV Podemos exportar el contenido de una o más tablas al formato CSV, por ejemplo:

```
> SELECT * FROM test_proyect
INTO OUTFILE '/var/tmp/get_all_queries.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';
```

MySQL y Python A diferencia de SQLite, no existe un módulo SQL de Python predeterminado que pueda usar para conectarse a una base de datos MySQL. En su lugar, deberá instalar un controlador Python SQL para MySQL a fin de interactuar con una base de datos MySQL desde una aplicación Python. Uno de estos controladores es *mysql-connector-python*. Puede descargar este módulo SQL de Python con pip:

```
$ pip install mysql-connector-python
```

Tenga en cuenta que MySQL es un sistema de gestión de bases de datos basado en servidor. Un servidor MySQL puede tener varias bases de datos. A diferencia de SQLite, donde crear una conexión equivale a crear una base de datos, una base de datos MySQL tiene un proceso de dos pasos para la creación de la base de datos:

- Establezca una conexión a un servidor MySQL.
- Ejecute una consulta separada para crear la base de datos.

Defina una función que se conecte al servidor de la base de datos MySQL y devuelva el objeto de conexión:

```
import mysql.connector
from mysql.connector import Error

def create_connection(host_name, user_name, user_password):
    connection = None
    try:
        connection = mysql.connector.connect(
```

```
                host=host_name,
                user=user_name,
                passwd=user_password
            )
            print("Connection to MySQL DB successful")
        except Error as e:
            print(f"The error '{e}' occurred")
        return connection

    connection = create_connection("localhost", "root", "")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que nos muestre las bases de datos existentes:

```
show databases
```

si necesitamos crear una, por ejemplo prueba, usamos:

```
create databases prueba
```

ahora, le indicamos que usaremos dicha base de datos, mediante:

```
use prueba
```

y ahora podremos, por ejemplo ver las tablas creadas usando:

```
show tables
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
Select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
From Ventas
INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

9.6.3 PostgreSQL

Las bases de datos relacionales son uno de los tipos más populares que existen, y su uso nos permite crear sitios web de toda clase. Entre los sistemas de bases de datos relacionales más utilizados se encuentra uno que es conocido como PostgreSQL, que dicho sea de paso está entre los más usados a nivel mundial.

PostgreSQL, o simplemente Postgres para darle un nombre más pintoresco, es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON.

Como decíamos, se trata de un sistema de código abierto y además gratuito, y su desarrollo es llevado adelante por una gran comunidad de colaboradores de todo el mundo que día a día ponen su granito de arena para hacer de este sistema una de las opciones más sólidas a nivel de bases de datos.

Dos detalles a destacar de PostgreSQL es que posee data types (tipos de datos) avanzados y permite ejecutar optimizaciones de rendimiento avanzadas, que son características que por lo general solo se ven en sistemas de bases de datos comerciales, como por ejemplo SQL Server de Microsoft u Oracle de la compañía homónima.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install postgresql postgresql-client postgresql-doc postgresql-contrib
# snap install postgresql10
```

y ya podemos conectarnos al servidor mediante:

```
# postgres psql
```

PostgreSQL y Python Al igual que MySQL, no existe una biblioteca Python SQL predeterminada que pueda usar para interactuar con una base de datos PostgreSQL. En su lugar, debe instalar un controlador Python SQL de terceros para interactuar con PostgreSQL. Uno de esos controladores Python SQL para PostgreSQL es *psycopg2*. Ejecute el siguiente comando en su terminal para instalar el módulo Python SQL psycopg2:

```
$ pip install psycopg2
```

Al igual que con las bases de datos SQLite y MySQL, definirás create_connection () para hacer una conexión con tu base de datos PostgreSQL:

```
import psycopg2
from psycopg2 import OperationalError

def create_connection(db_name, db_user, db_password, db_host, db_port):
    connection = None
    try:
```

```
            connection = psycopg2.connect(
                database=db_name,
                user=db_user,
                password=db_password,
                host=db_host,
                port=db_port,
            )
            print("Connection to PostgreSQL DB successful")
        except OperationalError as e:
            print(f"The error '{e}' occurred")
        return connection

connection = create_connection("postgres", "postgres", "abc123",
"127.0.0.1", "5432")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que nos muestre las bases de datos existentes:

```
\l
```

si necesitamos crear una, por ejemplo prueba, usamos:

```
create databases prueba
```

ahora, le indicamos que usaremos dicha base de datos, mediante:

```
\c prueba
```

y ahors podremos, por ejemplo ver las tablas creadas usando:

```
\d
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text not null,
    precio money,
    precioVenta money,
    primary key (id_prod)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '12/01/2020 8:01:00.00'), (2, 5, '12/01/2020 10:15:00.00'),
(2, 4, '12/01/2020 13:34:00.00'), (1, 3, '12/01/2020 21:56:00.00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

9.6.4 MongoDB

MongoDB (del inglés humongous, "enorme") es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina

y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades.

Si no conocemos a MongoDB, al principio podemos sentirnos un poco perdidos. Al no tener tablas ni nada que se parezca a SQL como referencia, tendremos que estudiar un poco su filosofía y características para entender cómo manejar los datos. Aun así, MongoDB es una seria candidata para almacenar los datos de nuestras aplicaciones.

Características de MongoDB si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Características principales:

- Consultas ad hoc. Con MongoDb podemos realizar todo tipo de consultas. Podemos hacer búsqueda por campos, consultas de rangos y expresiones regulares. Además, estas consultas pueden devolver un campo específico del documento, pero también puede ser una función JavaScript definida por el usuario.

- Indexación. El concepto de índices en MongoDB es similar al empleado en bases de datos relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.

- Replicación. Del mismo modo, la replicación es un proceso básico en la gestión de bases de datos. MongoDB soporta el tipo de replicación primario-secundario. De este modo, mientras podemos realizar consultas con el primario, el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad con la particularidad de que los nodos secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

- Balanceo de carga. Resulta muy interesante cómo MongoDB puede escalar la carga de trabajo. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo

de carga o servicio de replicación de datos, de modo que podemos mantener el sistema funcionando en caso de un fallo del hardware.

- Almacenamiento de archivos. Aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos, Mongo puede ser utilizado también como un sistema de archivos. Esta funcionalidad, llamada GridFS e incluida en la distribución oficial, permite manipular archivos y contenido.
- Ejecución de JavaScript del lado del servidor. MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install mongodb mongodb-clients mongodb-server
# snap install mongo44-configurable
```

y ya podemos iniciar su uso mediante:

```
$ mongo
```

MongoDB y Python A diferencia de SQLite, no existe un módulo noSQL de Python predeterminado que pueda usar para conectarse a una base de datos MongoDB. En su lugar, deberá instalar un controlador Python noSQL para MongoDB a fin de interactuar con una base de datos MongoDB desde una aplicación Python. Uno de esos controladores es *pymongo*. Puede descargar este módulo noSQL de Python con pip:

```
$ pip install pymongo
```

Defina una función que se conecte al servidor de la base de datos MongoDB y devuelva el objeto de conexión:

```
from pymongo import MongoClient
import pymongo

def get_database():
    # Provide the mongodb atlas url to connect python to
mongodb using pymongo
```

```
        CONNECTION_STRING = "mongodb+srv:
    //<username>:<password>@<cluster-name>.mongodb.net/myFirstDatabase"
        # Create a connection using MongoClient. You can im-
  port MongoClient or use pymongo.MongoClient
        client = MongoClient(CONNECTION_STRING)
        # Create the database for our example
        return client['user_shopping_list']

    dbname = get_database()
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema nos muestre las colecciones existentes (bases de datos), mediante:

```
    db
```

podemos crear una usando:

```
    db.createCollection(prueba)
```

podemos ver las existentes mediante:

```
    show collections
```

y le pedimos usar alguna, mediante:

```
    use prueba
```

y estamos en posibilidad de crear nuestras documentos, por ejemplo:

```
    var miProductos =
    [
        {"_id" : 1, "descripcion" : "Cigarros", "precio" : 25.5,
  "precioVenta" : 30.0},
        {"_id" : 2, "descripcion" : "Refresco", "precio" : 4.5,
  "precioVenta" : 6.0},
    ];
    db.Productos.insert(miProductos);
```

y los visualizamos:

```
db.Productos.find()
```

ahora procedemos hacer lo pertinente para nuestra otro documento, usando:

```
var miVentas =
[
    {"_id" : 1, "id_prod" : 1, "Cantidad" : 2, "Fecha" :
"2020/12/01 8:01:00"},
    {"_id" : 2, "id_prod" : 2, "Cantidad" : 5, "Fecha" :
"2020/12/01 10:15:00"},
    {"_id" : 3, "id_prod" : 2, "Cantidad" : 4, "Fecha" :
"2020/12/01 13:34:00"},
    {"_id" : 4, "id_prod" : 1, "Cantidad" : 3, "Fecha" :
"2020/12/01 21:56:00"},
];
db.Ventas.insert(miVentas);
db.Ventas.find()
```

de esta forma, ahora podemos usar Find para hacer el equivalente a un JOIN como en los ejemplos anteriores y mostrar el resultado de la solicitud, mediante:

```
db.Ventas.find().forEach(
    function (object) {
        var commonInBoth=db.Productos.findOne({ "_id":
object.id_prod} );
        if (commonInBoth != null) {
            print(object.Fecha, object.Cantidad, common-
InBoth.descripcion, commonInBoth.precioVenta, (commonInBoth.precioVenta
* object.Cantidad));
        }
    }
);
```

Exportar Base de Datos en MongoDB En MongoDB, cuando se exporta información, se puede obtener un archivo de texto legible por el ser humano con sus datos. Por defecto, la información se exporta en formato *json*, pero también se puede exportar en formato *csv*.

Para exportar información desde MongoDB, utilizaremos el comando mongoexport. Este comando nos permite exportar de forma detallada, por lo que podemos especificar una base de datos, una colección, un campo e incluso utilizar una consulta para la exportación.

Si queremos exportar debemos de ejecutar lo siguiente:

```
# mongoexport –db mibasededatos -c restaurants –out Restaurants.json
```

En el comando anterior, se utilizó –db para especificar la base de datos, -c para la colección y –out, para el archivo en el que se guardarán los datos.

Importar Base de Datos en MongoDB Importaremos los datos de Restaurante.json a una nueva base de datos llamada mibasededatos y a una colección llamada restaurantes. Utilizaremos el comando mongoimport de la siguiente forma:

```
# mongoimport –db mibasededatos –collection restaurantes –file Restaurants.json
```

Como podemos observar, la base de datos no la hemos tenido que crear, si no que al importar los datos se ha creado de forma automática.

Comprobar los datos importados Nos conectaremos a la base de datos:

```
$ mongo mibasededatos
```

Se nos habrá cambiado el prompt, por lo que estaremos conectados a la base de datos, contaremos los documentos de la colección importada:

```
> db.restaurantes.count()
```

y debe coincidir con los datos exportados. Y con esto veremos que hemos importado los mismos registros que hemos exportado.

9.6.5 Redis

El servidor de diccionarios remotos o Redis (Remote Dictionary Server) es un almacén de datos de valores clave que destaca por su gran velocidad y su facilidad de uso. Un almacén de valores clave es una base de datos NoSQL que utiliza un sistema de clave/valor para guardar la información. El proyecto se inició cuando Salvatore Sanfilippo, el desarrollador original de Redis, trataba de mejorar la escalabilidad de su empresa emergente italiana. A partir de ahí, desarrolló Redis, que ahora se utiliza como base de datos, caché, agente de mensajes y cola.

En Redis se almacenan los datos como un grupo de clave-valor, donde la clave es un identificador único y el valor puede contener cualquier tipo de dato como un texto, un archivo, un documento, o una imagen, entre otros. Redis utiliza la memoria para almacenar de forma rápida datos que pueden rescatarse de manera inmediata cuando son necesarios.

Redis ofrece tiempos de respuesta inferiores al milisegundo, lo que permite que se realicen millones de solicitudes por segundo para aplicaciones en tiempo real de la industria, como videojuegos, tecnología publicitaria, servicios financieros, sanidad e IoT. Hoy en día, Redis es uno de los motores de código abierto más populares en la actualidad, denominado la base de datos "preferida" por Stack Overflow durante cinco años consecutivos. Por su rápido rendimiento, Redis es una opción muy habitual en aplicaciones de almacenamiento en caché, administración de sesiones, videojuegos, tablas de clasificación, análisis en tiempo real, datos geoespaciales, servicios de vehículos compartidos, Chat/mensajería, Streaming de contenido multimedia y publicación/suscripción.

Para qué se Usa y sus Funcionalidades Redis tiene muchos usos debido a sus características especiales. Se trata de una gran alternativa para implementar una caché y reducir la latencia a la hora de acceder a los datos de un servidor, en salas de chat para ofrecer un alto rendimiento, en el desarrollo de videojuegos para ofrecer tablas de clasificación en tiempo real, como almacenamiento de sesiones de usuarios, como complemento en Streamings multimedia para guardar metadatos, o en Machine Learning para procesar grandes cantidades de información y ofrecerla en tiempo real.

Los principales beneficios que aporta Redis son:

- Guarda la información en memoria en lugar de en discos HDD o SSD como las bases de datos SQL.

- Cuenta con una amplia variedad de estructuras de datos que permite adaptarse a cualquier aplicación.
- El código para utilizar Redis es sencillo y simple, haciendo que sean necesarias menos líneas de código.
- Permite replicar los datos de forma asíncrona en distintos servidores, favoreciendo un mejor rendimiento, acelerando los tiempos de recuperación e incrementando el nivel de seguridad.
- Aporta un alto nivel de disponibilidad y escalabilidad.
- Es un proyecto de código abierto que se encuentra en constante evolución.

Instalación en Debian GNU/Linux de Redis Para instalar el paquete usamos:

```
# apt install redis-server
```

Una vez instalado el servicio de Redis es necesario acceder al mismo, iniciamos Redis usando:

```
$ redis-server
```

Para comprobar si Redis funciona correctamente, iniciamos la interfaz de comunicación con la base de datos:

```
$ redis-cli
```

La interfaz debería mostrar entonces la dirección IP y el puerto a través del cual se ejecuta Redis, a los que se puede enviar un ping de comprobación.

```
127.0.0.1:6397> ping
PONG
```

Si Redis responde, queda demostrado que el sistema de bases de datos se ha instalado correctamente. Ahora también se puede comprobar si se puede escribir texto.

```
127.0.0.1:6397> set test "OK!"
127.0.0.1:6397> get test
"OK!"
```

Configurar Redis Redis se instala, en un principio, con la configuración estándar, que luego puede modificarse con los comandos correspondientes.

```
127.0.0.1:6397> config get *
```

En la lista de ajustes para la configuración, las parejas de elementos se descomponen en dos posiciones, una debajo de la otra: al elemento *dbfilename* le corresponde, entonces, el valor *dump.rdb*. El asterisco que hemos usado para abrir la lista actúa de marcador de posición para un ajuste determinado de la lista. Cuando solo se quiere examinar un ajuste, el asterisco se reemplaza por el nombre del elemento, utilizando siempre para ello el que se encuentra en primera posición, que es la llave para el valor de configuración correspondiente.

```
127.0.0.1:6397> config get dbfilename
1) "dbfilename"
2) "dump.rdb"
```

Para cambiar una entrada en el archivo de configuración[105], se usa el comando *set*. Puede usarse, por ejemplo, para definir una contraseña:

```
127.0.0.1:6397> config set requirepass "password"
OK
```

Tras hacerlo, si solicitamos la contraseña con el comando get, se nos pedirá que la introduzcamos nosotros primero, al fin y al cabo, por algo le hemos puesto una. Para introducirla, usamos el comando authy, a continuación, consultamos la entrada en el archivo de configuración como acabamos de hacer.

```
127.0.0.1:6397> auth "password"
127.0.0.1:6397> config get requirepass
1) "requirepass"
2) "password"
```

[105] Existen más formas de hacer más segura tu base de datos. En la página web oficial de Redis, los desarrolladores resumen diversos puntos al respecto.

En realidad, Redis guarda todos los datos en la memoria principal. Teniendo esto en cuenta, para lograr la persistencia de los datos, se puede almacenar una copia (*snapchot*) de la base de datos en el disco duro, que se ubicará en el archivo *dump.rdb*.

```
127.0.0.1:6397> save
```

Con el comando *save*, se crea manualmente una copia, pero también pueden programarse para que se realicen de forma automática.

```
127.0.0.1:6397> save 60 10
```

En este ejemplo, hemos asignado dos parámetros al comando: ahora, se creará una copia cada 60 segundos si ya se han producido 10 cambios en dicho intervalo de tiempo.

Sin embargo, el comando *save* no se recomienda mientras el sistema esté en funcionamiento, ya que impide que los clientes accedan a la base de datos. En estos casos es más conveniente *bgsave*, que realiza el proceso en un segundo plano.

Además de la posibilidad de hacer una copia de la base de datos, también existe el modo *append only file (AOF)*, en el que Redis guarda en un archivo cada acción realizada. Gracias a este archivo, si el servidor se averiará inesperadamente, se podría averiguar qué fue lo último que se hizo. Para activar el modo AOF[106], debe modificarse el archivo de configuración.

```
127.0.0.1:6397> config set appendonly yes
```

Crear Entradas Una vez hayas configurado Redis, ya puedes trabajar con la base de datos. Dispones para ello de varios tipos distintos de datos y de comandos.

[106] Si tus datos requieren la máxima seguridad, deberías activar el modo AOF y, además, hacer copias regulares de la base de datos: así será prácticamente imposible que pierdas tus datos. Estos procesos, no obstante, ralentizan en cierta medida el funcionamiento de la base de datos.

Strings Lo más fácil es crear una String[107], es decir, una cadena o secuencia de elementos. Para ello, utiliza el comando *set*.

```
127.0.0.1:6397> set foo "bar"
127.0.0.1:6397> set value 1
```

Si se solicitan ahora las entradas foo y value mediante el comando *get*, se mostrarán los valores correspondientes.

```
127.0.0.1:6397> get foo
"bar"
127.0.0.1:6397> get value
"1"
```

El comando para borrar una entrada es *del*.

```
127.0.0.1:6397> del foo
(integer) 1
127.0.0.1:6397> get foo
(nil)
```

Si no quieres crear muchas entradas usando una fila nueva cada vez, puedes usar la función avanzada *mset*. Para solicitar los valores de varios campos a la vez, también existe el comando *mget*.

```
127.0.0.1:6397> mset foo1 "bar1" foo2 "bar2" foo3 "bar3"
OK
127.0.0.1:6397> mget foo1 foo2 foo3
1) "bar1"
2) "bar2"
3) "bar3"
```

[107] No importa si se introducen los valores entre comillas o no. Para hacer el código más legible, puede ponerse el texto entre comillas y los valores numéricos sin ellas.

Listas Con Redis se pueden usar, además, otros tipos de datos. Algunos de los más populares para trabajar con la base de datos son, por ejemplo, las listas y los sets. Ambos son conjuntos de valores, pero, mientras que los sets no tienen un orden concreto, los valores de las listas están numerados. En una lista se pueden añadir, solicitar o borrar entradas.

```
127.0.0.1:6397> lpush mylist foo
(integer) 1
127.0.0.1:6397> lpush mylist bar
(integer) 2
127.0.0.1:6397> lrange mylist 0 10
1) "foo"
2) "bar"
127.0.0.1:6397> linsert mylist before "bar" "test"
(integer) 3
127.0.0.1:6397> lrange mylist 0 10
1) "foo"
2) "test"
3) "bar"
127.0.0.1:6397> lrem mylist 0 foo
(integer) 1
127.0.0.1:6397> lrange mylist 0 10
1) "test"
2) "bar"
```

En este ejemplo hemos añadido en primer lugar dos elementos a una lista (*lpush*) y luego hemos solicitado que se muestren. Con el comando *lrange* se indica qué segmento debe mostrarse (aquí del 0 al 10, pero pueden usarse también números negativos). A continuación, mediante el comando *linsert* hemos añadido un valor nuevo delante de uno que ya existía (también podría usarse *after*), con lo cual hemos cambiado la numeración. El comando *lrem* permite borrar de la lista entradas con un valor específico.

Sets Para los sets, Redis utiliza otros comandos, pero con resultados muy similares:

```
127.0.0.1:6397> sadd myset "foo"
(integer) 1
```

```
127.0.0.1:6397> sadd myset "bar"
(integer) 1
127.0.0.1:6397> smembers myset
1) "bar"
2) "foo"
127.0.0.1:6397> sismember myset "bar"
(integer) 1
127.0.0.1:6397> srem myset "bar"
(integer) 1
127.0.0.1:6397> smembers myset
1) "foo"
```

Con el comando *sadd* también se pueden integrar varios elementos en el set si se introducen en el comando uno detrás de otro. Para visualizar el set, basta con usar el comando *smembers* y el nombre del set en cuestión. El comando *sismember* permite, además, buscar una entrada concreta. De manera análoga a la lista, con *srem* se pueden borrar entradas sueltas.

Sin embargo, Redis también ofrece a los usuarios la posibilidad de utilizar sets en un formato ordenado.

```
127.0.0.1:6397> zadd mysortedset 1 "foo"
(integer) 1
127.0.0.1:6397> zadd mysortedset 2 "bar"
(integer) 1
127.0.0.1:6397> zadd mysortedset 2 "foobar"
(integer) 1
127.0.0.1:6397> zrange mysortedset 0 10
1) "foo"
2) "bar"
3) "foobar"
```

Para añadir elementos se utiliza, en este caso, el comando *zadd* y un *score* o puntaje. Mientras que los valores propiamente dichos no pueden aparecer más de una vez, con un *score* se puede indicar un mismo valor varias veces. El score no es, por lo tanto, una numeración directa dentro del set, sino una ponderación, de manera que todas las entradas con el puntaje o score2 aparecerán tras los que tengan el score1. Con el comando *zrange* se pueden visualizar todos los elementos o los que se seleccionen.

Hashes Un tipo especial de datos son los hashes: entradas individuales compuestas de varios valores, de manera similar a los sets y las listas, pero en los que cada valor va acompañado de una clave, formando así los llamados pares clave-valor o key-value.

```
127.0.0.1:6397> hset user1 name "bob" email "bob@example.com"
password "rK87_x"
OK
127.0.0.1:6397> hget user1 name
1) "bob"
127.0.0.1:6397> hgetall user1
1) "name"
2) "bob"
3) "email"
4) "bob@example.com"
5) "password"
6) "rK87_x"
127.0.0.1:6397> hvals user1
1) "bob"
2) "bob@example.com"
3) "rK87_x"
127.0.0.1:6397> hkeys user1
1) "name"
2) "email"
3) "password"
> hdel user1 password
(integer) 1
127.0.0.1:6397> hgetall user1
1) "name"
2) "bob"
3) "email"
4) "bob@example.com"
127.0.0.1:6397> del user1
(integer) 1
127.0.0.1:6397> hgetall user1
(empty list or set)
```

En este ejemplo, hemos usado *hset* para crear un hash con el nombre user1 y tres campos. Mediante el comando *hget* podemos solicitar el valor de cada campo. Para que se muestren todos, se puede usar *hgetall*. Otras opciones para visualizar valores son *hvals* (muestra todos los valores guardados en el hash) y *hkeys* (muestra todas las claves guardadas en el hash). Con *hdel* se pueden borrar valores sueltos, mientras que con *del*, como ya hemos visto, se borra el hash entero[108].

Otras opciones Naturalmente, con Redis no solo se pueden crear entradas en una base de datos, sino que también pueden asignarse propiedades concretas a los datos. En este sentido, pueden resultar muy útiles, por ejemplo, los comandos de incremento y decremento.

```
127.0.0.1:6397> set foo 1
OK
127.0.0.1:6397> get foo
"1"
127.0.0.1:6397> incr foo
(integer) 2
127.0.0.1:6397> incr foo
(integer) 3
127.0.0.1:6397> get foo
"3"
127.0.0.1:6397> decr foo
(integer) 2
127.0.0.1:6397> get foo
"2"
```

Con la ayuda de estas funciones, se pueden incrementar o reducir los valores en una unidad. A veces, en cambio, se quieren introducir valores que solo permanezcan en la base de datos durante cierto tiempo: para ello existe la función expire.

```
127.0.0.1:6397> set foo "bar"
OK
127.0.0.1:6397> expire foo 100
(integer) 1
```

[108] El comando flushall sirve para borrar todas las entradas de la base de datos.

```
127.0.0.1:6397> ttl foo
(integer) 50
127.0.0.1:6397> ttl foo
(integer) -50
127.0.0.1:6397> get foo
(nil)
```

El comando expire requiere una indicación de tiempo en segundos. En este ejemplo, hemos decidido que la entrada debe durar 100 segundos. Una vez transcurrida la mitad del tiempo, hemos usado el comando *ttl* para solicitar el time-to-live, es decir, el tiempo restante. Si esperamos aún más, el TTL pasará a ser negativo a partir del momento en el que la entrada ya haya desaparecido.

El comando *setex* permite asignar un TTL a una entrada de la base de datos ya desde su creación.

```
127.0.0.1:6397> setex foo 100 "bar"
OK
```

Una vez se ha creado una entrada, esta se puede ampliar: el comando *append* añade otro valor al que ya existía.

```
127.0.0.1:6397> set foo "Hello"
OK
127.0.0.1:6397> append foo " World"
(integer) 11
127.0.0.1:6397> get foo
"Hello World"
127.0.0.1:6397> set bar 5
OK
127.0.0.1:6397> append bar 10
(integer) 3
127.0.0.1:6397> get bar
"510"
```

Como se puede ver, al solicitar los valores correspondientes, aparecen los nuevos elementos simplemente tras los que ya estaban. Si la entrada en cuestión aún no existe, *append* realiza entonces la misma función que *set*.

Además, también se puede cambiar el nombre de las entradas usando el comando *rename*.

```
127.0.0.1:6397> set foo 100
OK
127.0.0.1:6397> rename foo bar
OK
127.0.0.1:6397> get foo
(nil)
127.0.0.1:6397> get bar
"100"
```

Existen muchos otros comandos para trabajar con Redis. En la descripción oficial se pueden consultar los detalles de todos los comandos disponibles.

9.7 LAMP y LEMP

LAMP y LEMP son acrónimos usados para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

Para LAMP:

- Linux, el sistema operativo
- Apache, el servidor de Web
- MySQL/MariaDB, el gestor de bases de datos
- PHP, el lenguaje de programación

Para LEMP

- Linux, el sistema operativo
- Nginx, el servidor de Web
- MySQL/MariaDB, el gestor de bases de datos
- PHP (Hypertext Preprocessor), el lenguaje de programación

La combinación de estas tecnologías es usada principalmente para definir la infraestructura de un servidor Web, utilizando un paradigma de programación para el desarrollo del sistema. A pesar de que el origen de estos programas de código abierto no fue específicamente diseñado para trabajar entre sí, la combinación se popularizó debido al bajo costo de adquisición y ubicuidad de sus componentes.

Para instalar apache2 usamos:

```
# apt install apache2 libapache2-mod-evasive apachetop
```

Para instalar Nginx usamos:

```
# apt install nginx
```

En este caso tenemos dos opciones, instalar MariaDB que ya es parte integral de múltiples aplicaciones dentro de GNU/Linux o usar MySQL, en cada caso el uso es prácticamente el mismo.

Para instalar el paquete MariaDB usamos:

```
# apt install mariadb-server mariadb-client
```

y para configurarlo, usamos:

```
# mysql
> use mysql
> update user set plugin=" where user='root';
> alter user root@localhost identified by 'XXXXXXX';
> flush privilegies;
> exit
```

y para usarlo

```
$ mysql -u root -p
```

Para instalar el paquete MySQL usamos:

```
# apt install mysql-common mysql-client mysql-server mytop
mysql-admin
```

y para configurarlo (las bases de datos se guardan en /var/lib/mysql), usamos:

```
# mysql_secure_installation
```

o

```
# mysql
```

en caso de no crear contraseña, podemos ingresar así:

```
$ mysql -u root -p
```

una vez en la consola del cliente de MySQL creamos un nuevo usuario

```
> create user nombre identified by 'XXXXX';
```

y a continuación le damos privilegios totales

```
> gran all privileges on *.* to nombre with grant option;
> flush privileges;
> exit;
```

si requerimos las pruebas de eficiencia, podemos instalar:

```
# apt install ferret mysql-workbench mysql-workbench-data
mycli
```

Para instalar la interface para PHP, usamos:

```
# apt install php libapache2-mod-php php-mysql php-gd ph-
pmyadmin
```

9.8 Lenguajes de Programación y Bases de Datos SQL

Quizás te estés preguntando, ¿por qué debería preocuparme por conectar un lenguaje de programación como Java o Python y una base de datos SQL?. Hay muchos casos de uso en los que alguien querría conectar Java o Python a una base de datos SQL. Es posible que se esté trabajando en una aplicación Web. En este caso, necesitaríamos conectar una base de datos SQL para poder almacenar los datos provenientes de la aplicación Web. Quizás estemos trabajando en ingeniería de datos y necesitemos crear una canalización ETL automatizada. Conectar Java o Python a una base de datos SQL nos permitirá utilizar las capacidades de automatización del lenguaje de programación. También podremos comunicar entre diferentes fuentes de datos, no tendremos que cambiar entre diferentes lenguajes de programación.

Conectar Java o Python y una base de datos SQL también hará que nuestro trabajo de ciencia de datos sea más conveniente. Podremos utilizar nuestras habilidades en el lenguaje de programación para manipular datos de una base de datos SQL. No necesitaremos un archivo *CSV*.

Bases de Datos Relacionales las bases de datos de Software libre más utilizadas son PostgreSQL, MariaDB y MySQL que rivalizan con la base de datos Oracle comercial. PostgreSQL es adecuada incluso para organizaciones y proyectos de gran tamaño.

Otras bases de datos relevantes son H2 una base de datos implementada en Java con características avanzadas que es posible utilizar para los tests de integración al no requerir de un servidor y ser posible ejecutarla en memoria. Para realizar las pruebas de integración utilizando la misma base de datos que en producción es posible utilizar *Testcontainers* que utiliza Docker para iniciar una instancia de la base de datos en un contenedor. La conexión a la base de datos por seguridad requiere de un usuario y contraseña que la aplicación ha de conocer, para aún mayor seguridad es posible generar las credenciales de conexión a la base de datos de forma dinámica por *Vault* en una aplicación de *Spring*.

El problema de Seguridad de SQL Injection es un grave problema de seguridad que afecta a las aplicaciones que construyen sentencias de forma dinámica a partir de datos provenientes de origen no confiable. Un origen no confiable es cualquier dato proveniente de forma externa a la aplicación, en el caso de las aplicaciones Web o servicios *REST* es un parámetro de la petición o un dato de un *JSON*.

El SQL injection es un grave problema de seguridad ya que permite al atacante tener acceso a datos de la base de datos, obtener acceso con una cuenta de otro usuario o realizar operaciones sobre los datos de forma no autorizada. El problema se produce en la construcción de forma dinámica de la sentencia SQL mediante la concatenación de cadenas y datos de origen no confiable.

9.8.1 Java y el Acceso a Base de Datos Relacionales

Java ofrece soporte para las bases de datos relacionales desde prácticamente las primeras versiones del JDK hasta día de hoy incorporando un conjunto de clases en el paquete *java.sql* en la denominada *API* en Java de *Java Database Connectivity* o *JDBC*.

Las principales clases de la *API* de *JDBC* son las clases *Statement* y *PreparedStatement* que representan la sentencia SQL ya sea de inserción, actualización, eliminación o consulta así como las sentencias *DDL* para la

creación de tablas, índices, campos o procedimientos almacenados. Normalmente se utiliza la clase *PreparedStatement* ya que tiene beneficios en cuanto a rendimiento al ejecutarse de forma repetida y utilizada correctamente permite evitar el grave problema de seguridad de SQL injection común en las aplicaciones que construyen de forma dinámica con concatenaciones sentencias SQL utilizando datos procedentes de fuentes no confiables como parámetros de una petición *HTTP*, de *JSON* u otras fuentes externas a la aplicación.

La clase *ResultSet* es la clase que proporciona el acceso a los datos cuando se ejecuta una sentencia SQL de consulta, la clase se itera en los resultados y se obtienen los datos según los nombres o índices asignados en la consulta para las columnas.

La clase *Connection* representa una conexión a una base de datos, hay que crear una conexión ya que las bases de datos trabajan con una arquitectura de cliente y servidor, la aplicación actúa de cliente y la base de datos actúa de servidor. A través de la clase *Connection* se inicia una transacción y finaliza con el *commit* o el *rollback*. Para crear la conexión de la base de datos se proporcionan las credenciales de usuario y contraseña.

Las bases de datos relacionales soportan transacciones para proporcionar las propiedades *ACID* para lo que se utilizan las transacciones. Atomicidad donde un grupo de operaciones individuales se ejecutan todas o ninguna, consistencia mediante la cual los cambios son válidos según las reglas incluyendo restricciones, cambios en cascada, y disparadores, aislamiento donde los cambios de una transacción no se ven afectados por los cambios realizados en otras transacciones concurrentes y finalmente durabilidad que garantiza que en caso de completarse la transacción perdura en el tiempo aún cuando el sistema sufra un fallo posterior.

Crear una conexión a una base de datos es costoso, para evitar incurrir en este tiempo de creación y destrucción de conexiones o limitar el número de conexiones que una aplicación utiliza como máximo las aplicaciones utilizan un pool de conexiones. Las conexiones se crean al iniciar la aplicación o bajo demanda según se van necesitando más hasta el límite máximo definido. Cuando la aplicación necesita una conexión la obtiene de forma rápida del pool de conexiones y cuando termina de utilizarla la devuelve al pool de conexiones para que sea reutilizada en posteriores usos.

Cada base de datos utiliza un protocolo diferente de comunicación con los clientes por lo que es necesario un componente que abstrae de las peculiaridades de cada base de datos y proporcione un marco común de trabajo

independiente de cada base de datos. Cada base de datos requiere de un *Driver* compatible también con la versión de la base de datos. Generalmente, son los desarrolladores de la propia base de datos los que proporcionan un *Driver* específico adecuado para el acceso a la base de datos desde Java que cumple con las *APIs* de *JDBC*.

Ejemplo de Conexión y Consulta a un Base de Datos Relacional con la API de Java En este ejemplo de código se muestra el uso de las clases fundamentales de Java para usar una base de datos relacional. El primer paso es establecer una conexión con la base de datos, en este caso usando la base de datos H2 en memoria.

Posteriormente se ejecuta una sentencia *DDL* para crear una tabla, se insertan varias filas con la sentencia *insert* y se obtienen los datos de la tabla con una sentencia *select*. Después de las inserciones se realiza un commit que completa una transacción en la base de datos, si en vez del *commit* se hiciese un *rollback* al obtener los datos con la consulta posterior la tabla aparecería vacía.

Las clases *Connection*, *Statemente*, *PreparedStatement* y *ResultSet* al finalizar su uso hay que invocar su método *close* para liberar los recursos que tienen reservados, especialmente en el caso de las conexiones ya que son un recurso limitado. Estas clases implementan la interfaz *AutoCloseable* con lo que son adecuadas para las sentencias *try-with-resources* de Java.

Establecer la Conexión a la Base de Datos por defecto después de cada sentencia Java emite un *commit*, esto no es lo deseado en el caso de querer agrupar la ejecución de varias sentencias en una transacción, para evitarlo hay que usar la opción *setAutoCommit* a *false*.

```
package io.github.picodotdev.blgbitix.javasql;

import java.math.BigDecimal;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.DecimalFormat;

import java.util.Locale;

public class Main {
```

```
        public static void main(String[] args) {
            DriverManager.drivers().forEach(d -> {
                System.out.printf("Driver: %s%n", d.getClass().getName());
            });
            try (Connection connection = DriverManager.getConnection("jdbc:h2:mem:database",
"sa", "")) {
                connection.setAutoCommit(false);
            ...
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

Ejecutar una Sentencia con Statement

```
    Statement statement = connection.createStatement();
    statement.execute("CREATE TABLE product(id INT IDENTITY NOT NULL PRIMARY KEY,
name VARCHAR(255), price DECIMAL(20, 2))");
```

Ejecutar una Sentencia con PreparedStatement

```
    PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO product
(name, price) values (?, ?)", new String[] { "id" });
    preparedStatement.setString(1, "PlayStation 5");
    preparedStatement.setBigDecimal(2, new BigDecimal("499.99"));
    preparedStatement.executeUpdate();
    ResultSet resultSet1 = preparedStatement.getGeneratedKeys();
    while (resultSet1.next()) {
        System.out.printf("Primary key: %s%n", resultSet1.getLong(1));
    }
    resultSet1.close();
    preparedStatement.setString(1, "Xbox Series X");
    preparedStatement.setBigDecimal(2, new BigDecimal("499.99"));
    preparedStatement.executeUpdate();
    ResultSet resultSet2 = preparedStatement.getGeneratedKeys();
    while (resultSet2.next()) {
        System.out.printf("Primary key: %s%n", resultSet2.getLong(1));
```

```
        }
        resultSet2.close();
        connection.commit();
```

Ejecutar una Consulta

```
        PreparedStatement preparedStatement = connection.prepareStatement("SELECT id, name, price
    FROM product");
        ResultSet resultSet = preparedStatement.executeQuery();
        while (resultSet.next()) {
            System.out.printf("Product (id: %s, name: %s, price: %s)%n", resultSet.getLong(1), result-
    Set.getString(2),
            DecimalFormat.getCurrencyInstance(new Locale("es", "ES")).format(resultSet.getBigDecimal(3)));
        }
        resultSet.close();
```

Cerrar la Conexión de Forma Explícita

```
connection.close();
```

Resultado el resultado del programa en la terminal es el siguiente.

```
Driver: org.h2.Driver
Primary key: 1
Primary key: 2
Product (id: 1, name: PlayStation 5, price: 499,99 EUR)
Product (id: 2, name: Xbox Series X, price: 499,99 EUR)
```

Dependencia con el Driver de la Base de Datos en el archivo de construcción hay que añadir la dependencia que contiene el Driver para la base de datos a conectarse.

```
plugins {
    id 'application'
}
repositories {
    mavenCentral()
}
```

```
dependencies {
    implementation 'com.h2database:h2:1.4.200'
}
application {
    mainClass = 'io.github.picodotdev.blgbitix.javasql.Main'
}
```

El problema de Seguridad de SQL Injection Las siguientes sentencias SQL sufren del problema de SQL injection, en la primera un atacante puede obtener un dato de cualquier columna de la tabla y con la segunda ejecutar una sentencia en este caso para eliminar todas las filas de cualquier tabla.

En la primera cambiando el valor de column se obtiene el dato de cualquier columna de la tabla, por ejemplo el campo password.

```
"SELECT id, " + column + " FROM users WHERE user_id = " + userId
"SELECT id, password FROM users WHERE user_id = 1
```

En esta sentencia se finaliza la sentencia original con ; y se inicia otra lo que provoca la eliminación de una tabla, con un valor para user_id especialmente construido para ejecutar la sentencia maliciosa de eliminación de la tabla.

```
"SELECT * FROM users WHERE user_id = " + userId
"SELECT * FROM users WHERE user_id = 105; DROP TABLE users;"
```

La solución al problema de seguridad de SQL *injection* en Java es no construir la sentencia de forma dinámica mediante concatenación de cadenas utilizando la clase *PreparedStatement* con argumentos para los datos que se inserten en la sentencia SQL.

```
    PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM users
WHERE user_id = ?"});
    preparedStatement.setInt(1, 1);
    preparedStatement.executeUpdate();
```

Librerías de Persistencia en Java Habitualmente no se utilizan directamente las clases de la API de Java sino que se utilizan otras librerías de más alto nivel. Una de las más conocidas es *Hibernate*, es un ORM que proporciona acceso a los datos con una correspondencia entre el modelo relacional de las bases de datos y el modelo de objetos de Java. La aplicación trabaja con objetos y relaciones entre los objetos e *Hibernate* se encarga de transformar esos objetos en el modelo relacional de la base de datos, la aplicación no ejecuta sentencias SQL de forma directa sino que es *Hibernate* el encargado de emitir las sentencias adecuadas según los cambios realizados en los datos. Es la implementación más utilizada de ORM en Java para la especificación JPA.

- Aplicación de ejemplo con Hibernate, jOOQ y Liquibase

Spring Data es una capa de abstracción para el acceso a datos ya sean de un modelo relacional, de bases de datos NoSQL o algunos otros sistemas de datos. Spring Data hace más sencillo el acceso a los datos utilizando de forma subyacente JDBC o JPA. Spring Data proporciona algunas clases e interfaces que la aplicación implementa.

jOOQ también es otra librería de acceso a bases de datos relacionales, proporciona DSL para la construcción de sentencias SQL mediante un API Java. A diferencia de JPA con su lenguaje JPQL, jOOQ soporta características avanzadas del lenguaje SQL como windows functions. Otra ventaja de jOOQ es que al utilizar su DSL el compilador de Java realiza validación de tipos y comprobaciones en la sintaxis de la construcción de la SQL.

- Alternativa a Hibernate u ORM y ejemplo de jOOQ

Algunas aplicaciones combinan el uso de varias de estas librerías en la misma aplicación según el caso, por ejemplo utilizando Hibernate para el modelo de escritura y jOOQ o Spring Data para el modelo de lectura. También es posible utilizar jOOQ para generar las sentencias SQL y posteriormente ejecutarlas con Hibernate o Spring Data.

Liquibase es otra librería de utilidad que permite lanzar scripts de migración con sentencias SQL para realizar cambios en la base de datos como modificar el esquema de tablas, insertar, actualizar o eliminar datos. Es necesario si los cambios en el código en una nueva versión de la aplicación requiere cambios en el esquema de la base de datos. Estas no son las únicas librerías existentes pero sí son de las más conocidas y utilizadas.

9.8.2 Python y el Acceso a Base de Datos Relacionales

Las bases de datos de Python y SQL se conectan a través de bibliotecas personalizadas de Python. Podemos importar estas bibliotecas en su secuencia de comandos de Python. Las bibliotecas de Python específicas de la base de datos sirven como instrucciones complementarias. Estas instrucciones guían a nuestra computadora sobre cómo puede interactuar con un base de datos SQL.

Cómo Configurar el Proyecto Tomemos una base de datos PostgreSQL, *AWS Redshift*, por ejemplo. Primero, necesitamos importar la biblioteca *psycopg*. Es una biblioteca de Python universal para bases de datos de por ejemplo PostgreSQL:

```
#Library for connecting to AWS Redshift
import psycopg
#Library for reading the config file, which is in JSON
import json
#Data manipulation library
import pandas as pd
```

Notarás que también importamos las bibliotecas *JSON* y *pandas*. Importamos *JSON* porque crear un archivo de configuración *JSON* es una forma segura de almacenar las credenciales de su base de datos. ¡No queremos que nadie más los mire!.

La biblioteca de *pandas* nos permitirá utilizar todas las capacidades estadísticas de *pandas* para su secuencia de comandos de Python. En este caso, la biblioteca permitirá que Python almacene los datos que devuelve su consulta SQL en un marco de datos.

A continuación, necesitamos acceder al archivo de configuración. La función *json.load()* lee el archivo *JSON* para que pueda acceder a las credenciales de la base de datos en el siguiente paso:

```
config_file = open(r"C:\Users\yourname\config.json")
config = json.load(config_file)
```

Ahora que la secuencia de comandos de Python puede acceder a el archivo de configuración *JSON*, queremos crear una conexión de base de datos. Deberá leer y usar las credenciales de su archivo de configuración:

```
    con = psycopg2.connect(dbname= "db_name", host=config[hostname], port = config["port"],user=config["user_id"],
password=config["password_key"])
    cur = con.cursor()
```

¡Acabamos de crear una conexión a la base de datos! Cuando importamos la biblioteca *psycopg*, tradujo el código Python que escribió anteriormente para hablar con la base de datos PostgreSQL (AWS Redshift).

En sí mismo, AWS Redshift no entendería el código anterior. Pero debido a que importó la biblioteca *psycopg*, ahora habla un idioma que AWS Redshift puede entender. Lo bueno de Python es que tiene bibliotecas para SQLite, MySQL y PostgreSQL. Podremos integrar las tecnologías con facilidad.

Cómo Escribir una Consulta SQL siéntase libre de descargar European Soccer Data a su base de datos PostgreSQL. Usaré sus datos para este ejemplo.

La conexión de base de datos que creó en el último paso le permite escribir SQL para luego almacenar los datos en una estructura de datos compatible con Python. Ahora que hemos establecido una conexión a la base de datos, podemos escribir una consulta SQL para comenzar a extraer datos:

```
query = "SELECT *
FROM League
JOIN Country ON Country.id = League.country_id;"
```

Sin embargo, el trabajo aún no está terminado. Necesitamos escribir código Python adicional que ejecute la consulta SQL:

```
#Runs your SQL query
execute1 = cur.execute(query)
result = cur.fetchall()
```

Luego, debemos almacenar los datos devueltos en un marco de datos de pandas:

```
#Create initial dataframe from SQL data
raw_initial_df = pd.read_sql_query(query, con)
print(raw_initial_df)
```

Ahora deberíamos obtener un marco de datos de pandas (raw_initial_df).

9.9 Seguridad en Bases de Datos

Las fugas de datos son extremadamente prominentes en el mundo cibernético debido a la falta de una implementación de seguridad adecuada. Proteger las bases de datos es una práctica esencial para garantizar que se eviten consecuencias como la pérdida de datos incluso por acceso no autorizado o el tiempo de inactividad del sistema.

Hay muchos desafíos cuando se trata de la seguridad de la base de datos, ya que cuanto más accesible es una base de datos, menos segura es. Algunos de los desafíos de seguridad comunes cuando se trata de la seguridad de la base de datos incluyen errores humanos, Malware, seguridad de la ubicación física y vulnerabilidades de Software. Sin embargo, hay muchos pasos que se pueden tomar para garantizar que una base de datos esté protegida y, por lo tanto, evitar cualquiera de los problemas mencionados anteriormente. Además, el cifrado de la base de datos es un paso esencial para proteger todos los datos.

El Software libre tiene una variedad de herramientas que pueden usarse para garantizar la seguridad de la base de datos. Brevemente presentaremos algunas de estas herramientas que se pueden usar para la evaluación de bases de datos y otras que ayudarán a proteger estas bases de datos.

Prácticas Recomendadas para Bases de Datos Algunas de las amenazas de seguridad comunes incluyen ataques DDoS, ataques de inyección SQL, ataques Man in the Middle, contraseñas débiles, corrupción de datos, condición de carrera, mala gestión del acceso a la cuenta, así como vulnerabilidades ocasionales. Afortunadamente, hay muchas prácticas que se pueden adoptar para poder evitar estos problemas de seguridad.

9.9.1 Cifrado

El cifrado es una de las prácticas más efectivas e importantes que siempre debe usarse al almacenar datos. Algunas bases de datos pueden cifrar los datos directamente o cifrar el contenedor en el que se encuentran. Para MySQL, algunas de las herramientas que se pueden usar para el cifrado son el cifrado asimétrico, el cifrado simétrico, la generación de claves públicas/privadas, las firmas digitales y los datos transparentes. En cuanto a MariaDB, hay cifrado de datos en reposo, cifrado de datos en tránsito, certificados TLS/SSL y Cipher Block Chaining.

Además, para garantizar la seguridad de la base de datos, otras prácticas útiles a tener en cuenta son el acceso condicional y la auditoría. Para el acceso condicional, siempre es útil aplicar el privilegio mínimo. Es muy importante otorgar sólo la menor cantidad de privilegios necesarios para que un usuario pueda completar el trabajo. En cuanto a la auditoría, existe una variedad de herramientas que se pueden usar para auditar bases de datos como ClusterControl y Cloud SQL para MySQL, mientras que MariaDB tiene su propio complemento de auditoría que se puede usar.

9.9.2 Restringir el Acceso y Personalizar la Configuración Predeterminada

Los próximos pasos a seguir para proteger una base de datos MySQL deben incluir la eliminación de cuentas predeterminadas, asignaciones de puertos y la personalización de la configuración predeterminada. Es muy importante eliminar la base de datos de prueba ya que todos los usuarios tienen acceso completo a ella.

El acceso remoto también debe estar restringido. El acceso a la red solo debe permitir el mínimo requerido, como es el caso de muchas prácticas de seguridad, incluido el enfoque de privilegio mínimo mencionado. Todo acceso remoto debe ser monitoreado y controlado.

9.9.3 Asegúrese de que los Servidores sean Físicamente Seguros

Un paso importante que con frecuencia se ignora es asegurarse de que los servidores estén físicamente seguros. Si está utilizando proveedores de la nube como AWS, GCP o Azure, esto está cubierto automáticamente.

Uno de los pasos más esenciales para asegurar cualquier cosa, no solo las bases de datos, es garantizar el uso de credenciales de autenticación sólidas. Las contraseñas débiles pueden ser fácilmente forzadas otorgando a los atacantes acceso a los servidores. Por lo tanto, debemos asegurarnos de que solo se utilicen contraseñas seguras que contengan una combinación de caracteres en mayúsculas y minúsculas, números y caracteres especiales.

Además, authn y authz, que se refieren a autenticación y autorización, nos permiten controlar el acceso de los usuarios a los elementos dentro de la base de datos.

9.9.4 Usar Herramientas de Evaluación de la Seguridad de la Base de Datos

Un paso fundamental para proteger una base de datos es utilizar herramientas de evaluación de la seguridad de la base de datos. Estas herramientas revisan el entorno de una base de datos y reconocen las amenazas que están presentes en el entorno. Además de resaltar las vulnerabilidades existentes presentes en el entorno, estas herramientas también evalúan las medidas de seguridad implementadas para verificar su efectividad. Presentamos cinco de las mejores herramientas de evaluación de seguridad de bases de datos de Linux para garantizar que tenga una base de datos segura.

SQLMap es una herramienta de inyección SQL de código abierto basada en Python. Esta es una gran herramienta para usar, ya que ofrece una variedad de características. Estas características incluyen la detección y explotación de diferentes tipos de ataques de inyección, Hash de contraseñas de fuerza bruta, carga o descarga de archivos de la base de datos, se pueden usar para conectarse directamente a la base de datos a través de la inyección de SQL y se pueden personalizar.

SQLNinja es una herramienta basada en Perl que se puede utilizar con aplicaciones que utilizan servidores Microsoft SQL como Backend para explotar las vulnerabilidades de inyección de SQL. Las características de SQLNinja incluyen la capacidad de tomar huellas dactilares de servidores SQL remotos, extracción de datos, carga de ejecutables, explotación de vulnerabilidades de inyección para obtener acceso a la base de datos, contraseñas de fuerza bruta mediante ataques de diccionario y ataques de escalada de privilegios si la fuerza bruta tiene éxito.

BBQSQL como SQLMap, es una herramienta de código abierto basada en Python. BBQSQL se usa para explotar las vulnerabilidades de inyección de SQL y se enfoca en la vulnerabilidad ciega de SQL. Esta herramienta requiere cierta información para ser utilizada, que incluye URL, método HTTP, Cookies, etc. Sin embargo, es una gran herramienta ya que es muy rápida para encontrar y explotar vulnerabilidades, es personalizable, realiza la validación de entrada en todas las opciones de configuración e incluso puede parchear las vulnerabilidades detectadas

JSQL Injection como su nombre lo indica, también es una herramienta de inyección SQL basada en Java. Se puede utilizar para encontrar y explotar vulnerabilidades de inyección. Se puede utilizar en 33 motores de bases de datos, realizar varios tipos de ataques de inyección, crear un Shell Web y un Shell SQL en un servidor remoto, utilizar Hashes de fuerza bruta y mucho más.

OScanner es un marco de evaluación de Oracle basado en Java. Se puede utilizar para enumeración de SID, contraseñas de fuerza bruta, versión de enumeración, roles de cuenta, privilegios y políticas de contraseña, información de auditoría y enlaces a bases de datos. OScanner da los resultados en un formato de árbol java gráfico.

Estas herramientas que hemos mencionado son fundamentales para garantizar que su base de datos esté segura. Se pueden usar para verificar qué pasos se deben tomar para garantizar que las bases de datos sean seguras, y también se pueden usar después de tomar estos pasos para evaluar la efectividad del procedimiento seguido.

9.9.5 Pruebas de Penetración

Cuando se trabaja con bases de datos que envían y reciben peticiones desde páginas Web o aplicaciones en internet, es necesario hacer pruebas de penetración (o Pentest) que nos permiten analizar los riesgos de los programas desarrollados y del sistema completo antes de entrar en la fase de producción.

Hay tres tipos de pruebas de penetración:

- Prueba de caja negra (Blackbox): Tratamos de atacar el sistema sin el conocimiento de la infraestructura y el funcionamiento del mismo. Esta es la forma menos efectiva de las pruebas Pentest.

- Pruebas de caja gris (Greybox): Tratamos de atacar el sistema con poco conocimiento de la infraestructura y el funcionamiento interno del mismo. Las pruebas de caja gris son más efectivas que las de caja negra.

- Pruebas de caja blanca (Whitebox): Tratamos de atacar el sistema con toda la información sobre la infraestructura y código fuente. Esta es la forma más efectiva de las pruebas Pentest.

Las pruebas de penetración tienen como objetivo prevenir ciberataques. ¿Qué es un ataque cibernético?

- Ataque cibernético: Es el que se ejecuta a través del espacio cibernético con el objetivo de dislocar, dañar o apropiarse de la gestión de la infraestructura de una empresa o dependencia gubernamental.
- Superficie del ataque cibernético: Es el ámbito en el cual se ataca a un sistema de cómputo. Esto puede ser muy amplio y abarca las irrupciones en las redes, los protocolos, el sistema operativo a las aplicaciones.
- Exploit: Es un ataque contra los elementos vulnerables de un sistema llevado a cabo mediante el uso de Software, un comando o una metodología específica. Un Exploit puede ser un elemento Malware; para el Hackeo ético un Exploit es una forma de identificar vulnerabilidades del sistema.

Debido a que se practicará con aplicaciones inseguras, es importante crear un ambiente seguro, pero siempre es deseable hacer pruebas con toda la infraestructura computacional disponible sobre la que se montará el sistema desarrollado, pero en muchas ocasiones esto no es posible o viable, por ello se opta por generar un sistema de prueba aislado y seguro usando máquinas virtuales[109] con la misma configuración que el sistema de interés.

Dependiendo del tamaño del sistema, es posible usar uno o más equipos anfitriones (Hosts) que alberguen nuestras máquinas virtuales y usar la red física o virtual para interconectar las aplicaciones que se montarán en las máquinas virtuales para implementar el sistema de pruebas de penetración.

Por ejemplo, si todo nuestro sistema desarrollado cabe en un solo equipo de cómputo (no importa que sistema operativo tenga el anfitrión que usemos para las pruebas), entonces podríamos usar una primer máquina virtual[110] para instalar el ambiente de pruebas como por ejemplo Kali Linux[111], dentro

[109] Entendamos por una máquina virtual a un programa de cómputo que simula a una computadora, en la cual se puede instalar y usar otros sistemas operativos de forma simultánea como si fuese una computadora real sobre nuestro sistema operativo huésped.

[110] Existen una gran variedad de manejadores de máquinas virtuales como: QEMU/KVM, VirtualBox, Vmware, Parallels, Windows Virtual PC, Gnome Box, Xen, LXC, Docker, etc.

[111] Kali Linux (https://www.kali.org), es una distribución para pruebas de penetración estándar de la industria. Es una de las distribuciones más populares entre Pentesters,

de esta instalación instalamos otro manejador de máquinas virtuales para soportar el sistema operativo y la paquetería necesaria[112] que soporte el sistema a probar, dentro de esta máquina virtual además instalaremos el proyecto abierto para la seguridad de aplicaciones OWASP-BWA[113] (Web Open Web Application Security Project), que es un proyecto de código abierto para profesionales de la seguridad, el cual permite crear un ambiente de prueba seguro donde podemos identificar vulnerabilidades en las aplicaciones desarrolladas.

De esta forma podemos crear y usar un ambiente seguro para evitar los controles y detectar vulnerabilidades en los sistemas desarrollados e informar de los hallazgos de las pruebas de penetración antes de que estos se conviertan en brechas de seguridad y se generen daños por ellas.

9.10 Programando desde la Web

Existen diferentes servicios WEB que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y Terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Algunos ejemplos de estos servicios son:

9.10.1 SQLite

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.codechef.com/ide

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Hackers éticos e investigadores de seguridad en todo el mundo y contiene cientos de herramientas para el trabajo forense, esta distribución está basada en Debian.

[112] Puede ser por ejemplo Linux, Apache, MySQL, PHP, Perl, Python, etc.

[113] Es un proyecto comunitario que ayuda a organizaciones e instituciones educativas a diseñar, implementar y administrar aplicaciones seguras. Encontrando amenazas como: Inyección (SQL, OS, XXE, LDAP), irrupción en la autentificación y la sesión, Cross Site Scripting, irrupción en el control de acceso, errores en la configuración de la seguridad, exposición de datos sensibles, protección insuficiente contra ataques, uso de componentes con vulnerabilidades conocidos, API con protección insuficiente, etc.

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

9.10.2 MySQL y MariaDB

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.jdoodle.com/

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

9.10.3 PostgreSQL

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://rextester.com/l/postgresql_online_compiler

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text not null,
    precio money,
    precioVenta money,
    primary key (id_prod)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '12/01/2020 8:01:00.00'), (2, 5, '12/01/2020 10:15:00.00'),
(2, 4, '12/01/2020 13:34:00.00'), (1, 3, '12/01/2020 21:56:00.00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

9.10.4 MongoDB

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.jdoodle.com/online-mongodb-terminal/

Por ser un servicio en red, no es necesario conocer que bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las colecciones necesarias, agregamos

valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos el document de Productos e insertamos los valores mediante:

```
var miProductos =
[
    {"_id" : 1, "descripcion" : "Cigarros", "precio" : 25.5,
"precioVenta" : 30.0},
    {"_id" : 2, "descripcion" : "Refresco", "precio" : 4.5,
"precioVenta" : 6.0},
];
db.Productos.insert(miProductos);
db.Productos.find()
```

ahora, creamos documento de ventas e insertamos los valores mediante:

```
var miVentas =
[
    {"_id" : 1, "id_prod" : 1, "Cantidad" : 2, "Fecha" :
"2020/12/01 8:01:00"},
    {"_id" : 2, "id_prod" : 2, "Cantidad" : 5, "Fecha" :
"2020/12/01 10:15:00"},
    {"_id" : 3, "id_prod" : 2, "Cantidad" : 4, "Fecha" :
"2020/12/01 13:34:00"},
    {"_id" : 4, "id_prod" : 1, "Cantidad" : 3, "Fecha" :
"2020/12/01 21:56:00"},
];
db.Ventas.insert(miVentas);
db.Ventas.find()
```

de esta forma, ahora podemos usar Find para hacer el equivalente a un JOIN como en los ejemplos anteriores y mostrar el resultado de la solicitud, mediante:

```
db.Ventas.find().forEach(
    function (object) {
        var commonInBoth=db.Productos.findOne({ "_id":
object.id_prod} );
```

```
                if (commonInBoth != null) {
                    print(object.Fecha, object.Cantidad, common-
InBoth.descripcion, commonInBoth.precioVenta, (commonInBoth.precioVenta
* object.Cantidad));
                }
            }
        );
```

10 Intercambio de Información

Hay una gran variedad de tecnologías disponibles para producir y almacenar datos. Como son: hojas de cálculo, bases de datos, Software estadístico más específico y más. Esto genera una enorme diversidad de formatos, a veces esto es por decir lo menos, caótico. Algunos de estos formatos, no siempre se adecuan a los niveles de apertura deseados, aquí ofrecemos algunas pautas y recomendaciones que facilitan la adaptación y/o transformación de estos formatos hacia otros más abiertos y fácilmente reutilizables.

XML es un metalenguaje simple pero estricto, desarrollado por W3C. Desarrolla un papel fundamental en el intercambio de una gran variedad de datos. XML es un formato que permite la interpretación de datos a través de varias aplicaciones. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. En realidad, XML es una manera de definir lenguajes para diferentes necesidades.

```
<catalogo>
<libro>
<autor>Raúl González Duque</autor>
<titulo>Python para todos</titulo>
<genero>Computación</genero>
</libro>
</catalogo>
```

JSON es un formato de texto popular para el intercambio de datos, es un acrónimo de JavaScript Object Notation. Por su característica de ser un formato de tipo estructurado es especialmente útil para el intercambio de datos entre máquina (machine - readable format). El formato JSON ha sido definido por la especificación RFC 7159 y, tal como CSV, también es un estándar abierto.

```
{
    libro: {
        autor: Raúl González Duque,
        titulo: Python para todos,
        genero: Computación
    }
}
```

YAML es un lenguaje que fue diseñado para ser legible (y como tal será fácil de editar con cualquier editor de texto estándar). Su noción es a menudo similar a reStructuredText o una sintaxis Wiki, que también intentan ser legibles tanto por seres humanos y computadoras. YAML 1.2 incluye también una noción de taquigrafía que sea compatible con JSON, y como tal, cualquier documento JSON es también válido YAML; Sin embargo esto no se sostiene al revés.

```
---
libro:
    autor: Raúl González Duque
    titulo: Python para todos
    genero: Computación
```

KML - KMZ gramática XML y formato de archivo para la creación de modelos y almacenamiento de funciones geográficas como puntos, líneas, imágenes, polígonos y modelos que se mostrarán principalmente en aplicaciones de mapas. Se utiliza para compartir lugares e información entre aplicaciones. Es el estándar del Open Geospatial Consortium y se puede utilizar a través de Google Earth. Los archivos KML se distribuyen comprimidos como archivos KMZ.

SHP Shapefile es un formato propietario de datos espaciales que es el estándar para el intercambio de información geográfica entre Sistemas de Información Geográfica (GIS). Es un formato vectorial de almacenamiento digital donde se guarda la localización de elementos geográficos y los atributos asociados a ellos, pero sin capacidad para almacenar información topológica. Lo generan varios archivos, mínimo 03 y tiene 03 tipos de extensiones: .shp, .shx y .dbf

RDF un formato recomendado por W3C llamado RDF permite representar datos en una forma que facilita la combinación de información de diferentes fuentes. Los datos RDF pueden ser almacenados en XML y JSON, entre otras serializaciones. RDF promueve el uso de URLs como identificadores, lo que brinda una manera conveniente de interconectar iniciativas de "Datos Abiertos" existentes. RDF todavía no se ha generalizado, pero es una tendencia entre iniciativas de Gobierno Abierto, incluso las iniciativas

de Linked Data de los gobiernos británico y español. El inventor de la Web, Tim Berners-Lee, propuso recientemente un esquema de puntuación de cinco estrellas que incluye Linked Data en formato RDF como una meta a alzanzar en las iniciativas de Datos Abiertos.

REBOL es un lenguaje que fue diseñado para ser legible y fácil de editar con cualquier editor de texto estándar. Para lograr que utiliza una sintaxis simple de forma libre con puntuación mínima y un rico conjunto de tipos de datos. REBOL datatypes como direcciones URL, correos electrónicos, valores de fecha y hora, tuplas, cuerdas, etiquetas, etc. respetar las normas comunes. REBOL está diseñado para no necesitar ningún adicional metalenguaje, siendo diseñado de una manera metacircular. El metacircularity de la lengua es la razón por qué por ejemplo, utilizan el dialecto Parse (no exclusivamente) para las definiciones y transformaciones de dialectos REBOL constituye también un dialecto de REBOL. REBOL fue utilizado como una fuente de inspiración del diseñador de JSON.

Gellish inglés es un subconjunto del inglés natural, que incluye una gramática simple y una gran formalizado extensible Diccionario Inglés-taxonomía Eso define la terminología específica general y dominio (términos de conceptos), mientras que los conceptos se organizan en una jerarquía de subtipo-supertipo (una taxonomía), que apoya la herencia de los conocimientos y requisitos. El diccionario-taxonomía también incluye tipos estandarizados hechos (también llamados tipos de relación). Los términos y tipos de relación juntos pueden utilizarse para crear e interpretar las expresiones de hechos, conocimientos, requisitos y otras informaciones. Gellish puede ser utilizado en combinación con SQL, RDF/XML, BÚHO y varias otras lenguas meta. El Gellish estándar está siendo adoptada como ISO 15926-11.

PDF documento de formato portable universal que mantiene la apariencia del documento independientemente del sistema operativo que se utilice (multiplataforma). Incluye cualquier combinación de texto, multimedia e hipertexto y además se puede cifrar el contenido y firmarlo digitalmente. Es el estándar ISO, desde 2008, para ficheros contenedores de documentos electrónicos de cara a la conservación de larga duración. Es una especificación que se puede crear, visualizar o modificar con herramientas de Software libre. Este formato originalmente era propietario (hasta 2008).

Hojas de Cálculo muchas autoridades tienen información que queda en la hoja de cálculo, por ejemplo Microsoft Excel. Estos datos a menudo pueden ser utilizados inmediatamente con las descripciones correctas de lo que las distintas columnas significan. Sin embargo, en algunos casos puede ser macros y fórmulas en hojas de cálculo, las cuales pueden ser más incómodas para manipular. Por tanto, es recomendable documentar este tipo de cálculos junto a la hoja de cálculo, ya que generalmente es más accesible para los usuarios al leer.

Archivos separado por comas los archivos CSV pueden ser un formato útil debido a que son compactos y por lo tanto adecuado para transferir grandes conjuntos de datos con la misma estructura. Sin embargo, el formato es tan espartano que los datos son frecuentemente inservibles sin documentación, ya que puede ser casi imposible adivinar el significado de las diferentes columnas. Por tanto, es particularmente importante para los formatos separados por comas que la documentación de los campos individuales sea precisa.

Además, es esencial que la estructura del archivo sea respetada, como una omisión única de un campo puede perturbar la lectura de todos los datos restantes en el archivo sin ninguna posibilidad real de rectificarla, porque no se puede determinar cómo los datos restantes deben ser interpretados.

Documento de Texto documentos clásicos en formatos como Word, ODF, OOXML, o PDF, puede ser suficiente para mostrar ciertos tipos de datos, por ejemplo, listas de correo o equivalente. Puede que sea barato para exponerlos, siempre que ese sea el formato en que la data se haya creado. El formato no da soporte para mantener la estructura coherente, lo cual significa que a veces es difícil para introducir datos por medios automatizados. Asegúrese de utilizar las plantillas como base de documentos en las que se mostrarán los datos para su reutilización, por lo que es al menos posible extraer información de los documentos. También se puede apoyar el uso posterior de los datos que utilice el marcador de la tipografía tanto como sea posible para que sea más fácil para una máquina para distinguir las partidas (de cualquier tipo) a partir del contenido y así sucesivamente. En general se recomienda no exhibir en formato de procesamiento de texto, si hay datos en un formato diferente.

Texto los documentos de texto (.Txt) son muy fáciles de leer en las computadoras. Por lo general, no incluyen metadatos estructurales, lo que significa que los desarrolladores necesitan para crear un programa de análisis que pueda interpretar cada documento tal y como aparece. Algunos problemas pueden ser causados al cambiar archivos de texto plano entre sistemas operativos. MS Windows, Mac OSX y otras variantes de Unix tienen su propia forma de decirle a la computadora que se ha llegado al final de la línea.

Imagen escaneada probablemente la forma menos adecuada para la mayoría de los datos, pero ambos TIFF y JPEG 2000-al menos puede marcarlos con la documentación de lo que está en la imagen - adecuada para marcar la imagen de un documento con contenido de texto completo del documento. Puede ser relevante a los datos que muestran como imágenes aquellos datos que no nacen por la vía electrónica -un ejemplo obvio es la antigua escuela de registros y el material de archivo- y una imagen es mejor que nada.

Formatos propietarios algunos sistemas tienen sus propios formatos de datos que pueden guardar o exportar datos. A veces puede ser suficiente para exponer los datos en un formato - sobre todo si se espera que el uso de más estaría en un sistema similar del que vienen. Sobre dónde obtener más información sobre estos formatos, debe estar siempre indicado, por ejemplo, al proporcionar un enlace a la página web del proveedor. En general se recomienda mostrar los datos en formatos libres, cuando sea factible.

HTML hoy en día, muchos datos se encuentran disponibles en formato HTML en varios sitios. Esto sería suficiente, si se trata de datos estables y de alcance limitado. En algunos casos, sería preferible tener los datos de una forma fácil para descargar y manipular, pero como es barato y fácil de hacer referencia a una página en un sitio web, podría ser un buen punto de partida en la pantalla de datos.

Generalmente, sería más apropiado el uso de tablas en documentos HTML para almacenar datos y, es importante que los distintos campos de datos se muestren y sean dadas identificaciones, que hacen más fácil encontrar y manipular los datos. Yahoo ha desarrollado una herramienta

$$(https://developer.yahoo.com/yql/)$$

que puede extraer información estructurada de un sitio web, y ciertas herramientas que pueden hacer mucho más con los datos siempre que se encuentren cuidadosamente etiquetados.

HDF (https://www.hdfgroup.org/solutions/hdf5/)

En el ámbito científico, se genera gran cantidad de información en formato digital, que es necesario distribuir. Dada la diferente naturaleza de ésta, es necesario utilizar distintos formatos especialmente adaptados a cada uno de los tipos de datos que se vayan a almacenar. El principal problema radica en la estandarización y la interoperabilidad, ya que, pese a que existen numerosos estándares de almacenamiento de información, no todos son soportados por las distintas aplicaciones informáticas utilizadas, lo que puede presentar problemas en el momento de la difusión y reutilización de dicha información. El formato HDF (Hierarchical Data Format), representa una alternativa eficaz al ser adoptado como estructura de almacenamiento de datos, cuyas características más representativas son las siguientes:

- Permite obtener información acerca de los datos de un archivo desde dentro de ese archivo, sin necesidad de recurrir a fuentes externas.
- Permite almacenar datos de distinta naturaleza en un mismo archivo y relacionarlos entre ellos.
- Estandariza los formatos y las descripciones de los tipos de datos más comúnmente empleados.
- Se trata de un formato abierto, con sus especificaciones publicadas, lo que permite su implementación en diversas aplicaciones informáticas, facilitando la portabilidad, así como permitiendo al usuario desarrollar sus propias aplicaciones específicas.
- Es flexible y puede ser adaptado para almacenar cualquier tipo de dato.

Con el fin de dar respuesta a las necesidades de almacenamiento y difusión de datos científicos de diversa naturaleza, en un formato flexible e independiente de la plataforma, comenzó a desarrollarse el formato HDF en los laboratorios de la NCSA (National Center for Supercomputing Applications) a partir del año 1988, siendo soportado en la actualidad por el HDF Group, dependiente de la Universidad de Illinois. Actualmente, este formato

se utiliza por múltiples organismos, tanto públicos como privados, para la difusión de sus datos y resultados, entre las que se encuentran la NASA, la Agencia Europea del Espacio (ESA), que los aplican en datos procedentes de los sensores MODIS, MERIS o ETM+, entre otros.

La estructura de los archivos HDF permite el almacenamiento de diversos tipos de datos, tales como:

- Scientific Data Sets (SD), utilizados para almacenar matrices n-dimensionales de datos enteros (8, 16 y 32 bits, con o sin signo) o reales (32 o 64 bits) en su formato estándar, o por medio de las APIs, crear datos en otros rangos de valores (1 a 32 bits), junto con sus metadatos (dimensiones de la matriz, atributos, etc.)

- Raster Images (RI), que permiten almacenar imágenes de 8 bits (0-255 niveles de gris) o 24 bits (RGB), o bien utilizar nuevamente las librerías de programación para guardar imágenes en otros formatos (16 a 32 bits enteros, 32 a 64 bits coma flotante). Además, se permite almacenar información sobre las dimensiones de la imagen, así como la paleta de color asociada a ella. También es posible utilizar compresión (RLE, JPEG, GZIP y adaptativa Huffman) para reducir el tamaño de los archivos resultantes.

- Text Annotations (TA), para almacenar cualquier tipo de información textual: Etiquetas, descripciones o información de archivo.

- VDatas (VD), para almacenar datos vectoriales sin topología.

- Vgroups (VG), que nos permiten asociar datos relacionados dentro de un archivo.

10.1 Archivos de Formatos Abiertos

Incluso si la información se proporciona en formato electrónico, formato legible por máquina, y en detalle, puede haber problemas relacionados con el formato del archivo en sí. Los formatos en los cuales la información es publicada - en otras palabras, la base digital en la cual la información es almacenada - puede ser "abierto" o "cerrado". Un formato abierto es aquel donde las especificaciones del Software están disponibles para cualquier persona, de forma gratuita, así cualquiera puede usar dichas especificaciones en su propio

Software sin ninguna limitación en su reutilización que fuere impuesta por derechos de propiedad intelectual.

Si el formato del archivo es "cerrado", esto puede ser debido a que el formato es propietario y sus especificaciones no están disponibles públicamente, o porque el formato es propietario y aunque las especificaciones se han hecho públicas, su reutilización es limitada. Si la información es liberada en un formato de archivo cerrados, esto puede causar grandes obstáculos para reutilizar la información codificada en él, forzando a aquellos que deseen usar la información a comprar Software innecesario.

La ventaja de los archivos de formatos abiertos, es que permiten a los desarrolladores producir varios paquetes de Software y servicios utilizando esos formatos. Esto entonces reduce al mínimo los obstáculos para la reutilización de la información que contienen.

El uso de formatos de archivo con propiedad, para el que la especificación no está disponible públicamente, puede crear dependencia de Software de terceros o de los titulares de licencias de los formatos de archivos. En el peor de los casos, esto puede significar que la información sólo se puede leer con ciertos Softwares especiales, que pueden ser muy caros, o que pueden quedar obsoletos.

La preferencia del término Gobierno de Datos Abiertos, es que la información se publicará en formatos de archivo abiertos, los cuales son de lectura mecánica.

¿Cómo usar un formato determinado? cuando una autoridad debe mostrar nuevos datos -datos que no han sido expuestos antes- se debe elegir el formato que proporciona el mejor equilibrio entre el costo y la idoneidad para el propósito. Para cada formato, hay algunas cosas que usted debe tener en cuenta, y esta sección tiene como objetivo explicarlas.

Servicios Web cuando los datos cambian frecuentemente, y el tamaño de cada descarga es limitado, es muy importante exponer la información a través de web services. Hay muchas maneras de crear web services, pero alguno de los más usados son SOAP y REST. Generalmente SOAP antes que REST. Los servicios REST son fáciles de desarrollar y entender, por eso es un estándar muy usado.

Base de datos como los servicios webs, las bases de datos permiten el acceso directo a los datos dinámicamente. Las bases de datos tienen la ventaja de extraer aquella información en la cual estén interesados. Hay algunas preocupaciones de seguridad en cuanto a permitir extracciones remotas a bases de datos, y el acceso a las bases de datos es útil si su estructura y la importancia de las tablas y campos individuales están bien documentados. Generalmente es relativamente simple y no tiene costo crear web services que expongan datos de una base de datos, esta puede ser una manera fácil de lidiar con las problemáticas de seguridad.

Intercambio de datos único dominio a menudo hay un par de docenas de diferentes fuentes y esquema de destino (formatos de datos propietarios) en un dominio específico. A menudo las personas desarrollan una formato de intercambio de o formato de intercambio de para algunos un solo dominio y luego escribir unas docena diferentes rutinas para traducir (indirectamente) cada esquema de origen de cada esquema de destino utilizando el formato de intercambio como un paso intermedio. Eso requiere mucho menos trabajo que escribir y depurar los cientos de diferentes rutinas que se necesitaría para traducir directamente cada esquema de origen directamente a cada esquema de destino. (Por ejemplo, Formato de intercambio estándar para la infraestructura de datos espaciales, Formato de intercambio de datos para los datos de la hoja de cálculo, GPS eXchange Format o Keyhole Markup Language para indicar las coordenadas GPS del mundo, Formato de intercambio de Quicken para los datos financieros, GDSII diseño de circuitos integrados, etc.)

Intercambio de datos de idiomas un lenguaje de intercambio de datos es un lenguaje que es independiente del dominio y puede utilizarse para cualquier tipo de datos. Su capacidad de expresión semántica y sus cualidades se determinan en gran medida en comparación con las capacidades de lenguajes naturales. El término también se aplica a cualquier formato de archivo puede ser leído por más de un programa, incluyendo formatos propietarios como Microsoft Office documentos. Sin embargo, un formato de archivo no es un lenguaje real como carece de una gramática y vocabulario.

La práctica ha demostrado que ciertos tipos de lenguajes formales son los más adecuados para esta tarea que otros, puesto que su especificación es impulsado por un proceso formal en lugar de una implementación de Software

en particular las necesidades. Por ejemplo XML es un lenguaje de marcado fue diseñado para permitir la creación de dialectos (la definición de dominio específico sublanguages) y una opción popular ahora en particular en internet. Sin embargo, no contiene diccionarios específicos de dominio o tipos de hecho. Beneficiosa para un intercambio de datos fiable es la disponibilidad de diccionarios-taxonomías estándar y las bibliotecas de herramientas tales como Analizadores, esquema Validadores y herramientas de transformación.

Antes de seguir, introduciremos dos conceptos que se usarán a lo largo de toda la guía:

- Distribución o Recurso: Una distribución o recurso es la unidad mínima en la que se publican datos. Se trata de los archivos que pueden ser descargados y re-utilizados por un usuario. Los recursos pueden tener diversos formatos (.csv, .shp, etc.).

- Dataset: Un conjunto de datos o dataset agrupa recursos referidos a un mismo tema que respetan una estructura de la información. Los mismos podrían diferir en el formato en que se los presenta (Ej. .csv, .json, .xls, etc.), la fecha a la que se refieren, el área geográfica cubierta o estar separados bajo algún otro criterio.

Un recurso en formato tabular es un archivo plano que se ajusta a un esquema predefinido de columnas, incluyendo el nombre de la columna y el tipo de datos. En la mayoría de los casos, corresponde a datos que llegan de bases de datos, reportes y planillas de cálculo en general. A diferencia de los formatos tabulares, los archivos JSON siguen una estructura diferente donde se definen listas de objetos con pares "clave" - "valor".

Recomendamos con énfasis la publicación de los datos en formato CSV y/o JSON. En caso de utilizar formatos propietarios o aún no estandarizados, es útil indicar Software, versión y aplicación que permite procesar esos formatos.

CSV es un formato de archivo de texto plano donde las campos (columnas) se separan por comas; y los registros (filas), por saltos de línea. Algunas versiones alternativas de esta forma de publicar datos usan otros separadores como punto y coma (";") o pipe ("|"). Las recomendaciones, siguiendo lo indicado por la especificación RFC4180 y las pautas de la W3C, para la publicación de archivos en formato CSV son:

- Las filas deben finalizar con los caracteres de "retorno de carro" (\r) y "salto de línea" (\n) unidos (\r\n). Esta forma de separar las líneas se denomina usualmente CRLF ("Carriage Return Line Feed").
- La primera fila siempre contiene los nombres de los campos.
- No se deben repetir nombres entre los campos.
- No se debe colocar espacios al principio ni al final del nombre de un campo, o de un valor.
- Tanto los campos como los valores deben estar separados por comas (",").
- En el caso de que un valor contenga el carácter separador (",") o cualquiera de los caracteres que separan las líneas ("\r", "\n" o "\r\n"), el valor completo debe ser encerrado entre comillas dobles "". Esto indica que el carácter no cumple el rol de separar columnas o filas, sino que es parte de un valor. Ejemplo:

  col1,col2\r\n
  "La tasa de Juan, está vacía",La tasa de Pablo está llena\r\n
  "La tasa de Juan\nestá vacía",La tasa de Pablo está llena\r\n
  "La tasa de Juan\r\nestá vacía",La tasa de Pablo está llena\r\n

- En el caso de que un valor contenga el carácter comilla doble ("), el valor debe ser encerrado entre comillas dobles como en el caso anterior ("") y, además, los caracteres comilla doble que se encuentren dentro del valor deben escribirse dos veces (""). Ejemplo:

  col1,col2\r\n
  "La tasa de ""Juan"" está vacía",La tasa de Pablo está llena\r\n

- Para todos los tipos de datos se considera válido el valor indefinido. Este se expresará con la ausencia de todo carácter y no con un carácter o string especial como podrían ser ".", "null", "none", "nan", etc. Ejemplo:

  col1,col2,col3\r\n
  a„b\r\n
  a,"",b\r\n

Fragmentación de archivos Para garantizar la accesibilidad a los datos, es necesario fragmentar los archivos excesivamente grandes, que superen el millón de filas.

Para esto, recomendamos usar conceptos simples, fragmentando:

- Por períodos en caso de tratarse de información temporal (Ej. Años, semestres, trimestres, meses, semanas, días),
- Por zonas en caso de tratarse de información geográfica (Ej. provincias, municipios, barrios, secciones, o manzanas) o
- Por dimensiones temáticas propias del dominio particular de la información.

Sin embargo, siempre que se decida fragmentar un archivo para garantizar su accesibilidad, recomendamos publicar también una versión no fragmentada que contenga el conjunto de datos completo (aunque sea muy grande), a los fines de evitar la tarea de consolidación.

Para eso, sugerimos usar protocolos de compresión, en especial para archivos muy grandes, y altamente compresibles. De hacerlo, aconsejamos protocolos sin pérdida, y abiertos.

Nomenclatura de archivos Recomendamos estas convenciones para nombrar archivos:

- Usar palabras siempre en minúsculas.
- No usar artículos.
- Usar únicamente letras y números ASCII, siempre en minúsculas, comprendidos en el rango "a-z" y "0-9".
- Separar las palabras con guión medio "-".
- En caso de corresponder, ubicar la referencia temporal o del atributo de fragmentación siempre al final.

Ejemplos:

acceso-informacion-publica.csv: Versión completa del recurso.

acceso-informacion-publica-2013.csv: Versión del recurso fragmentada por año.

acceso-informacion-publica-201302.csv: Versión del recurso fragmentada por mes.

acceso-informacion-publica-caba.csv: Versión del recurso fragmentada por división político-territorial (provincia o caba).

acceso-informacion-publica-caba-2013.csv: Versión del recurso fragmentada por división político-territorial (provincia o caba) y año.

acceso-informacion-publica-jujuy-20130208.csv: Versión del recurso fragmentada por división político-territorial (provincia o caba) y fecha.

Para la fragmentación temporal, recomendamos el estándar de los ejemplos, ya que es compacto y ordena los recursos por tiempo: YYYYMMDD. Por favor, recordemos mantener siempre dos dígitos para el mes y el día, incluso si el número es menor a 10.

Codificación Todos los recursos de datos, incluyendo los geográficos, deben publicarse usando la codificación UTF-8 siguiendo las recomendaciones de la W3C en lugar de ASCII

https://cio-wiki.org/wiki/American_Standard_Code_for_Information_Interchange_(ASCII)

y

$$https://en.wikipedia.org/wiki/ASCII.$$

Una de las principales razones es que UTF-8

$$https://es.wikipedia.org/wiki/UTF-8$$

soporta una gran variedad de lenguajes, según la W3C es un "estándar en el que se definen todos los caracteres necesarios para la escritura de la mayoría de los idiomas hablados en la actualidad. Su objetivo es ser, y, en gran medida, ya lo ha logrado, un superconjunto de todos los sets de caracteres que se hayan codificado".

Estructura y características de los datos tabulares En esta sección veremos:

- Recomendaciones generales para el trabajo con datos, y
- Recomendaciones para el trabajo con planillas de cálculo, orientadas tanto a facilitar su exportación a formatos abiertos, como a su propia usabilidad en el contexto de cualquier aplicación de planillas de cálculo.

Recomendaciones generales Estas son recomendaciones generales para el trabajo con datos tabulares. Sugerimos adoptarlas sea cual sea la tecnología usada.

Muchas de las recomendaciones aquí presentadas se encuadran en los principios de Tidy Data delineados por Hadley Wickham. Éstos establecen, por ejemplo, que en una tabla de datos "cada variable es una columna, cada observación es una fila, y cada tipo de unidad observacional es una tabla". Sugerimos complementar la lectura de esta guía con la del trabajo que mencionamos.

Nomenclatura de los campos (nombres de las columnas) La "nomenclatura de los campos" es el nombre de las columnas en los datos de estructura tabular. Estas recomendaciones aplican a la generalidad de los casos, pero cuando haya convenciones particulares según la temática o rubro de datos de que se trate y éstas entren en conflicto, puede ser conveniente privilegiar primero la convención de la temática específica y luego la convención general.

Los nombres de los campos deben:

- Estar en español.
- Ser lo más explícitos, descriptivos y declarativos como sea posible.
  - Es preferible que el nombre de un campo sea claro antes que corto, pero se recomienda no superar los 50 caracteres.
  - No usar abreviaturas si no es estrictamente necesario o recomendado por una convención ampliamente difundida. En caso de usarlas, incluirlas en el diccionario de datos.
- Estar en minúsculas, no incluir caracteres especiales, ni estar subrayados.

- Usar palabras compuestas únicamente de caracteres en minúsculas comprendidos en el rango a-z (letras sin tildes) y en el rango 0-9 (dígitos).
- Las palabras deben unirse con guión bajo " _ ".
- No contener espacios.
- Las palabras deben separarse siempre con " _ ", en lugar de no tener separación alguna: fecha_audiencia_solicitada en lugar de fechaaudienciasolicitada

- Referirse a un sólo atributo de los datos, indivisible en más de un campo.

  - Los campos deben separar los atributos de los datos en la forma más desagregada que sea posible.
  - Se debe evitar definir campos que contengan más de un tipo de información (por ejemplo: e-mail y sitio web, número de teléfono, etc bajo "datos_de_contacto").

- Si existe una entidad que engloba varias características separadas en campos diferentes, comenzar nombrando los campos con esa entidad y luego con los atributos más específicos (de lo más general a lo más específico).

  - Ej.: solicitante y solicitante_documento son entidades más generales que se repiten en varios campos, que corresponden a atributos más específicos.

    solicitante_nombre
    solicitante_cargo
    solicitante_documento_tipo
    solicitante_documento_numero

- Resulta más fácil identificar qué campos están relacionados entre sí porque configuran atributos de una misma entidad, en lugar de parecer campos conceptualmente independientes. Además, el ordenamiento alfabético de los campos los dejaría automáticamente agrupados por su pertenencia a una entidad más importante.

- Incluso cuando la entidad de un atributo parezca evidente (ej.: un dataset llamado "audiencias" donde todos los campos son atributos de la entidad "audiencia"), se recomienda nombrar el campo incluyendo la entidad a la que hace referencia el atributo.

  No recomendado: "fecha_hora"
  Recomendado: "audiencia_id", "audiencia_fecha_hora".

Los campos que sean identificadores o códigos, deberán incluir el sufijo "_id" en el nombre del campo, salvo casos excepcionales donde un nombre alternativo sea más conveniente porque ofrece información sobre el sistema de identificación usado.

- En cuanto a los campos que contengan la descripción de ese identificador, se recomienda que incluya el sufijo "_desc" (por "descripción") en caso de que no exista una forma más conveniente de nombrar el campo.

Nivel de granularidad de los datos Por favor, no incluir totales, subtotales ni agrupamientos de datos. Un dataset debe ser consistente en el nivel de granularidad de los datos que contiene. Está bien tener un dataset con la cantidad de convenios firmados por provincia y está bien tener un dataset con la cantidad de convenios firmados por municipio. No está bien tener un dataset que mezcle ambos.

Dicho esto, el dato agregado "convenios firmados por provincia" siempre se puede calcular a partir de un proceso del dataset más desagregado, pero esto no es así a la inversa (es imposible recuperar los datos a nivel de municipio desde el dataset provincial).

Usar orientación vertical en lugar de horizontal Es preferible que la orientación de los datos sea "vertical" en lugar de "horizontal" en los casos en que esto sea posible. La principal razón es que los datos orientados de manera vertical facilitan el tratamiento y análisis de los datos.

Incluir sólo un atributo por campo Se recomienda definir los campos de forma atómica de modo de incluir un sólo atributo por elemento, en lugar de datos múltiples, generando campos adicionales de ser necesario.

Valores nulos, desconocidos o en blanco en campos numéricos Los valores de los datos deben ser siempre explícitos y respetando el tipo de datos del campo de que se trate. Los elementos o celdas en blanco se interpretarán siempre como "valor ausente". Si existen distintas interpretaciones posibles de un "valor ausente", éstas deben ser explicitadas en un campo aparte. Si sólo hay "valores ausentes" (no hay distintas interpretaciones, es siempre un "valor ausente") no es necesario agregar una columna adicional.

Es importante destacar, por ejemplo, que cuando un valor numérico sea "0" siempre debe ponerse un "0" como dato (y no un valor nulo, en blanco o vacío).

Recomendaciones para estructurar planillas de cálculo Las recomendaciones de esta sección aplican exclusivamente al trabajo sobre planillas de cálculo.

Usar celdas simples recomendamos usar celdas simples y, en ningún caso, combinar celdas.

Fila de encabezado Los datos deben contener sólo una fila de encabezado. Desde la segunda fila en adelante, sólo debe haber datos, pero nunca un encabezado.

Celdas vacías en filas para agrupar conceptos Recomendamos no dejar celdas vacías en filas bajo la presunción de que valores en blanco posteriores a un valor positivo contienen implícitamente a ese mismo valor en una suerte de "agrupamiento conceptual". Este error es muy común en la construcción de planillas de cálculo y suele generar problemas graves cuando cambia el orden original de las filas. Además, impide el uso de tablas dinámicas y otras formas de analizar los datos.

Formato de celdas Las celdas de una planilla de cálculo deben estar formateadas acorde al tipo de datos de que se trate. Específicamente, los números siempre deben estar en celdas de formato/tipo "número", los campos de tipo textual deben estar en celdas de formato/tipo "texto" y los campos de tipo fecha deben estar en celdas de formato/tipo "fecha".

Mantener el formato correcto de las celdas según el tipo de datos que contengan:

- Mejora las probabilidades de que una exportación a otro formato salga correctamente.
- Hace que los datos sean más operables en la propia planilla de cálculo, aprovechando mejor sus funcionalidades.

Exportación a CSV Insistimos: CSV es el formato más recomendado para la publicación de archivos tabulares. A la hora de exportar una planilla de cálculo a CSV hay 3 parámetros que deben ser especificados durante la exportación, independientemente del Software de que se use:

- Codificación (encoding, en inglés): siempre debe ser UTF-8.
- Carácter separador (separator character, en inglés): siempre debe ser "," (coma).
- Carácter calificador (quote character o enclosing character, en inglés): siempre debe ser " " " (comillas dobles).

10.2 Estándares Según el Tipo de Datos

El formato recomendado para los distintos tipos de datos está mayormente basado en las especificaciones de la W3C. En los otros casos, las recomendaciones surgen de la experiencia de trabajo del equipo de la Dirección Nacional de Datos e Información Pública y del esfuerzo realizado en la búsqueda de estándares más adecuados.

Texto

- Los campos de texto no deben contener espacios en blanco innecesarios al principio ni al final.

Entidades Las entidades que aparezcan entre los datos de un campo textual deben tener una descripción única. Es decir, toda mención que se realice a una entidad dada debe hacerse usando exactamente la misma cadena de caracteres cada vez:

- Las descripciones de entidades deberían elegirse siempre de forma tal que cumplan con el estándar específico que las describe, en caso de que este exista.

- Cuando este estándar no existe y hay dudas respecto del criterio a adoptar para elegir la descripción única de una entidad, debe privilegiarse siempre aquella que sea lo más explícita, descriptiva y declarativa posible.

Siempre que sea posible, la elección deberá fundamentarse en el estándar establecido para ese tipo de entidad (para más información ver la Guía para la identificación y uso de entidades interoperables). En el caso de no existir un estándar, deberá adecuarse a las pautas generales del contexto del dataset de que se trate.

Nombres propios Se capitalizan (la primera letra de cada palabra es mayúscula, el resto de las letras son minúsculas) todas las palabras significativas, salvo las siglas. Las palabras significativas son aquellas que no cumplen la función de artículos o preposiciones.

Siglas Todas las siglas se escriben en mayúsculas, sin usar puntos ni espacios intermedios.

Número

- El separador decimal debe ser el carácter ".".
- No se usará separador de miles.
- No se deberán usar espacios en blanco.
- Para los números negativos debe incluirse el símbolo menos "-" inmediatamente antes del número, sin espacio en blanco intermedio.

Moneda Los valores numéricos que sean además valores monetarios se consideran números y, por lo tanto, valen las mismas recomendaciones que para ellos. Además, agregamos las siguientes recomendaciones:

- La cantidad de decimales debe limitarse a dos, salvo que el uso de una mayor cantidad de decimales sea significativo para el caso particular.
- En ningún caso se incluirán símbolos o letras en el campo numérico -ya sea "$", "DOL", "USD", etc.

Si en el recurso los valores monetarios están expresados en diferentes monedas, se recomienda indicarlo en un campo aparte (que puede llamarse "moneda_id") usando los códigos alfabéticos definidos en la ISO 4127.

Números telefónicos En este apartado, proponemos una solución para incluir números telefónicos nacionales en los recursos de datos. A nivel internacional, el estándar para los números telefónicos fue desarrollado por el "Sector de Normalización de las Telecomunicaciones de la Unión Internacional de Telecomunicaciones" (ITU Telecommunication Standardization Sector) bajo la recomendación E.164.

Para el caso de los números nacionales, el ENACOM tiene la competencia sobre el sistema de numeración telefónica. Este organismo determina que el número nacional de abonado debe estar compuesto por 10 dígitos. Estos 10 dígitos están conformados por un indicativo interurbano más un número de abonado. Pudiendo el indicativo interurbano tener entre 2 y 4 dígitos, y el número de abonado entre 6 y 8 dígitos.

Coordenadas Para registrar datos de coordenadas geográficas de puntos, usamos números decimales. Los campos deberán llamarse "latitud" y "longitud". Cuando sea conveniente especificar el nombre de la entidad de la cual se consignan las coordenadas, se usarán los sufijos "_latitud" y "_longitud".

Tiempo Se usará el estándar ISO 8601

$$(YYYY - MM - DDTHH : MM : SS[.mmmmmm][+HH : MM]).$$

A menos que se indique lo contrario, se asumirá que la zona horaria.

- Fecha: YYYY-MM-DD
- Hora: HH:MM:SS[.mmmmmm][+HH:MM]
- Fecha y Hora: YYYY-MM-DDTHH:MM:SS[.mmmmmm][+HH:MM]
- Duración: YYYY-MM-DDTHH:MM:SS[.mmmmmm]

Rangos horarios

- Los rangos estarán divididos en dos partes separadas por un doble guión bajo "__", la primera indica el día y la segunda, la hora.
- Se puede omitir la parte del día o bien de la hora pero nunca ambas.
- Si se omite la parte que indica el día se asumirá que el rango abarca todo el horario indicado.
- Si se omite la parte que indica el horario se asumirá que el rango abarca todo el día indicado.
- El día se puede indicar tanto mediante rangos separando los días con guiones medios "-" o bien como particulares con el guión bajo "_".
- La hora se indica mediante rangos, separando los horarios con guiones medios ("-"). También se pueden indicar varios horarios con el guión bajo "_".
- En caso de que se necesite cubrir más de una franja horaria y esta sintaxis sea insuficiente, se pueden incluir varias separadas por espacios.
- Los días se indicarán con sus iniciales en castellano: LUN, MAR, MIE, JUE, VIE, SAB y DOM

Booleano A menos que se indique lo contrario, se identificarán con los valores true o false.

- Esta convención puede variar en algunos rubros específicos de datos, pero en caso de no existir una convención clara y definida aplicable al rubro o contexto del dataset, se recomienda utilizar true o false.
- Este campo puede contener "valores ausentes". En ese caso, el campo deberá estar totalmente vacío, no conteniendo ningún caracter.
- Si existe la posibilidad de que haya otro valor que no sea true, false o "valor ausente" significa que se eligió un tipo de datos incorrecto: este no es booleano, el tipo de dato booleano es binario y sólo admite 2 valores de verdad (aparte del caso del "valor ausente").

Datos de entidades interoperables ¿Qué son? Las entidades interoperables son aquellas que se repiten y usan frecuentemente dentro de datasets de:

- Temáticas diversas entre sí.
- Una misma temática (ej.: Salud), pero no de otras (como Educación, Economía, Transporte, etc).

La mayoría de los datasets incluyen campos que responden al dónde, quién, cuándo y qué. Estos campos permiten que los datasets sean interoperables entre sí.

10.3 ¿Qué son los Metadatos?

Los metadatos son los datos sobre los datos. Es decir: elementos descriptivos que dan contexto a un conjunto de datos, y acercan al usuario la información necesaria para entenderlos y usarlos eficazmente.

Un título y una breve descripción son los metadatos básicos que cualquier conjunto de datos a publicar debería tener. Después, existen muchos otros elementos que ayudan al lector a hacer un buen uso de los datos. Por ejemplo:

- Nombre, tipo de datos y descripción de los campos: ¿qué significa cada campo? ¿qué datos puedo encontrar en esa columna? ¿qué dicen y qué no dicen esos datos, cómo debo leerlos?
- Palabras clave: clasifican a un dataset como perteneciente a un conjunto de tópicos.
- Tema: clasifican a un dataset como perteneciente a un determinado tema, dentro de una jerarquía temática.
- Fecha de publicación: ¿cuándo se publicó por primera vez este dataset?
- Fecha de última modificación: ¿cuándo se actualizó por última vez este dataset?
- Frecuencia de actualización: ¿cada cuánto se actualiza este dataset?
- URL de descarga: ¿cómo dispongo de los datos, desde dónde puedo descargarlos?

Una lista curada de campos de metadatos, junto con las instrucciones de cómo deben utilizarse, define un perfil de metadatos.

¿Cómo se publican los metadatos? la publicación de los metadatos puede ser muy diversa en detalle, calidad y forma. Una publicación muy elemental es un documento de texto que ofrece una descripción del dataset y de cada uno de los recursos que lo componen.

Sin embargo, las computadoras no pueden leer fácilmente documentos de texto. La organización sistemática de colecciones de datasets (es decir, la creación de un catálogo de datos) exige un nivel de complejidad mayor para facilitar su descubrimiento, indexación, y reutilización por parte de scripts y aplicaciones de todo tipo.

10.4 Qué son los NFTs ¿Burbuja, Moda o Nuevo Paradigma?

A fines de los noventa y principios del nuevo siglo, muchos académicos consideraron que la Economía como ciencia estaba experimentando una experiencia similar a la de la Teología con Galileo o Darwin. Habiéndose basado en la idea de la escasez de un elemento primordial (oro, petróleo, silicio) ahora debería reinventarse para una era basada en un elemento superabundante; la información.

Con la aparición de la impresión 3D permitiendo la producción a pequeña escala de productos hasta entonces reservados a grandes plantas industriales y, la de las criptomonedas que desafiaba el poder de los Estados en la producción de divisas, esa opinión pareció confirmarse.

¿Qué son los NFTs? Los NFTs nacieron de la mano de la tecnología de la cadena de bloques[114] para garantizar que un artículo se mantenga único en un ambiente donde todo puede copiarse. Las letras representan las siglas en inglés de *Tokens No Fungibles*. Esto significa que no se pueden reemplazar ni intercambiar porque tienen propiedades únicas. De la misma forma, no se pueden falsificar ni manipular porque su autenticidad está respaldada por un certificado creado utilizando la tecnología de cadena de bloques.

[114] Es una colección de información almacenada en forma electrónica a la que se puede acceder, filtrar y manipular rápida y fácilmente cualquier número de usuarios a la vez.

Los NFT, como dijimos, son Tokens individuales con información valiosa almacenada en ellos. Esta información contienen los datos únicos de los NFTs son los que permiten la verificación y validación de su propiedad y la transferencia de Tokens entre propietarios. La creación y circulación de NFTs falsos no funciona porque cada artículo puede rastrearse hasta el creador o emisor original.

Dependiendo del mercado en que se comercialice un NFT, el creador original puede seguir obteniendo regalías de las ventas sucesivas.

Diferencias entre NFTs y criptomonedas Aunque la tecnología detrás de las criptomonedas y los NFTs es la mísma, hay una diferencia fundamental.

Las criptomonedas son fungibles, es decir que pueden intercambiarse sin que se note la diferencia. La firma digital que acompaña a cada Token No Fungile hace esto imposible.

Productos que pueden comercializarse como NFTs

- Arte digital.
- Gif animados.
- Vídeos de acontecimientos destacables.
- Avatares virtuales y otros complementos para videojuegos.
- Música.

¿Moda, burbuja o nuevo paradigma? En principio, los NFTs parecen una idea genial. los creadores pueden vender su contenido directamente al consumidor como un NFT, lo que también les permite quedarse con un mayor porcentaje de ganancia que la que dejan la mayoría de las plataforma. Como dijimos, se pueden programar regalías para recibir un porcentaje de las ventas cada vez que se produzca un cambio de manos.

Los precios también son bastante atractivos. Nyan Cat, un GIF de un gato con un cuerpo de tartaleta que apareció en el 2011, se vendió por casi $ 600,000 a principio de año. Por otra parte, videos con los momentos más destacados de la NBA generaron más de $ 500 millones en ventas a fines de marzo del 2021. Un solo NFT destacado de LeBron James se vendió por más de 200.000 dólares. Pero, los expertos aconsejan cautela. ya que se trata de una modalidad relativamente nueva.

Tokens En principio, la palabra Token puede usarse como sinónimo de «criptomoneda» o criptodivisa. Aunque, lo más correcto, teniendo en cuenta el tema, deberíamos hablar de «criptoactivos». Siendo más específicos se utiliza para describir ciertos activos digitales que se ejecutan sobre la cadena de bloques de otras criptomonedas, como lo hacen muchos Tokens de finanzas descentralizadas (o DeFi) o los propios NFTs. Gracias a los Tokens se hacen posible los intercambios descentralizados, por ejemplo la venta de complementos para videojuegos. Siempre manteniendo la posibilidad de pueden negociarse o atesorarse como cualquier criptodivisa.

Algunos tipos de Tokens

- Tokens DeFi: Existen protocolos basados en criptomonedas que tienen como objetivo reproducir las funciones tradicionales del sistema financiero; préstamos, ahorros, seguros, comercio internacional. Dichos protocolos emiten Tokens que realizan una amplia variedad de funciones, pero también se pueden comerciar o mantener como cualquier otra criptomoneda.

- Tokens de gobernanza: Es el equivalente digital a las acciones de una sociedad. Estos son Tokens DeFi especializados que les dan a los propietarios derecho a voto en cuestiones sobre el protocolo.

- Tokens no fungibles: Estos Tokens representan derechos de propiedad sobre un activo digital o del mundo real único. Se utilizan para restringir la copia y el intercambio de creaciones digitales. También sirven para emitir un número limitado de obras de arte digitales o vender activos virtuales únicos.

- Tokens de propiedad (Security Tokens): Son el equivalente criptográfico a las acciones y bonos. Se utilizan para vender participaciones en empresas o proyectos sin la necesidad de utilizar un agente de bolsa o cumplir las normas de los entes reguladores.

Cadena de Bloques (Blockchain) Para definirlo en forma simple, una cadena de bloques es un tipo de base de datos. Es decir que es una colección de información almacenada en forma electrónica a la que se puede acceder, filtrar y manipular rápida y fácilmente cualquier número de usuarios a la vez.

Sin embargo, existe una diferencia importante con las bases de datos tradicionales. La forma en que se estructuran los datos. En una cadena de bloques se recopila información en grupos (los bloques que le dan nombre), estos grupos contienen conjuntos de información. Los bloques tienen ciertas capacidades de almacenamiento y, cuando se llenan, se encadenan al bloque previamente llenado, formando una cadena de datos conocida como «Blockchain». Toda la información nueva que sigue a ese bloque recién agregado se ingresa en un bloque recién formado que luego también se agregará a la cadena una vez que se complete.

Este sistema también crea una línea de tiempo irreversible de datos cuando se implementa de manera descentralizada. Cuando se llena un bloque, se hace inmodificable y se convierte en parte de esta línea de tiempo. A cada bloque de la cadena se le asigna una marca de tiempo exacta cuando se agrega a la cadena.

En una cadena de bloques, cada nodo tiene un registro completo de los datos que se han almacenado en la cadena de bloques desde su inicio. Si un nodo tiene un error en sus datos, puede usar los miles de otros nodos como punto de referencia para corregirse. De esta manera, ningún nodo dentro de la red puede alterar la información contenida en ella.

Contratos Inteligentes Se trata de un programa que puede integrarse en la cadena de bloques para facilitar, verificar o negociar un acuerdo contractual. Los contratos inteligentes trabajan de acuerdo a un conjunto de reglas que los usuarios aceptan. Cuando se cumplen esas condiciones, los términos del acuerdo se llevan a cabo automáticamente.

Billetera Criptográfica (Crypto Wallet) Se trata de un Software que interactúa con la cadena de bloques permitiendo generar claves públicas y privadas, monitorear los saldos y enviar y recibir criptomonedas.

Los Tokens No Fungibles Debido al estrecho parentesco de los NFTs con las criptomonedas, el término correcto para referirse a su creación, es acuñar. El primer paso para la creación y venta es elegir una plataforma de cadena de bloques. La preferida por la mayoría de los creadores parece ser *Ethereum*.

La idea de los creadores del proyecto fue crear un ordenador mundial descentralizado imparable, resistente a la censura y autosuficiente. En él

corre un Software que garantiza que los datos y los pequeños programas informáticos llamados contratos inteligentes se replican y procesan en todos los ordenadores de la red, sin un coordinador central.

Habíamos dicho que los contratos inteligentes son programas que garantizan que cuando una de las partes cumple una condición la otra cumpla con lo comprometido. Es decir que a diferencia de las cadenas de bloques tradicionales, Ethereum no solo almacena datos, además realiza operaciones con esos datos y almacena los resultados.

El segundo criterio es contar con una billetera compatible. El uso de la red Ethereum requiere el pago de una tarifa, para abonarla necesitarás tener ETH, su moneda nativa, en la billetera. La puedes adquirir en cualquier sitio de intercambio.

La gran ventaja de usar Ethereum para el intercambio de los NFTs, es que es compatible con los estándares de creación de código abierto. Los NFTs se guardan en tu billetera. Esto hace que no tengas que comprometerte con ninguna plataforma de compraventa.

Plataformas de Creación de NFTs Dado que los NFTs están basados en estándares abiertos, es posible crearlos sin utilizar una plataforma. Sin embargo, iniciemos por el camino más fácil:

- OpenSea Cuenta con una función que permite crear NFT sin pagar tarifas en ETH. La plataforma publicó una guía paso a paso sobre cuáles son los pasos a seguir. Primero hay que crear una colección en la cual se almacenarán los NFTs. A partir de esto, se pueden crear utilizando archivos de imagen, archivos de video, modelos 3D, archivos de música o cualquier otro tipo de archivo de contenido digital. U paso adicional pero optativo es incluir un nombre, una descripción y establecer la rareza.

- Mintbase es otra plataforma ideal para principiantes. Utiliza un procedimiento muy parecido a OpenSea salvo que en lugar de crear una colección, lo que se crea es una tienda. Un punto en contra es que Mintbase solo admite imágenes por el momento, por lo que solo sirve para artistas visuales. Después de crear la tienda, llega el momento de acuñar los NFTs con su nombre, descripción y cantidad. La opción predeterminada es que una vez creado, un NFT sale a la venta, pero es posible retirarlos de manera momentánea.

- Foundation es para artistas con pretensiones de exclusividad. Cualquiera puede inscribirse, pero solo aquellos que reciben una invitación pueden crear sus NFTs. Si quieres saber cuales son los requisitos, no tienes más que visitar su página. Foundation permite la acuñación de NFT con imágenes, archivos de video, archivos de audio y modelos 3D. Se puede elegir el nombre, la descripción y la cantidad de ejemplares de cada uno.

Costo Energético El consumo ligado a los NFTs es preocupante... Es difícil estimar la huella de carbono ligada al proceso de 'acuñado' (Minting) y comercialización de un NFT porque hay muchos procesos implicados. De hecho lo que se ha estudiado sobre todo no es el impacto medioambiental de la creación de un NFT en particular, sino de una transacción en la red Ethereum en general.

Hay varios estudios que intentan estimar esa cantidad. Digiconomist sitúa la huella de carbono de una transacción con Ethereum en 46.54 kg de CO2, mientras que el artista y programador Memo Akten afirma que esa huella es era de 20 kg de CO2.

Atken iba más allá y explicaba cómo ese clic de ratón con el que compramos un NFT implica que se ponga en marcha esa compleja maquinaria del minado de la criptomoneda ETH en la red Ethereum. Ese minado, que actualmente se realiza porque Ethereum usa un algoritmo Proof-of-Work (PoW), hace que el consumo energético sea elevado, pero en NFT implica además otros consumos que llevan la huella de carbono hasta los 48 kg de CO2 según sus datos.

De hecho si tenemos en cuenta el resto de procesos implicados en su comercialización (acuñado, subastas, cancelación de subastas, venta, transferencia de propiedad), Atken afirma que la huella total va mucho más allá y roza los 200 kg de CO2.

Una sola transacción de Ethereum consume lo mismo que un hogar medio en EE.UU. durante tres días, y su huella de carbono equivale a casi 8.000 horas viendo YouTube. Fuente: Digiconomist.

El dato es desde luego preocupante, sobre todo cuando lo comparamos con acciones mucho más «inocuas» como el envío de un correo electrónico (unos pocos gramos dependiendo del contenido, 50 si tiene muchos adjuntos) o ver una hora Netflix (unos 22 gramos según este estudio). La huella de carbono de un NFT es por lo tanto miles de veces mayor que el de esos dos

escenarios que, eso sí, son muchísimo más comunes.

Eso es cierto: en realidad la mayor parte de las transacciones se deben al Trading, es decir, a las operaciones de compraventa de distintas criptodivisas que ahora están teniendo especial popularidad en esta red por el auge de las finanzas descentralizadas (DeFi) y de plataformas como Uniswap.

Aquí de hecho Marcos hace una comparación con el consumo de la red bancaria mundial o con el consumo energético que por ejemplo generan Netflix o YouTube: entre ambos, indicaban en Statista, son responsables de una cuarta parte de todo el tráfico de internet mundial. Ahí es nada.

Pero si hablamos de comparaciones, hay otras que también dejan otra perspectiva sobre el consumo de Ethereum y los NFTs. En Loop News hablaban precisamente de este tema y revelaban que según la WWF fabricar una sola camiseta de manga corta de algodón requiere 494 kWh, que es la cantidad equivalente de hervir 2.700 litros de agua.

Eso hace que la energía consumida por ese aparentemente sencillo proceso sea 10 veces superior a la energía que consume una transacción de Ethereum. Teniendo en cuenta que crear y vender un NFT equivale unas cuatro de esas transacciones (sin incluir las subastas), esto parece dejar claro que aunque los NFTs consumen mucho, cosas aparentemente mundanas como fabricar una camiseta (o ver Netflix) también consumen una barbaridad.

En un artículo publicado en Bit2Me News, se habló sobre el informe emitido por la Agencia Internacional de la Energía (IEA) donde revela que Bitcoin consume solo el 0.3% de toda la energía que se usa en el planeta al año. En términos más precisos, la red Blockchain ocupa cerca de 80 TWh anuales, mientras que Ethereum consume cerca de 25 TWh al año.

Como explica el informe, estas cifras pueden parecer grandes en comparación con países como Irlanda, que consume cerca de 26 TWh; o de tecnologías emergentes como los vehículos eléctricos, que consumen cerca de 58 TWh al año, según cifras de 2018.

Pero en comparación con el consumo de la refrigeración, por ejemplo, el consumo energético de Bitcoin es bastante pequeño. Los sistemas de refrigeración en el mundo consumían cerca de 2,020 TWh anuales para 2018. Y si comparamos a Bitcoin con el consumo de YouTube, una popular plataforma de video, también se queda pequeño. Para finales de 2019, YouTube consumía cerca del 2% de la energía global, unos 243.6 TWh al año. Ahora imagina cuánta energía consumen plataforma gigantescas como Google, Apple o Facebook para mantener sus redes operativas. Estas plataformas están buscando alternativas que les permite reducir su huella de carbono en el

planeta, pero las criptomonedas también lo están haciendo.

En las criptomonedas, los mineros de Bitcoin están diseñando nuevos equipos mineros que requieran menos consumo energético para funcionar, y están preocupados por crear nuevas fuentes de energía limpia con las que garantizar la operatividad de la red con el menor impacto ambiental posible. En Estados Unidos y Rusia ya se desarrollan granjas mineras con fuentes de energía limpia que aprovechan las emisiones de CO2 de las grandes plantas petroleras y las transforman en electricidad aprovechable.

En el caso de Ethereum, la red se prepara para migrar hacia un nuevo protocolo de consenso llamado prueba de participación o Proof of Stake (PoS) que elimina la necesidad de los mineros, y que por ende, consume menos energía eléctrica.

Mientras todo esto ocurre, el criptoartista Beeple vendió su colección de criptoarte "Everydays - The First 5000 Days" por 69 millones de dólares, y el CryptoPunk 7804 se vendió por 7.5 millones de dólares recientemente.

La Mayoría de NFTs no Valen (al menos, por ahora) El fenómeno de los NFTs parece conquistarlo todo. Conquista el mundo del cine, del fútbol, de los videojuegos o de internet. Su avance parece imparable, y todos los que apuestan por esta tecnología parecen tener claro que este es el futuro del arte y los activos y contenidos digitales.

Eso, claro, ha hecho que se tengan muchas expectativas sobre los NFTs para invertir y como forma de ganar potencialmente mucho dinero, pero lo cierto es que la mayoría de los NFTs no tienen apenas valor. No lo decimos nosotros, lo dice un reciente estudio científico.

Solo Unos Pocos Elegidos Ganan Dinero el célebre caso del artista digital Beeple es continuamente puesto como recordatorio de hacia dónde pueden ir los NFTs. Que vendiera su NFT por 69 millones de dólares en la casa de subastas Christie's fue un verdadero bombazo, pero esa fue más bien la excepción que confirma la regla.

Un estudio publicado por The Alan Turing Institute en el 2021 y realizado por investigadores de la Universidad de Londres trató de estudiar si estos activos tienen realmente el valor que muchos les asignan. Andrea Baronchelli, profesor asociado de matemáticas, y Matthieu Nadini, científico de datos, analizaron la venta de 4.7 millones de NFTs que se intercambiaron entre 500,000 compradores y vendedores y que representaban casi 1,000 millones

de dólares en transacciones.

¿Su conclusión? La mayoría de NFTs no valen nada. Solo un 1% de todos los analizados se vendieron por más de 1,500 dólares, y el 75% se vendió por 15 dólares o menos. De hecho, explicaban "la mayoría de estos activos ni siquiera logran ser vendidos, así que ni siquiera llegaron a entrar en nuestros análisis".

Eso significa que la amplísima mayoría de NFTs ni siquiera entraban a formar parte de esos porcentajes, que serían mucho más bajos si se tuvieran en cuenta todos esos NFTs que están esperando a que alguien los compre.

Para Mauro Martino, responsable del Visual AI Lab en IBM, "la gente se gasta dinero en crear el NFT y ahí queda todo. Sería difícil sugerirle a un amigo artista que se metiera en este segmento para hacerse rico, porque muy poca gente puede lograr beneficios en este mercado".

Martino lleva también un año "mapeando la revolución NFT" en un proyecto que dio como resultado un vídeo y un artículo en la prestigiosa Nature. En su opinión, no es probable que los NFTs puedan hacerle a uno rico, pero al estudiar esta tecnología se dio cuenta de que los NFTs están aquí para quedarse.

10.5 La Web 3

En los últimos días del 2021, el término «Web3» se ha puesto de moda. Algunos creen que puede cambiar Internet para siempre, mientras que otros consideran que no es más que otra vuelta de tuerca en ese control de la Red que se lleva desarrollando desde hace años, pero desde una perspectiva diferente.

Entre los segundos, Jack Dorsey. El ex-CEO de Twitter ha estado en el centro de la polémica cuando el pasado 22 de diciembre utilizó su red social preferida para decir que «la Web3 no es un fuerza para democratizar la web, sino una herramienta para los fondos de capital riesgo». La declaración no sentó muy bien en determinados círculos y como el ambiente ya llevaba unos días enrarecido, Marc Andreessen, uno de los principales VC con intereses en lo que la Web3 representa, acabó bloqueando a Dorsey.

Pero para entender la razón del enfado de algunos fondos de inversión con Dorsey, o por qué el creador de Twitter se ha mostrado tan abiertamente contrario a esta idea, es necesario dar un paso atrás y explicar que es Web3, sobre qué principios se asienta y si finalmente, estos van a importar en algún momento.

De la Web 1.0 a la Web3 En el desarrollo de internet, los usuarios nos hemos relacionado con la Web 1.0 (desde sus inicios hasta probablemente 2004), la Web 2.0 (desde 2004 al momento actual) y algunos consideran que tímidamente comenzamos a hacerlos con la Web3.

La Web 1.0 se basaba en una idea auténticamente democratizadora de la Red y el uso de estándares y tecnologías abiertas. Además de las páginas HTML, es la era de los foros On-line, los grupos de noticias en Usenet, las charlas a través de clientes IRC, o el uso del FTP para la descarga de archivos. También en muchos casos representaba una conversación unidireccional, de uno hacia muchos.

El desarrollo de la Web 2.0 supone una democratización en esa charla, ya que permite conversaciones masivas de muchos a muchos, gracias a entre otras cosas, la popularización de las redes sociales. Pero también supone el desembarco de las grandes corporaciones en Internet, con sus cosas buenas como el impulso del comercio electrónico o el ayudarnos a conectar con otros, y las no tan buenas, como la creación de plataformas propietarias, el control de la conversación y la compra-venta indiscriminada de los datos personales de los usuarios con fines publicitarios y de control.

En el apogeo de esta Web 2.0 en la que todavía nos encontramos, comienza a fraguarse el concepto de Web3: una posible Internet del futuro en la que todos los datos y contenidos se registran en Blockchains, se Tokenizan y se accede a ellos en redes distribuidas P2P con el fin de democratizar Internet, poner el poder en manos de los creadores de contenidos y arrebatar el control a gobiernos y empresas.

Criptomonedas, NFT y Blockchain En esta definición de Web3 encontramos algunas de las ideas que ya estaban presentes en la Web 1.0, como la democratización y la descentralización de la información, pero que de alguna forma, se perdieron con el desarrollo de una por otro lado muy bien intencionada Web 2.0 en sus orígenes.

Por supuesto la idea de Web3 puede sonar bien...para los entusiastas de las criptomonedas, los que han empezado a hacer negocios vendiendo sus propios NFTs y sí, los fondos de capital riesgo que consideran que en ambos casos hay una gran oportunidad de negocio que se puede aprovechar.

Y sin embargo, no está del todo claro de qué forma se va a pasar de las palabras a los hechos o incluso si en algún momento esto va a pasar. El propio Elon Musk, que de criptomonedas sabe algo, ha afirmado recientemente que,

como el Metaverso, Web3 es solo «Marketing-hype» y él no lo entiende.

Llama la atención cómo dos de los términos que más se han repetido en los últimos meses (Metaverso y Web3) describen plataformas que no existen en realidad y de las que no se espera nada realmente interesante hasta como mínimo dentro de una década o más. Es probable por supuesto, que el negocio de las divisas virtuales, de las cadenas de bloques y los NFTs siga creciendo. Pero que la relación de estos tres conceptos de una forma aún por determinar acabe dando lugar a una nueva forma de concebir internet, es algo que se antoja muy lejano. En todo caso, la Web3 se quiere parecer más a ese «Next Big Thing» que buscan las compañías tecnológicas que a una revolución democratizadora que juegue a favor de los intereses de los usuarios. Y ahí Jack Dorsey seguramente esté en lo cierto.

Del Hype a la Realidad Por mucho que ese empeñen los defensores de Web3, lo cierto es que como aseguran en ComputerWord, a corto y medio plazo resultará básicamente imposible conseguir que usuarios, empresas y organizaciones se pongan de acuerdo en el desarrollo de una nueva internet basada en la cadena de bloques. Entre otras cosas porque para el usuario medio, que no crea contenidos monetizables, esto no tiene ningún valor.

Y no es que no se haya intentado. Frente a plataformas como Facebook o Twitter, empresas como iniciativas como Mastodon han propuesto a los usuarios el apostar por redes sociales auténticamente descentralizadas y no controladas por ninguna empresa. ¿El resultado? Los usuarios han seguido utilizando Facebook, Twitter, TikTok y YouTube.

Por otro lado, no parece muy probable que compañías como Meta, Google, Amazon o incluso Apple, que se han hecho mil millonarias gracias al desarrollo de la Web 2.0 estén muy por la labor de apostar por una red completamente descentralizada en la que son los usuarios los que toman el control y las grandes corporaciones el que lo pierden.

Incluso si esto fuera así, en un mundo ideal Web3 supone una participación activa de cada uno de los usuarios de esa futura Internet, como inversores de su propia «empresa personal». Como esto no es posible, son las Startups que previsiblemente actuarían como intermediarias en esta internet descentralizada (albergando por ejemplo esos contenidos Tokenizados), las que más tendrían que ganar y que de hecho aspiran, por la propia lógica capitalista de los negocios, a convertirse a grandes empresas que controlen esa nueva «internet democrática». Porque seamos serios: los inversores en

capital riesgo invierten para ganar dinero, no para hacer felices a los internautas.

Ni siquiera en estos momentos, el mundo de las criptomonedas, uno de esos pilares sobre los que se quiere fundar la nueva Web, es democrático o igualitario. Basta recordar en este sentido que un nuevo estudio publicado por Baystreet ponía el foco de atención en lo que cada vez resulta más evidente: el grueso de los Bitcoins en circulación está en cada vez menos manos (se calcula que el 0,01% de los titulares de Bitcoin ya controlan el 27% de las monedas en circulación).

En definitiva, una cosa es tener servicios basados en Blockchain y Tokenizados que funcionen sobre la Web que ya conocemos y otra muy distinta pensar que van a sustituir la infraestructura existente. Lo primero es desde luego muy real y va a ir a más; lo segundo, altamente improbable.

10.6 Protegiendo Nuestros Metadatos

¿Qué son los metadatos? los metadatos son "datos sobre datos" o "información sobre información" que se incluyen en archivos informáticos, normalmente de forma automática. Los metadatos se utilizan para describir, identificar, categorizar y ordenar archivos. Sin embargo, los metadatos también se pueden utilizar para desanonimizar a los usuarios y exponer información privada.

Ejemplos de metadatos incluyen:

- En archivos de imagen y vídeo:

  El lugar donde se tomó la foto o vídeo.

  La fecha y hora en que se tomó la foto o vídeo.

  El modelo y número de serie de la cámara utilizada.

- En archivos de documentos de texto:

  El autor del documento.

  Cambios en el documento.

Algunos tipos de archivos que guardan metadatos que pueden poner en peligro el anonimato de los usuarios:

- Audio Interchange File Format (.aiff)
- Audio Video Interleave (.avi)
- Electronic Publication (.epub)
- Free Lossless Audio Codec (.flac)
- Graphics Interchange Format (.gif)
- High Efficiency Image Format (.heic, .heif)
- Hypertext Markup Language (.html, .xhtml)
- Portable Network Graphics (PNG)
- JPEG (.jpeg, .jpg, ...)
- MPEG Audio (.mp3, .mp2, .mp1, .mpa)
- MPEG-4 (.mp4)
- Office Openxml (.docx, .pptx, .xlsx, ...)
- Ogg Vorbis (.ogg)
- Open Document (.odt, .odx, .ods, ...)
- Portable Document Fileformat (.pdf)
- Portable Pixmap Format (.ppm)
- Scalable Vector Graphics (.svg)
- Tape ARchive (.tar, .tar.bz2, .tar.gz, .tar.zx)
- Torrent (.torrent)
- Waveform Audio (.wav)
- Windows Media Video (.wmv)
- ZIP (.zip)

Es imposible encontrar y eliminar de manera confiable todos los metadatos en formatos de archivos complejos. Por ejemplo, los documentos de Microsoft Office pueden contener imágenes incrustadas, audio y otros archivos que contienen sus propios metadatos que no se pueden eliminar. Por ello se debe eliminar los metadatos de cualquier archivo antes de incrustarlo en otro documento.

Además, siempre que sea posible, debes guardar los archivos en formatos más simples. Por ejemplo, en lugar de guardar un documento de texto como un archivo .docx, puede guardarlo como un archivo .txt simple.

Metadata Anonymisation toolkit v2 permite eliminar metadatos de archivos antes de publicarlos o compartirlos, funciona en muchos formatos de archivos, incluidos:

- Archivos de imagen, como .jpeg, .png y .gif
- Archivos de LibreOffice, como .odt y .ods
- Documentos de Microsoft Office, como .docx, .xlsx y .pptx
- Archivos de audio, como .mp3, .flac y .ogg
- Archivos de vídeo, como .mp4 y .avi
- Archivar archivos, como .zip y .tar

En GNU/Linux podemos instalar el paquete mat2, mediante:

```
# apt install mat2
```

Para conocer la lista de archivos soportados usamos:

```
$ mat2 -l
```

Para visualizar los metadatos del archivo usamos:

```
$ mat2 -s nombreArchivo
```

Para remover los metadatos del archivo usamos:

```
$ mat2 -V nombreArchivo
```

en la hipótesis de eliminación de metadatos, debes tener en cuenta que mat2 no elimina el archivo en el que interviene porque deja el archivo original como está, sino que crea otro archivo.

Exchangeable Image File al tomar fotografías o vídeos es recomendable por seguridad desactivar el guardado de datos Exif (Exchangeable Image File) también conocidos como metadatos, ya que estos contienen información sobre la cámara, sobre la fotografía y sobre su origen como ubicación por GPS, la hora de creación, la última modificación, etc.

En GNU/Linux podemos instalar el paquete ExifTool que permite conocer y borrar los datos Exif en fotografías. Para instalarlo usamos:

```
# apt install libimage-exiftool-perl
```

Para visualizar los datos Exif, usamos:

```
$ exiftool imagen.gif
```

Para borrar todos los datos Exif, usamos:

```
$ exiftool -all=imagen.gif
```

11 POO Java vs Python

Muchos programadores de Java en últimas fechas se trasladan a Python, a menudo luchan con el enfoque de Python para la programación orientada a objetos (POO). El enfoque para trabajar con objetos, tipos de variables y otras capacidades de lenguaje tomadas por Python vs Java es bastante diferente. Puede hacer que cambiar entre ambos lenguajes sea muy confuso.

Este apartado se compara y contrasta el soporte de programación orientada a objetos en Python vs Java. Al final, podremos aplicar nuestros conocimientos de programación orientada a objetos a Python al comprender cómo reinterpretar nuestra comprensión de los objetos de Java en Python y utilizar los objetos de una manera Pythonica.

A lo largo de este apartado, podremos:

- Construir una clase básica tanto en Java como en Python
- Explorar cómo funcionan los atributos de objeto en Python vs Java
- Comparar y contrastar los métodos de Java y las funciones de Python
- Descubrir los mecanismos de herencia y polimorfismo en ambos lenguajes
- Investigar la reflexión en Python vs Java
- Aplicar todo en una implementación de clase completa en ambos lenguajes

Este apartado no es una introducción a la programación orientada a objetos. Más bien, compara características y principios orientados a objetos de Python vs Java. Es recomendable que el lector tenga conocimiento de Java y también este familiarizado con la codificación de Python.

11.1 Ejemplo de Clases Java vs Python

Para comenzar, implementaremos la misma clase pequeña tanto en Java como en Python para ilustrar las diferencias entre ellos (se harán modificaciones a ellas a medida que avancemos).

Primero, supongamos que se tiene la siguiente definición de clase Carro en Java:

```
public class Carro {
   private String color;
   private String modelo;
   private int anio;
   public Car(String color, String modelo, int anio) {
      this.color = color;
      this.modelo = modelo;
      this.anio = anio;
   }
   public String getColor() {
      return color;
   }
   public String getmodelo() {
      return modelo;
   }
   public int getAnio() {
      return anio;
   }
}
```

Las clases de Java se definen en archivos con el mismo nombre que la clase. Entonces, se debe guardar esta clase en un archivo llamado *Carro.java*. Solo se puede definir una clase en cada archivo.

Una clase de Carro pequeña similar está escrita en Python de la siguiente manera:

```
class Carro:
   def __init__(self, color, modelo, anio):
      self.color = color
      self.modelo = modelo
      self.anio = anio
```

En Python se puede declarar una clase en cualquier lugar, en cualquier archivo, en cualquier momento. Por ejemplo, guardamos esta clase en el archivo carro.py.

Usando estas clases como base, podremos explorar los componentes básicos de clases y objetos.

11.2 Atributos de Objetos

Todos los lenguajes orientados a objetos tienen alguna forma de almacenar datos sobre el objeto. En Java y Python, los datos se almacenan en atributos, que son variables asociadas con objetos específicos.

Una de las diferencias más significativas entre Python y Java es cómo se definen y administran los atributos de clase y objeto. Algunas de estas diferencias provienen de las limitaciones impuestas por los lenguajes, mientras que otras provienen de las mejores prácticas de programación.

Declaración e Inicialización En Java, declaramos atributos en el cuerpo de la clase, fuera de cualquier método, con un tipo definido. Se debe definir los atributos de la clase antes de que se utilicen:

```
public class Carro {
   private String color;
   private String modelo;
   private int anio;
...
```

En Python, ambos se declaran y definen atributos dentro de la clase .__init__(), que es el equivalente al constructor de Java:

```
def __init__(self, color, modelo, anio):
   self.color = color
   self.modelo = modelo
   self.anio = anio
```

Al prefijar los nombres de las variables con *self*, le dice a Python que estos son atributos. Cada instancia de la clase tendrá una copia. Todas las variables en Python están escritas de forma flexible, y estos atributos no son una excepción.

También se puede crear variables de instancia fuera de .__init__(), pero no es una práctica recomendada ya que su alcance suele ser confuso. Si no se utilizan correctamente, las variables de instancia creadas fuera de .__init__() pueden provocar errores sutiles que son difíciles de encontrar. Por ejemplo, puede agregar un nuevo atributo .ruedas a un objeto *carro* como este:

```
1  >>> import carro
2  >>> mi_carro = carro.Carro("amarillo", "beetle", 1967)
3  >>> print(f"mi carro es {mi_carro.color}")
4  mi carro es amarillo
5
6  >>> mi_carro.ruedas = 5
7  >>> print(f"ruedas: {mi_carro.ruedas}")
8  ruedas: 5
```

Sin embargo, si olvidamos mi_carro.ruedas = 5 en la línea 6, Python muestra un error:

```
1  >>> import Carro
2  >>> mi_carro = Carro.Carro("amarillo", "beetle", 1967)
3  >>> print(f"Mi carro es {mi_carro.color}")
4  Mi carro es amarillo
5
6  >>> print(f"ruedas: {mi_carro.ruedas}")
7  Traceback (most recent call last):
8  File "<stdin>", line 1, in <module>
9  AttributeError: 'Carro' object has no attribute 'ruedas'
```

En Python, cuando declaras una variable fuera de un método, se trata como una variable de clase. Si actualizamos la clase Carro de la siguiente manera:

```
class Carro:
  ruedas = 0
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
```

Esto cambia la forma en que usa la variable .ruedas. En lugar de referirse a él usando un objeto, se refiere a él usando el nombre de la clase:

```
1  >>> import carro
2  >>> mi_carro = carro.Carro("amarillo", "beetle", 1967)
3  >>> print(f"Mi carro es {mi_carro.color}")
4  Mi carro es amarillo
```

```
5
6  >>> print(f"Este tiene {carro.Carro.ruedas} ruedas")
7  Este tiene 0 ruedas
8
9  >>> print(f"Este tiene {mi_carro.ruedas} ruedas")
10  Este tiene 0 ruedas
```

Nota: En Python, se hace referencia a una variable de clase con la siguiente sintaxis:

El nombre del archivo que contiene la clase, sin la extensión .py
Un punto
el nombre de la clase
Un punto
El nombre de la variable

Como guardó la clase Carro en el archivo carro.py, se refiere a la variable de clase ruedas en la línea 6 como carro.Carro.ruedas.

Puedo consultar *mi_carro.ruedas* o *carro.Carro.ruedas*, pero hay que tener cuidado. Cambiar el valor de la variable de instancia *mi_carro.ruedas* no cambiará el valor de la variable de clase *carro.Carro.ruedas*:

```
1  >>> from carro import *
2  >>> mi_carro = carro.Carro("amarillo", "Beetle", "1966")
3  >>> mi_otro_carro = carro.Carro("rojo", "corvette",
"1999")
4
5  >>> print(f"Mi carro es {mi_carro.color}")
6  Mi carro es amarillo
7  >>> print(f"Este tiene {mi_carro.ruedas} ruedas")
8  Este tiene 0 ruedas
9
10  >>> print(f"Mi otro carro es {mi_otro_carro.color}")
11  Mi otro carro es rojo
12  >>> print(f"Este tiene {mi_otro_carro.ruedas} ruedas")
13  Este tiene 0 ruedas
```

```
14
15  >>> # cambiar el valor de la variable de clase
16  ... carro.Carro.ruedas = 4
17
18  >>> print(f"Este tiene {mi_carro.ruedas} ruedas")
19  Este tiene 4 ruedas
20  >>> print(f"Mi otro carro tiene {mi_otro_carro.ruedas}
ruedas")
21  Mi otro carro tiene 4 ruedas
22
23  >>> # Cambiar el valor de la variable de instancia para
mi_carro
24  ... mi_carro.ruedas = 5
25
26  >>> print(f"Mi carro tiene {mi_carro.ruedas} ruedas")
27  Mi carro tiene 5 ruedas
28  >>> print(f"Mi otro carro tiene {m_otro_carro.ruedas}
ruedas")
29  Mi otro carro tiene 4 ruedas
```

Se han definido dos objetos Coche en las líneas 2 y 3:

1. mi_carro
2. mi_otro_carro

Al principio, ambos tienen cero ruedas. Cuando establece la variable de clase usando *carro.Carro.ruedas = 4* en la línea 16, ambos objetos ahora tienen cuatro ruedas. Sin embargo, cuando configura la variable de instancia usando *mi_ carro.ruedas = 5* en la línea 24, solo ese objeto se ve afectado.

Esto significa que ahora hay dos copias diferentes del atributo de *ruedas*:

1. Una variable de clase que se aplica a todos los objetos *Carro*
2. Una variable de instancia específica aplicable solo al objeto *mi_ carro*

No es difícil referirse accidentalmente al incorrecto e introducir errores sutiles.

El equivalente de Java a un atributo de clase es un atributo estático:

```
public class Carro {
  private String color;
  private String modelo;
  private int anio;
  private static int ruedas;
  public Car(String color, String modelo, int anio) {
    this.color = color;
    this.modelo = modelo;
    this.anio = anio;
  }
  public static int getRuedas() {
    return ruedas;
  }
  public static void setRuedas(int count) {
    ruedas = count;
  }
}
```

Normalmente, se refiere a variables estáticas utilizando el nombre de clase de Java. Puede hacer referencia a una variable estática a través de una instancia de clase como Python, pero no es una práctica recomendada.

La clase de Java se está alargando. Una de las razones por las que Java es más detallado que Python es la noción de métodos y atributos públicos y privados.

Pública y Privada Java controla el acceso a métodos y atributos diferenciando entre datos públicos y datos privados.

En Java, se espera que los atributos se declaren como privados o protegidos si las subclases necesitan acceso directo a ellos. Esto limita el acceso a estos atributos desde el código fuera de la clase. Para proporcionar acceso a atributos privados, se declaran métodos públicos que establecen (*setters*) y recuperan (*getters)* datos de manera controlada.

Recordando de la clase Java anterior que la variable de color se declaró como privada. Por lo tanto, este código Java mostrará un error de compilación en la última línea:

```
Carro miCarro = new Carro("azul", "Ford", 1972);
// Pintando el carro
```

```
miCarro.color = "Rojo";
```

Si no especifica un nivel de acceso, el atributo predeterminado es paquete protegido, lo que limita el acceso a las clases en el mismo paquete. Debe marcar el atributo como público si desea que este código funcione.

Sin embargo, declarar atributos públicos no se considera una práctica recomendada en Java. Se espera que declare los atributos como privados y utilice métodos de acceso público, como *.getColor()* y *.getModelo()* que se muestran en el código.

Python no tiene la misma noción de datos privados o protegidos que tiene Java. Todo en Python es público. Este código funciona bien con su clase Python existente:

```
>>> mi_carro = Carro.Carro("azul", "Ford", 1972)
>>> # Pintando el carro
... mi_carro.color = "rojo"
```

En lugar de privada, Python tiene la noción de una variable de instancia no pública. Cualquier variable que comience con un carácter de subrayado se define como no pública. Esta convención de nomenclatura dificulta el acceso a una variable, pero es solo una convención de nomenclatura y aún puede acceder a la variable directamente.

Si Agregamos la siguiente línea a la clase Python Carro:

```
class Carro:
  ruedas = 0
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self._portavasos = 6
```

Puede acceder a la variable ._portavasos directamente:

```
1  >>> import carro
2  >>> mi_carro = carro.Carro("amarillo", "Beetle", "1969")
3  >>> print(f"Fue construido en {mi_carro.anio}")
4  Fue construido en 1969
5  >>> mi_carro.anio = 1966
```

```
>>> print(f"Fue construido en {mi_carro.anio}")
Fue construido en 1966
>>> print(f"Tiene {mi_carro._portavasos} portavasos.")
Tiene 6 portavasos.
```

Python reconoce además el uso de caracteres de subrayado doble delante de una variable para ocultar un atributo en Python. Cuando Python ve una variable de subrayado doble, cambia el nombre de la variable internamente para dificultar el acceso directo. Este mecanismo evita accidentes pero aún así no imposibilita el acceso a los datos.

Para mostrar este mecanismo en acción, volvemos a cambiar la clase Python Carro:

```
class Carro:
  ruedas = 0
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self.__portavasos = 6
```

Ahora, cuando se intenta acceder a la variable .__portavasos se ve el siguiente error:

```
>>> import Carro
>>> mi_carro = Carro.Carro("amarillo", "Beetle", "1969")
>>> print(f"Fue contruido en {mi_carro.anio}")
Fue construido en 1969
>>> mi_carro.anio = 1966
>>> print(f"Fue construido en {my_car.anio}")
Fue construido en 1966
>>> print(f"Tiene {mi_carro.__portavasos} portavasos.")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Carro' object has no attribute '__portava-
sos'
```

Entonces, ¿por qué no existe el atributo de .__portavasos?

Cuando Python ve un atributo con guiones bajos dobles, cambia el atributo anteponiendo el nombre original del atributo con un guión bajo, seguido del nombre de la clase. Para usar el atributo directamente, también debe cambiar el nombre que usa:

```
>>> print(f"Tiene {mi_carro._Carro__portavasos} portavasos")
Tiene 6 portavasos
```

Cuando usa guiones bajos dobles para ocultar un atributo al usuario, Python cambia el nombre de una manera bien documentada. Esto significa que un desarrollador determinado todavía puede acceder al atributo directamente.

Entonces, si sus atributos de Java se declaran privados y sus atributos de Python están precedidos por guiones bajos dobles, ¿cómo proporciona y controla el acceso a los datos que almacenan?.

Control de acceso En Java, se accede a atributos privados usando *getters* y se establecen usando *setters* de manera controlada. Para permitir que los usuarios pinten sus autos, agregando el siguiente código a la clase de Java:

```
public String getColor() {
  return color;
}
public void setColor(String color) {
  this.color = color;
}
```

Dado que *.getColor* () y *.setColor* () son públicos, cualquiera puede llamarlos para cambiar o recuperar el color del automóvil. Las mejores prácticas de Java de usar atributos privados a los que se accede con captadores *get* y definidores *set* públicos es una de las razones por las que el código Java tiende a ser más detallado que Python.

Como se vio anteriormente, acceder a los atributos directamente en Python. Dado que todo es público, se puede acceder a cualquier cosa en cualquier momento y desde cualquier lugar. Establece y obtiene valores de atributo directamente haciendo referencia a sus nombres. Incluso puede eliminar atributos en Python, lo que no es posible en Java:

```
>>> mi_carro = Carro("amarillo", "beetle", 1969)
>>> print(f"Mi carro fue construido en {mi_carro.anio}")
Mi carro fue construido en 1969
>>> mi_carro.anio = 1966
>>> print(f"Mi carro fue construido en {mi_carro.anio}")
Mi carro fue construido en 1966
>>> del mi_carro.anio
>>> print(f"Mi carro fue construido en {mi_carro.anio}")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Carro' object has no attribute 'anio'
```

Sin embargo, es posible que en ocasiones desee controlar el acceso a un atributo. En ese caso, puede usar las propiedades de Python.

En Python, las propiedades proporcionan acceso controlable a los atributos de la clase mediante la sintaxis del decorador de Python. Las propiedades permiten que se declaren funciones en clases de Python que son análogas a los métodos getter y setter de Java, con la ventaja adicional de permitirle eliminar atributos también.

Podemos ver cómo funcionan las propiedades agregando una a la clase de Carro:

```
class Carro:
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self._voltaje = 12
  @property
    def voltaje(self):
    return self._voltaje
  @voltaje.setter
  def voltaje(self, volts):
    print("Aviso: esto puede causar problemas!")
    self._voltaje = volts
  @voltaje.deleter
  def voltaje(self):
    print("Aviso: La radio dejara de funcionar!")
    del self._voltaje
```

Aquí, expandimos la noción de *Carro* para incluir vehículos eléctricos. Al Declarar el atributo *._voltaje* para mantener el voltaje de la batería.

Para proporcionar acceso controlado, definimos una función llamada *voltaje()* para devolver el valor privado. Al usar la decoración *@property*, lo marca como un captador al que cualquiera puede acceder directamente.

De manera similar, se define una función de establecimiento, también llamada *voltaje()*. Sin embargo, decoramos esta función con *@voltaje.setter*. Por último, usamos *@voltaje.deleter* para decorar un tercer *voltaje()*, lo que permite la eliminación controlada del atributo.

Los nombres de las funciones decoradas son todos iguales, lo que indica que controlan el acceso al mismo atributo. Los nombres de las funciones también se convierten en el nombre del atributo que se usa para acceder al valor. Así es como funcionan estas propiedades en la práctica:

```
1  >>> from carro import *
2  >>> mi_carro = Carro("amarillo", "beetle", 1969)
3
4  >>> print(f"Mi carro usa {mi_carro.voltaje} volts")
5  Mi carro usa 12 volts
6
7  >>> mi_carro.voltaje = 6
8  Aviso: esto puede causar problemas!
9
10  >>> print(f"Mi carro ahora usa {mi_carro.voltaje} volts")
11  Mi carro ahora usa 6 volts
12
13  >>> del mi_carro.voltaje
14  Aviso: la radio dejara de trabajar!
```

Tengamos en cuenta que usamos *.voltaje*, no *._voltaje*. Esto le dice a Python que use las funciones de propiedad que se definió:

- Cuando imprime el valor de mi_carro.voltaje en la línea 4, Python llama a *.voltaje()* decorado con *@property*.
- Cuando asigna un valor a mi_carro.voltaje en la línea 7, Python llama a *.voltaje()* decorado con *@voltaje.setter*.

- Cuando elimina *mi_ carro.voltaje* en la línea 13, Python llama a *.voltaje()* decorado con *@voltaje.deleter*.

Las decoraciones *@property*, *@.setter* y *@.delete*r permiten controlar el acceso a los atributos sin requerir que los usuarios utilicen métodos diferentes. Incluso puede hacer que los atributos parezcan propiedades de solo lectura omitiendo las funciones decoradas *@.setter* y *@.deleter*.

self and this En Java, una clase se refiere a sí misma con la referencia *this*:

```
public void setColor(String color) {
  this.color = color;
}
```

this está implícito en el código Java: normalmente no es necesario escribirlo, a menos que pueda haber confusión entre dos variables con el mismo nombre.

Podemos escribir el mismo código de esta manera:

```
public void setColor(String newColor) {
  color = newColor;
}
```

Dado que *Carro* tiene un atributo llamado *.color*, y no hay otra variable en el alcance con el mismo nombre, una referencia a ese nombre funciona. Usamos esto en el primer ejemplo para diferenciar entre el atributo y el parámetro, ambos con nombre color.

En Python, la palabra clave *self* tiene un propósito similar. Así es como se refiere a las variables de miembro, pero a diferencia de *this* de Java, es necesario si desea crear o hacer referencia a un atributo de miembro:

```
class Carro:
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self._voltaje = 12
  @property
  def voltaje(self):
    return self._voltaje
```

Python requiere cada uno de ellos en el código anterior. Cada uno crea o hace referencia a los atributos. Si los omite, Python creará una variable local en lugar de un atributo.

La diferencia entre cómo se usa *self* y *this* en Python y Java se debe a diferencias subyacentes entre los dos lenguajes y cómo nombran variables y atributos.

11.3 Métodos y Funciones

Esta diferencia entre Python y Java es, en pocas palabras, que Python tiene funciones y métodos, mientras que Java sólo métodos.

En Python, el siguiente código está perfectamente bien (y es muy común):

```
>>> def di_hola():
... print("Hola!")
...
>>> di_hola()
Hola!
```

Se puede llamar a *di_hola()* desde cualquier lugar que sea visible. Esta función no tiene referencia a *self*, lo que indica que es una función global, no una función de clase. No puede alterar ni almacenar ningún dato en ninguna clase, pero puede usar variables locales y globales.

Por el contrario, cada línea de código Java que escribe pertenece a una clase. Las funciones no pueden existir fuera de una clase y, por definición, todas las funciones de Java son métodos. En Java, lo más cerca que puede llegar a una función pura es mediante el uso de un método estático:

```
public class Utils {
   static void DiHola() {
      System.out.println("Hola!");
   }
}
```

Utils.DiHola() se puede llamar desde cualquier lugar sin crear primero una instancia de *Utils*. Como puede llamar a *DiHola()* sin crear un objeto primero, esta referencia no existe. Sin embargo, esta todavía no es una función en el sentido de *di_hola()* de Python.

11.4 Herencia y Polimorfismo

La herencia y el polimorfismo son dos conceptos fundamentales en la programación orientada a objetos.

La herencia permite a los objetos derivar atributos y funcionalidad de otros objetos, creando una jerarquía que se mueve de objetos más generales a más específicos. Por ejemplo, un automóvil y un bote son tipos específicos de vehículos. Los objetos pueden heredar su comportamiento de un único objeto principal o de varios objetos principales, y se denominan objetos secundarios cuando lo hacen.

El polimorfismo permite que dos o más objetos se comporten entre sí, lo que permite que se usen indistintamente. Por ejemplo, si un método o función sabe cómo pintar un objeto Vehículo, también puede pintar un objeto Coche o Barco, ya que heredan sus datos y comportamiento del Vehículo.

Estos conceptos fundamentales de POO se implementan de manera bastante diferente en Python vs Java.

Herencia Python admite la herencia múltiple o la creación de clases que heredan el comportamiento de más de una clase principal.

Para ver cómo funciona esto, actualicemos la clase de Carro dividiéndola en dos categorías, una para vehículos y otra para dispositivos que usan electricidad:

```
class Vehiculo:
  def __init__(self, color, modelo):
    self.color = color
    self.modelo = modelo
class Dispositivo:
  def __init__(self):
    self._voltaje = 12
class Carro(Vehiculo, Dispositivo):
  def __init__(self, color, modelo, anio):
    Vehiculo.__init__(self, color, modelo)
    Dispositivo.__init__(self)
    self.anio = anio
  @property
  def voltaje(self):
    return self._voltaje
```

```
@voltaje.setter
def voltaje(self, volts):
  print("Aviso: esto puede causar problemas!")
  self._voltaje = volts
@voltaje.deleter
def voltaje(self):
  print("Aviso: el radio dejara de trabajar!")
  del self._voltaje
```

Un vehículo se define por tener atributos *.color* y *.modelo*. Entonces, un dispositivo se define para tener un atributo *._voltaje*. Dado que el objeto Carro original tenía estos tres atributos, se puede redefinir para heredar las clases Vehiculo y Dispositivo. Los atributos de color, modelo y _voltaje serán parte de la nueva clase Carro.

En *.__init__()* para Carro, llama a los métodos *.__init__()* para ambas clases principales para asegurarse de que todo se inicialice correctamente. Una vez hecho esto, puede agregar cualquier otra funcionalidad que desee a su Carro. En este caso, agrega un atributo *.anio* específico para los objetos Carro y los métodos *getter* y *setter* para *.voltaje*.

Funcionalmente, la nueva clase Carro se comporta como siempre. Creas y usa los objetos Carro como antes:

```
>>> from carro import *
>>> mi_carro = Carro("amarillo", "beetle", 1969)
>>> print(f"Mi carro es {mi_carro.color}")
Mi carro es amarillo
>>> print(f"Mi carro usa {mi_carro.voltaje} volts")
Mi carro usa 12 volts
>>> mi_carro.voltaje = 6
Aviso: esto puede causar problemas!
>>> print(f"Mi carro ahora usa {mi_carro.voltaje} volts")
Mi carro ahora usa 6 volts
```

Java, por otro lado, solo admite herencia única, lo que significa que las clases en Java pueden heredar datos y comportamiento de una sola clase principal. Sin embargo, los objetos Java pueden heredar el comportamiento de muchas interfaces diferentes. Las interfaces proporcionan un grupo de métodos relacionados que un objeto debe implementar y permiten que varias clases secundarias se comporten de manera similar.

Para ver esto en acción, divida la clase Java Carro en una clase principal y una interfaz:

```
public class Vehiculo {
   private String color;
   private String modelo;
   public Vehiculo(String color, String modelo) {
      this.color = color;
      this.modelo = modelo;
   }
   public String getColor() {
      return color;
   }
   public String getModelo() {
      return modelo;
   }
}
public interface Dispositivo {
   int getVoltaje();
}
public class Carro extends Vehiculo implements Dispositivo {
   private int voltaje;
   private int anio;
   public Car(String color, String modelo, int anio) {
      super(color, modelo);
      this.anio = anio;
      this.voltaje = 12;
   }
   @Override
   public int getVoltaje() {
      return voltaje;
   }
   public int getAnio() {
      return anio;
   }
}
```

Recordemos que cada clase e interfaz debe vivir en su propio archivo.

Como hicimos con Python, creamos una nueva clase llamada Vehículo para contener los datos y la funcionalidades más generales relacionados con el vehículo. Sin embargo, para agregar la funcionalidad del dispositivo, debe crear una interfaz en su lugar. Esta interfaz define un método único para devolver el voltaje del dispositivo.

La redefinición de la clase *Carro* requiere que heredes de *Vehiculo* usando extender e implemente la interfaz de *Dispositivo* usando implements. En el constructor, llama al constructor de la clase padre usando el *super()* incorporado. Dado que solo hay una clase principal, solo puede hacer referencia al constructor del vehículo. Para implementar la interfaz, escribe *getVoltaje()* usando la anotación *@Override*.

En lugar de obtener la reutilización del código de Dispositivo como lo hizo Python, Java requiere que implemente la misma funcionalidad en cada clase que implementa la interfaz. Las interfaces solo definen los métodos, no pueden definir datos de instancia o detalles de implementación.

Entonces, ¿por qué es este el caso de Java? Todo se reduce a tipos.

Tipos y Polimorfismo La estricta verificación de tipos de Java es lo que impulsa el diseño de su interfaz.

Cada clase e interfaz en Java es un tipo. Por lo tanto, si dos objetos Java implementan la misma interfaz, se considera que son del mismo tipo con respecto a esa interfaz. Este mecanismo permite usar indistintamente diferentes clases, que es la definición de polimorfismo.

Puede implementar la carga del dispositivo para sus objetos Java creando un .cargar() que necesita un dispositivo para cargarse. Cualquier objeto que implemente la interfaz del dispositivo se puede pasar a .cargar(). Esto también significa que las clases que no implementan Dispositivo generarán un error de compilación.

Cree la siguiente clase en un archivo llamado Rinoceronte.java:

```
public class Rinoceronte {
}
```

Ahora puede crear un nuevo *Main.java* para implementar *.cargar()* y explorar en qué se diferencian los objetos Carro y Rinoceronte:

```
public class Main{
```

```
    public static void cargar(Dispositivo dispositivo) {
      dispositivo.getVoltaje();
    }
    public static void main(String[] args) throws Exception {
      Carro carro = new Carro("amarillo", "beetle", 1969);
      Rinoceronte rinoceronte = new Rinoceronte();
      cargar(carro);
      cargar(rinoceronte);
    }
  }
```

Esto es lo que debería ver cuando intente compilar este código:

```
Information:2021-02-02 15:20 - Compilation completed with
1 error and 0 warnings in 4 s 395 ms
Main.java
Error:(43, 11) java: incompatible types: Rinoceronte cannot
be converted to Dispositivo
```

Dado que la clase Rinoceronte no implementa la interfaz del dispositivo, no se puede pasar a *.cargar()*.

En contraste con la escritura de variables estricta de Java, Python usa un concepto llamado escritura de pato, que en términos básicos significa que si una variable "camina como un pato y grazna como un pato, entonces es un pato". En lugar de identificar objetos por tipo, Python examina su comportamiento.

Podemos explorar la escritura de pato implementando capacidades de carga de dispositivo similares para la clase de Dispositivo Python:

```
>>> def cargar(dispositivo):
... if hasattr(Dispositivo, '_voltaje'):
... print(f"Cargando a {dispositivo._voltaje} voltaje disposi-
tivo ")
... else:
... print(f"No se puede cargar {dispositivo.__class__.__name__}")
...
>>> class Telefono(Dispositivo):
... pass
```

```
...
>>> class Rinoceronte:
... pass
...
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> mi_telefono = Telefono()
>>> mi_Rinoceronte = Rinoceronte()
>>> cargar(mi_carro)
Cargando 12 volts dispositivo
>>> cargar(mi_telefono)
Cargando a 12 volts Dispositivo
>>> cargar(mi_Rinoceronte)
No se puede cargar Rinoceronte
```

cargar() debe verificar la existencia del atributo *._voltaje* en el objeto al que se le pasa. Dado que la clase Dispositivo define este atributo, cualquier clase que herede de él (como Carro y Telefono) tendrá este atributo y, por lo tanto, mostrará que se está cargando correctamente. Las clases que no heredan de Dispositivo (como Rinoceronte) pueden no tener este atributo y no podrán cargar (lo cual es bueno, ya que cargar rinocerontes puede ser peligroso).

Métodos Predeterminados Todas las clases de Java descienden de la clase Object, que contiene un conjunto de métodos que heredan todas las demás clases. Las subclases pueden anularlas o mantener los valores predeterminados. La clase Object define los siguientes métodos:

```
class Object {
   boolean equals(Object obj) { ... }
   int hashCode() { ... }
   String toString() { ... }
}
```

De forma predeterminada, *equals()* compara las direcciones del objeto actual con un segundo objeto pasado, y *hashcode()* calcula un identificador único que también usa la dirección del objeto actual. Estos métodos se utilizan en muchos contextos diferentes en Java. Por ejemplo, las clases de

servicios públicos, como las colecciones que clasifican objetos en función del valor, necesitan ambas.

toString() devuelve una representación de cadena del objeto. De forma predeterminada, este es el nombre de la clase y la dirección. Este método se llama automáticamente cuando se pasa un objeto a un método que requiere un argumento de cadena, como *System.out.println()*:

```
Carro carro = new Carro("amarillo", "Beetle", 1969);
System.out.println(carro);
```

Al ejecutar este código, se usará el *.toString()* predeterminado para mostrar el objeto de Carro:

```
Carro@61bbe9ba
```

No es muy útil, ¿verdad? Puede mejorar esto anulando el *.toString()* predeterminado. Agregue este método a la clase Java Carro:

```
public String toString() {
   return "Carro: " + getColor() + " : " + getmodelo() + "
: " + getanio();
}
```

Ahora, cuando ejecutemos el mismo código de muestra, veremos lo siguiente:

```
Carro: amarillo : Beetle : 1969
```

Python proporciona una funcionalidad similar con un conjunto común de *dunder* (abreviatura de "doble subrayado") de métodos. Cada clase de Python hereda estos métodos y puede anularlos para modificar su comportamiento.

Para las representaciones de cadena de un objeto, Python proporciona __repr__() y *__str__()*, que puede conocer en Pythonic OOP String Conversion: *__repr__* vs *__str__*. La representación inequívoca de un objeto es devuelta por *__repr__()*, mientras que *__str__()* devuelve una representación legible por humanos. Estos son aproximadamente análogos a *.hashcode()* y *.toString()* en Java.

Al igual que Java, Python proporciona implementaciones predeterminadas de estos métodos *dunder*:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> print(repr(mi_carro))
<Carro.Carro object at 0x7fe4ca154f98>
>>> print(str(mi_carro))
<Carro.Carro object at 0x7fe4ca154f98>
```

Puede mejorar esta salida anulando *.__str__()*, agregando esto a la clase Python Carro:

```
def __str__(self):
  return f'Carro {self.color} : {self.modelo} : {self.anio}'
```

Esto le da un resultado mucho mejor:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> print(repr(mi_carro))
<Carro.Carro object at 0x7f09e9a7b630>
>>> print(str(mi_carro))
Carro amarillo : Beetle : 1966
```

Anular el método *dunder* nos dio una representación más legible de su Carro. Es posible que también desee anular el *.__repr__()*, ya que a menudo es útil para depurar.

Python ofrece muchos más métodos *dunder*. Usando métodos *dunder*, puede definir el comportamiento de su objeto durante la iteración, comparación, adición o hacer que un objeto sea invocable directamente, entre otras cosas.

Sobrecarga del Operador La sobrecarga de operadores se refiere a redefinir cómo funcionan los operadores de Python cuando operan en objetos definidos por el usuario. Los métodos *dunder* de Python le permiten implementar la sobrecarga de operadores, algo que Java no ofrece en absoluto.

Modificando la clase Python Carro con los siguientes métodos adicionales:

```
class Carro:
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
```

```
def __str__(self):
  return f'Carro {self.color} : {self.modelo} : {self.anio}'
def __eq__(self, other):
  return self.anio == other.anio
def __lt__(self, other):
  return self.anio < other.anio
  def __add__(self, other):
  return Carro(self.color + other.color,
    self.modelo + other.modelo,
    int(self.anio) + int(other.anio))
```

La siguiente tabla muestra la relación entre estos métodos *dunder* y los operadores de Python que representan:

Método Dunder	Operator	Propósito
__eq__	==	Revisa si son del mismo año
__lt__	<	Cual carro es un modelo más reciente
__add__	+	Concatena dos objetos Carro de una manera absurda

Cuando Python ve una expresión que contiene objetos, llama a los métodos *dunder* definidos que correspondan a los operadores de la expresión. El siguiente código utiliza estos nuevos operadores aritméticos sobrecargados en un par de objetos Carro:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> tu_carro = Carro("rojo", "Corvette", "1967")
>>> print (mi_carro < tu_carro)
True
>>> print (mi_carro > tu_corro)
False
>>> print (mi_carro == tu_carro)
False
>>> print (mi_carro + tu_Carro)
Car amarillorojo : BeetleCorvette : 3933
```

Hay muchos más operadores que puede sobrecargar usando métodos *dunder*. Ofrecen una forma de enriquecer el comportamiento de su objeto de una manera que los métodos predeterminados de la clase base común de Java no lo hacen.

11.5 Reflexión

La reflexión se refiere a examinar un objeto o clase desde dentro del objeto o clase. Tanto Java como Python ofrecen formas de explorar y examinar los atributos y métodos de una clase.

Examinar el Tipo de un Objeto Ambos lenguajes tienen formas de probar o verificar el tipo de un objeto.

En Python, usa *type()* para mostrar el tipo de una variable, e *isinstance()* para determinar si una variable dada es una instancia o hija de una clase específica:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> print(type(mi_carro))
<class 'Carro.Carro'>
>>> print(isinstance(mi_carro, Carro))
True
>>> print(isinstance(mi_carro, Dispositivo))
True
```

En Java, consulta el objeto por su tipo usando *.getClass()*, y usa el operador *instanceof* para verificar una clase específica:

```
Carro carro = new Carro("amarillo", "beetle", 1969);
System.out.println(carro.getClass());
System.out.println(carro instanceof Carro);
```

Este código genera la siguiente salida:

```
class com.realpython.Carro
true
```

Examinar los Atributos de un Objeto En Python, puede ver todos los atributos y funciones contenidos en cualquier objeto (incluidos todos los métodos *dunder*) usando *dir()*. Para obtener los detalles específicos de un atributo o función determinados, use *getattr()*:

```
>>> print(dir(mi_carro))
['_Carro__portavasos', '__add__', '__class__', '__de-
lattr__', '__dict__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__',
'__gt__', '__hash__', '__init__', '__init_subclass__',
'__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__re-
duce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__',
'__weakref__',
'_voltaje', 'color', 'modelo', 'voltaje', 'ruedas', 'anio']
>>> print(getattr(mi_carro, "__format__"))
<built-in method __format__ of Carro object at 0x7fb4c10f5438>
```

Java tiene capacidades similares, pero el control de acceso del lenguaje y la seguridad de tipos lo hacen más complicado de recuperar.

.getFields() recupera una lista de todos los atributos de acceso público. Sin embargo, dado que ninguno de los atributos de Carr es público, este código devuelve una matriz vacía:

```
Field[] fields = carro.getClass().getFields();
```

Java trata los atributos y métodos como entidades independientes, por lo que los métodos públicos se recuperan mediante *.getDeclaredMethods()*. Dado que los atributos públicos tendrán un método *.get* correspondiente, una forma de descubrir si una clase contiene una propiedad específica podría verse así:

> Utilice *.getDeclaredMethods()* para generar una matriz de todos los métodos.
>
> Recorra todos los métodos devueltos:
>
> Para cada método descubierto, devuelve verdadero si el método:
>
> Comienza con la palabra get OR acepta cero argumentos
>
> Y no vuelve vacío
>
> Y incluye el nombre de la propiedad
>
> De lo contrario, devuelve falso.

Aquí hay un ejemplo rápido y sucio:

```
    public static boolean getProperty(String name, Object ob-
ject) throws Exception {
      Method[] declaredMethods = object.getClass().getDeclaredMethods();
      for (Method method : declaredMethods) {
        if (isGetter(method) &&
          method.getName().toUpperCase().contains(name.toUpperCase()))
{
            return true;
        }
      }
      return false;
    }
    // Función auxiliar para obtener si el método es un método
getter
    public static boolean isGetter(Method method) {
      if ((method.getName().startsWith("get") ||
        method.getParameterCount() == 0 ) &&
        !method.getReturnType().equals(void.class)) {
          return true;
      }
      return false;
    }
```

getProperty() es su punto de entrada. Llamandolo con el nombre de un atributo y un objeto. Devuelve verdadero si se encuentra la propiedad y falso si no.

Métodos de Llamada a Través de la Reflexión Tanto Java como Python proporcionan mecanismos para llamar a métodos mediante la reflexión.

En el ejemplo de Java anterior, en lugar de simplemente devolver verdadero si se encuentra la propiedad, puede llamar al método directamente. Recuerde que *getDeclaredMethods()* devuelve una matriz de objetos *Method*. El objeto *Method* en sí tiene un método llamado *.invoke()*, que llamará al método. En lugar de devolver true cuando el método correcto se encuentra en la línea 7 anterior, puede devolver *method.invoke(object)* en su lugar.

Esta capacidad también existe en Python. Sin embargo, dado que Python no distingue entre funciones y atributos, debe buscar específicamente entradas que sean invocables:

```
>>> for method_name in dir(mi_carro):
... if callable(getattr(mi_carro, method_name)):
... print(method_name)
...
__add__
__class__
__delattr__
__dir__
__eq__
__format__
__ge__
__getattribute__
__gt__
__init__
__init_subclass__
__le__
__lt__
__ne__
__new__
__reduce__
__reduce_ex__
__repr__
__setattr__
__sizeof__
__str__
__subclasshook__
```

Los métodos de Python son más simples de administrar y llamar que en Java. Agregar el operador () (y cualquier argumento requerido) es todo lo que necesita hacer.

El siguiente código encontrará el *.__str__ ()* de un objeto y lo llamará a través de la reflexión:

```
>>> for method_name in dir(mi_carro):
... attr = getattr(mi_carro, method_name)
... if callable(attr):
... if method_name == '__str__':
... print(attr())
...
Carro amarillo : Beetle : 1966
```

Aquí, se comprueba cada atributo devuelto por *dir()*. Obtiene el objeto de atributo real usando *getattr()* y verifica si es una función invocable usando *callable()*. Si es así, compruebe si su nombre es *__str__()* y luego llámelo.

12 Apéndice A: Software Libre y Propietario

Con el constante aumento de la comercialización de equipos de cómputo y/o comunicación (teléfonos inteligentes, tabletas, computadoras portátiles y de escritorio, etc.) y su relativo bajo costo, estos equipos se han convertido en objetos omnipresentes en nuestra vida diaria, ya que estos permiten realizar un creciente número de actividades cotidianas de miles de millones de usuarios.

Dichos equipos de cómputo y/o comunicación por sí solos tienen poca utilidad, pero su uso en conjunción con el Software adecuado forman un dúo que nos ha permitido tener los avances de los que actualmente disfrutamos. El Software -sistema operativo y los programas de aplicaciones- son los que realmente generan las soluciones al interactuar uno o más paquetes informáticos con los datos del usuario. También, es común que al comprar un equipo de cómputo y/o comunicación, en el costo total, se integre el del sistema operativo, aplicaciones ofimáticas y de antivirus, sean estos usados por el usuario o no y en la mayoría de los casos no es posible solicitar que no sean incluidos en el costo del equipo.

Por otro lado, el Software comercial suele quedar obsoleto muy rápido, ya que constantemente se le agregan nuevas funcionalidades al mismo y estas en general son vendidas como versiones independientes de la adquirida originalmente. Esto obliga al usuario -si quiere hacer uso de ellas- a comprar las nuevas versiones del Software para satisfacer sus crecientes necesidades informáticas y la obsolescencia programada.

Por lo anterior y dada la creciente complejidad de los paquetes de cómputo y el alto costo de desarrollo de aplicaciones innovadoras, en muchos casos, el costo total del Software que comúnmente los usuarios instalan -y que no necesariamente usan las capacidades avanzadas del programa, por las cuales el Software tiene un alto costo comercial- en sus equipos, suele ser más caro que el propio equipo en el que se ejecutan.

Hoy en día los usuarios disponemos de dos grandes opciones para adquirir el Software necesario para que nuestros equipos funcionen, a saber:

- Por un lado, podemos emplear programas comerciales (Software propietario), de los cuales no somos dueños del Software, sólo concesionarios al adquirir una licencia de uso del Software y nos proporcionan un instalable del programa adquirido. La licencia respectiva es en la gran mayoría de los casos muy restrictiva, ya que restringe su uso a un solo

equipo y/o usuario simultáneamente.

- Por otro lado, existe el Software libre[115], desarrollado por usuarios y para usuarios que, entre otras cosas, comparten los códigos fuente, el programa ejecutable y dan libertades para estudiar, adaptar y redistribuir a quien así lo requiera el programa y todos sus derivados.

Sobre la Obsolescencia Programada Es un conjunto de estrategias deliberadas destinadas a asegurarse que la versión actual de un determinado producto quedará desfasada o inservible en un plazo de tiempo predeterminado. De esta manera, los fabricantes se aseguran que los consumidores se verán obligados a reemplazarlo aunque funcione adecuadamente.

La obsolescencia puede lograrse mediante la introducción de un modelo con características superiores o diseñando intencionadamente un producto para que deje de funcionar correctamente en un plazo determinado. En cualquiera de los dos casos, se espera que los consumidores opten por el nuevo producto de la misma marca. Muchas veces la obsolescencia no es sobre el propio producto sino aplicando restricciones al producto de un competidor con la ayuda de una tercera empresa.

Tipos de Obsolescencia Programada Podemos dividir la obsolescencia programada en 4 tipos:

1- Establecimiento artificial del plazo de duración: Los productos se fabrican con piezas cuya duración tienen una vida útil limitada cuando, si se usaran otras de calidad superior ese plazo se extendería.

2- Actualizaciones de Software: Los desarrolladores de Software sacan nuevas versiones de sus aplicaciones que en un momento determinado dejan de ser compatibles con dispositivos antiguos. En muchos casos se ha podido comprobar que esa incompatibilidad es absolutamente artificial ya que al «engañar» al Software este funcionaba sin problemas.

[115] A veces también se han usado términos como FOSS y FLOSS. Ambas cosas son similares, ya que FOSS (Free and Open Source Software) traducido como "Software de código abierto" y FLOSS (Free/Libre and Open Source Software) "Software libre y de código abierto". Según quienes adoptan estos términos, lo hacen por tener una imparcialidad entre la carga filosófica del Software libre y el aspecto técnico y/o las ventajas que brinda este modelo de desarrollo. Richard Stallman nos invita a no usarlas y no se trata de un ad hóminem. Stallman y el proyecto GNU nos aconsejan que hablemos siempre de Software libre y aquí no cabe imparcialidad.

3- Obsolescencia percibida: Esta es una táctica psicológica, se trata de convencer al consumidor mediante publicidad y el uso de influenciadores de que el producto que se tiene actualmente está viejo y que se necesita uno nuevo. Como por ejemplo: ¿cuantos megapíxeles necesitas en tu teléfono para sacar una buena foto de tu mascota?

4- Trabas a la reparación: En el caso de los teléfonos por ejemplo, lo de impedir sacar la batería (con la excusa de hacer los teléfonos más delgados) es una forma de obligar a los consumidores a recurrir a los servicios oficiales y a disuadirlos de reemplazarlas por sustitutos más económicos. Otras tácticas son la utilización de piezas no estándar o que necesitan herramientas específicas para la reparación. Muchas veces se suele restringir el acceso a estas piezas o hacer una reducida producción de las mismas para aumentar artificialmente el costo.

Ejemplos de Obsolescencia Programada

- iPhone cada vez más lentos: La Justicia francesa comprobó que actualizaciones de Software hacían cada vez más lento el rendimiento de los modelos más viejos. La empresa le echó la culpa a las baterías, pero pagó una compensación de decenas de millones de dólares. Además rebajó los precios de sus baterías de repuesto para que los teléfonos fueran más rápidos con el nuevo Software y se comprometió a hacer más en el futuro para garantizar que los teléfonos no volvieran a ser más lentos. Con la salida de un nuevo modelo de teléfono cada año, seguro que hay algo de obsolescencia planificada en alguna parte.

- Impresoras: Esto es algo que todos conocemos. Muchas veces nos encontramos con impresoras a precio rebajado, pero al momento de tener que comprar un cartucho de tinta nos encontramos con que este tiene un precio igual o superior a comprar una nueva. Además, se ponen restricciones a la recarga o al uso de cartuchos alternativos. Hubo denuncias de que algunos modelos dejaban de funcionar a partir de cierta cantidad de páginas impresas o cierto tiempo desde la primera impresión.

- Certificados de seguridad: Por ejemplo, el pasado 30 de septiembre de 2021 caduco otro certificado de autentificación (CA de DST Root CA X3 de Let's Encrypt) que ayudaba a validar la conexión en internet a

los dispositivos que no fueron actualizados a otro certificado más actual -en la mayoría de los casos por no ser del interés económico de sus creadores-. Esto ocasionó que millones de dispositivos (teléfonos inteligentes, Smart TV, tabletas, computadoras portátiles y de escritorio, etc.) con algunos años de ser creados y perfectamente funcionales dejarán de conectarse a internet de un día para otro, forzando a sus dueños a desechar el dispositivo por carecer del servicio de internet en las aplicaciones instaladas.

- Cambio de la versión del sistema operativo: En el caso del sistema operativo Windows 10 a 11, la solicitud de requisitos mínimos de Hardware es para muchos equipos excesivo, ya que se estima que dejará fuera en su actualización a casi todos los equipos con más de 4 años de antigüedad por no contar por ejemplo con el Chip TPM 2.0 o GPU compatible con DirectX 12, siendo perfectamente funcionales con la versión actual del sistema operativo. Si bien Windows 10 seguirá con soporte hasta 2025, los usuarios que deseen tener las nuevas características del sistema operativo tendrán que cambiar de equipo.

12.1 Software Propietario

No existe consenso sobre el término a utilizar para referirse al opuesto del Software libre. La expresión «Software propietario (Proprietary Software)» (véase [14]), en la lengua anglosajona, "Proprietary" significa «poseído o controlado privadamente (Privately Owned and Controlled)», que destaca la manutención de la reserva de derechos sobre el uso, modificación o redistribución del Software. Inicialmente utilizado, pero con el inconveniente de que la acepción proviene de una traducción literal del inglés, no correspondiendo su uso como adjetivo en el español, de manera que puede ser considerado como un barbarismo.

El término "propietario" en español resultaría inadecuado, pues significa que «tiene derecho de propiedad sobre una cosa», por lo que no podría calificarse de "propietario" al Software, porque éste no tiene propiedad sobre nada (es decir, no es dueño de nada) y además, no podría serlo (porque es una cosa y no una persona). Así mismo, la expresión "Software propietario" podría ser interpretada como: "Software sujeto a propiedad" (derechos o titularidad) y su opuesto, el Software libre, también está sujeto al derecho de autor. Otra interpretación es que contrariamente al uso popular del término, se puede

afirmar que "todo Software es propietario", por lo que la forma correcta de referirse al Software con restricciones de uso, estudio, copia o mejora es la de Software privativo, según esta interpretación el término "propietario" podría aplicarse tanto para Software libre como Software privativo, ya que la diferencia entre uno y otro está en que el dueño del Software privativo lo licencia como propiedad privada y el de Software libre como propiedad social.

Con la intención de corregir el defecto de la expresión "Software propietario" aparece el llamado "Software con propietario", sin embargo se argumenta contra el término "con propietario" y justamente su similitud con Proprietary en inglés, que sólo haría referencia a un aspecto del Software que no es libre, manteniendo una de las principales críticas a éste (de "Software sujeto a derechos" o "propiedad"). Adicionalmente, si "propietario" se refiere al titular de los derechos de autor -y está claro que no se puede referir al usuario, en tanto éste es simplemente un cesionario-, no resuelve la contradicción: todo el Software libre tiene también titulares de derechos de autor.

La expresión Software no libre (en inglés Non-Free Software) es usado por la FSF para agrupar todo el Software que no es libre, es decir, incluye al llamado en inglés "Semi-Free Software" (Software semilibre) y al "Propietary Software". Asimismo, es frecuentemente utilizado para referirse al Software que no cumple con las Directrices de Software libre de Debian GNU/Linux, las cuales siguen la misma idea básica de libertad en el Software, propugnada por la FSF y sobre las cuales está basada la definición de código abierto de la Open Source Initiative.

Adicionalmente el Software de código cerrado nace como antónimo de Software de código abierto y por lo tanto se centra más en el aspecto de ausencia de acceso al código que en los derechos sobre el mismo, éste se refiere sólo a la ausencia de una sola libertad por lo que su uso debe enfocarse sólo a este tipo de Software y aunque siempre signifique que es un Software que no es libre, no tiene que ser Software de código cerrado.

La expresión Software privado es usada por la relación entre los conceptos de tener y ser privado. Este término sería inadecuado debido a que, en una de sus acepciones, la palabra "privado" se entiende como antónimo de "público", es decir, que «no es de propiedad pública o estatal, sino que pertenece a particulares», provocando que esta categoría se interpretará como no referente al Estado, lo que produciría la exclusión del Software no libre generado por el aparato estatal. Además, el "Software público" se asocia generalmente con Software de dominio público.

12.2 Software Libre

La definición de Software libre (véase [16], [17], [12], [13], [11] y [15]) estipula los criterios que se tienen que cumplir para que un programa sea considerado libre. De vez en cuando se modifica esta definición para clarificarla o para resolver problemas sobre cuestiones delicadas. «Software libre» significa que el Software respeta la libertad de los usuarios y la comunidad. En términos generales, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el Software. Con estas libertades, los usuarios -tanto individualmente como en forma colectiva- controlan el programa y lo que hace.

Cuando los usuarios no controlan el programa, el programa controla a los usuarios. Los programadores controlan el programa y a través del programa, controlan a los usuarios. Un programa que no es libre, llamado «privativo o propietario», es considerado por muchos como un instrumento de poder injusto.

El Software libre es la denominación del Software que respeta la libertad de todos los usuarios que adquirieron el producto y por tanto, una vez obtenido el mismo puede ser usado, copiado, estudiado, modificado y redistribuido libremente de varias formas. Según la Free Software Foundation (véase [16]), el Software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir y estudiar el mismo, e incluso modificar el Software y distribuirlo modificado.

Un programa es Software libre si los usuarios tienen las cuatro libertades esenciales:

0. La libertad de usar el programa, con cualquier propósito.

1. La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.

2. La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.

3. La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Un programa es Software libre si los usuarios tienen todas esas libertades. Por tanto, el usuario debe ser libre de redistribuir copias, tanto con o sin modificaciones, ya sea gratuitamente o cobrando una tarifa por la distribución,

a cualquiera en cualquier parte. El ser libre de hacer estas cosas significa, entre otras cosas, que no tiene que pedir ni pagar el permiso.

También debe tener la libertad de hacer modificaciones y usarlas en privado para su propio trabajo o pasatiempo, sin siquiera mencionar que existen. Si publica sus cambios, no debe estar obligado a notificarlo a nadie en particular, ni de ninguna manera.

La libertad de ejecutar el programa significa que cualquier tipo de persona u organización es libre de usarlo en cualquier tipo de sistema de computación, para cualquier tipo de trabajo y finalidad, sin que exista obligación alguna de comunicarlo al programador ni a ninguna otra entidad específica. En esta libertad, lo que importa es el propósito de los usuarios, no el de los programadores. El usuario es libre de ejecutar el programa para alcanzar sus propósitos y si lo distribuye a otra persona, también esa persona será libre de ejecutarlo para lo que necesite; nadie tiene derecho a imponer sus propios objetivos.

La libertad de redistribuir copias debe incluir las formas binarias o ejecutables del programa, así como el código fuente, tanto para las versiones modificadas como para las que no lo estén. Distribuir programas en forma de ejecutables es necesario para que los sistemas operativos libres se puedan instalar fácilmente. Resulta aceptable si no existe un modo de producir un formato binario o ejecutable para un programa específico, dado que algunos lenguajes no incorporan esa característica, pero debe tener la libertad de redistribuir dichos formatos si encontrara o programara una forma de hacerlo.

Para que se de la libertad que se menciona en los puntos 1 y 3 de realizar cambios y publicar las versiones modificadas tenga sentido, el usuario debe tener acceso al código fuente del programa. Por consiguiente, el acceso al código fuente es una condición necesaria para el Software libre. El «código fuente» compilado no es código fuente real y no cuenta como código fuente.

La libertad 1 incluye la libertad de usar su versión modificada en lugar de la original. Si el programa se entrega con un producto diseñado para ejecutar versiones modificadas de terceros, pero rechaza ejecutar las suyas, una práctica conocida como «tivoización» o «arranque seguro» [«Lockdown»] la libertad 1 se convierte más en una ficción teórica que en una libertad práctica, esto no es suficiente, en otras palabras, estos binarios no son Software libre, incluso si se compilaron desde un código fuente que es libre.

Una manera importante de modificar el programa es agregándole subrutinas y módulos libres ya disponibles. Si la licencia del programa especifica que no se pueden añadir módulos que ya existen y que están bajo una licencia

apropiada, por ejemplo si requiere que usted sea el titular de los derechos de autor del código que desea añadir, entonces se trata de una licencia demasiado restrictiva como para considerarla libre.

La libertad 3 incluye la libertad de publicar sus versiones modificadas como Software libre. Una licencia libre también puede permitir otras formas de publicarlas; en otras palabras, no tiene que ser una licencia de Copyleft. No obstante, una licencia que requiera que las versiones modificadas no sean libres, no se puede considerar libre.

«Software libre» no significa que «no es comercial». Un programa libre debe estar disponible para el uso comercial, la programación comercial y la distribución comercial. La programación comercial de Software libre ya no es inusual; el Software libre comercial es muy importante, ejemplo de ello es la empresa RedHat (ahora propiedad de IBM). Puede haber pagado dinero para obtener copias de Software libre, o puede haber obtenido copias sin costo. Pero sin tener en cuenta cómo obtuvo sus copias, siempre tiene la libertad de copiar y modificar el Software, incluso de vender copias.

El término Software no libre se emplea para referirse al Software distribuido bajo una licencia de Software más restrictiva que no garantiza estas cuatro libertades. Las leyes de la propiedad intelectual reservan la mayoría de los derechos de modificación, duplicación y redistribución para el dueño del Copyright; el Software dispuesto bajo una licencia de Software libre rescinde específicamente la mayoría de estos derechos reservados.

Los manuales de Software deben ser libres por las mismas razones que el Software debe ser libre y porque de hecho los manuales son parte del Software. También tiene sentido aplicar los mismos argumentos a otros tipos de obras de uso práctico, es decir, obras que incorporen conocimiento útil, tal como publicaciones educativas y de referencia. Wikipedia es el ejemplo más conocido.

La lista de proyectos de este tipo es realmente impresionante, algunos han conseguido un uso y alta calidad, por ejemplo el compilador GCC, el Kernel de Linux y el sistema operativo Debian GNU/Linux y Android. Mientras que otros proyectos han caído en el olvido, pero de la gran mayoría se tiene copia del código fuente que permitiría a quienes estén interesados en dicho proyecto poder reusarlo y en su caso ampliarlo.

La característica más importante que aparece típicamente en un proyecto de este tipo, es que un conjunto de personas separadas a gran distancia, sean capaces, a través de la Web, de los E-mail y de foros de aunar sus esfuerzos para crear, mejorar y distribuir un producto, de forma que todos

ellos se benefician unos de otros. Evidentemente, la mayor parte del peso recae en los desarrolladores, pero también es necesaria una difusión para que los usuarios documenten, encuentren errores, hagan foros de discusión, etc.

Si bien, el Software libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia estriba en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución, y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar fácilmente si alguien introdujo código malicioso a una determinada aplicación.

¿Por qué se Interesan los Autores, Alumnos y Profesores Universitarios en el Software Libre? La ventaja principal es porque bajo el Software libre subyace la idea de compartir conocimiento y favorecer la existencia de nuevas ideas[116]; y ¿qué es investigar y enseñar?, sino crear conocimiento y procurar que los alumnos aprendan e incluso vayan más allá de lo aprendido. Se comparte la idea, que el espíritu del Software libre es similar al que debería reinar en las instituciones educativas:

- Porque así no se condiciona a los estudiantes a usar siempre lo mismo.
- No se fomenta la piratería en los estudiantes y se evita pagar licencias que no son necesarias al existir alternativas gratuitas.
- Es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.
- Se ofrece libertad de elección a los estudiantes y profesores al no limitarlos a usar una solución determinada, ampliando sus opciones y permitiendo un mayor aprendizaje.

Concretando estas ideas, profesores e investigadores necesitan herramientas para la investigación y docencia y estas deben tener una calidad mínima y ser fácilmente distribuibles entre los alumnos. En muchos casos las compañías desarrolladoras y distribuidoras de programas de cómputo no han

[116] ¿Por qué el Software creado con dinero de los impuestos no se publica como Software Libre?

¡El código pagado por los ciudadanos debería estar disponible para los ciudadanos y el mismo gobierno!

sabido ofrecer sus productos con la flexibilidad adecuada para las labores docentes o, en otros casos, los productos desarrollados no tienen la calidad esperada.

El Software libre es aún joven, pese a las decenas de miles de proyectos actuales -en los que se trabaja constantemente en mejorar la parte computacional de los algoritmos involucrados en el proyecto, haciendo y puliendo interfaces gráficas, generando ayuda en línea así como la documentación necesaria para que usuarios noveles y avanzados usen la mayor cantidad de opciones programadas- existen muchas otras necesidades profesionales y de investigación que requieren el desarrollo innovador de programas de cómputo para automatizarlas y hacerlas eficientes. Esto queda plasmado en las decenas de proyectos que a diario son registrados en las páginas especializadas en busca de difusión y apoyo para su proyecto.

En los últimos años, muchos proyectos han pasado de ser simples programas en línea de comandos a complejas aplicaciones multiplataforma -se ejecutan en distintos sistemas operativos como son Windows, Linux, Unix, Mac OS, Android- con ambientes gráficos multimedia que en muchos casos han superado a sus contrapartes comerciales -por ejemplo los navegadores Web-. Para muestra de este maravilloso avance, tomemos el proyecto del sistema operativo Android, que actualmente se ejecuta en millones de equipos -como celulares, tabletas, electrodomésticos, etc.- y en los cuales se pueden descargar miles de aplicaciones y está soportado por una gran cantidad de usuarios y empresas comerciales como Google, IBM y últimamente Microsoft -que años atrás era acérrima enemiga del Software libre-.

El Software libre ha logrado desplazar a muchos de sus competidores por sus múltiples bondades y bajo costo de desarrollo -es el caso de Windows Phone que fue reemplazado por Android de Google-, al reusar miles de aplicaciones ya existentes que usan Software libre y permitir desarrollar otro tanto de aplicaciones bajo una plataforma que se ejecuta en los más diversos procesadores. Además, el uso de Software libre y su posibilidad de ampliarlo y/o especializarlo según sea necesario, ha permitido crear de forma cada vez más rápida y confiable; para poner a disposición de un gran público programas de uso común, así como especializado que satisfagan las nuevas necesidades de los usuarios.

Software Libre en Ciencia y Educación Algunos puntos y reflexiones sobre porqué se considera que es interesante el Software libre en Ciencia y

Educación son:

- Accesible a todo el mundo aunque no sea rentable su desarrollo: El Software libre entre sus libertades permite que se pueda ejecutar por terceros, copiarlo, distribuirlo y estudiarlo/modificarlo. Eso hace que si en ciencia se usa Software libre el acceso a esos programas no suponga una barrera (se puede distribuir y ejecutar).

- Muchos de los desarrollos en el campo de la accesibilidad se realizan en universidades y son distribuidos como Software libre: Aunque no sea rentable muchas veces el desarrollo de herramientas de accesibilidad (no sea algo monetizable) en la universidad se consigue escapar a esa la lógica capitalista de solamente invertir en lo que pueda ofrecer beneficio económico.

- Transparencia: En ciencia es importante ver las costuras para comprobar si es verdad lo que se afirma, tener acceso al código fuente del Software empleado permite poder estudiarlo por si realiza algún cálculo mal.

- Propicia el espíritu crítico: Si no tienes acceso a las revistas o el acceso es privativo para los bolsillos de mucha gente no puedes comprobar la información. Se nos pide que seamos críticos con la información que se nos da del mundo científico pero no podemos entrenar el espíritu crítico sin acceso al conocimiento libre.

- Caramelos con droga en la puerta del colegio: Muchas empresas buscan introducir en los colegios, institutos y universidad su Software. Ofrecer un programa que permita trabajar y genere una dependencia quedando los datos muchas veces en las nubes (ordenadores de otras personas). Un ejemplo es Microsoft con Office 365. Dando cuentas gratuitas durante un tiempo para que se use su Software. Otro ejemplo podría ser Unity3D en vez de por ejemplo Godot.

- Especificaciones de protocolos abiertas VS cerradas: Gracias a que los protocolos TCP/IP, HTTP, POP, SNMP, DHCP, etc. son abiertos es posible construir herramientas por cualquier con conocimientos de programación. Con protocolos cerrados solamente quienes tuvieran acceso a las especificaciones podrían desarrollar y conocer cómo funcionan.

- Uso de estándares: Existiendo un estándar para documentos ofimáticos (procesador de textos, hoja de cálculo, presentaciones, etc.) algunas empresas como Microsoft se empeñan en ir con su propio formato y estándar en vez de sumarse a que sea más sencillo ir a una y que el usuario pueda optar por que herramientas usar para editar o trabajar con documentos ofimáticos.

- Software libre para la Ciencia ciudadana: Un ejemplo en el que es importante la colaboración ciudadana es el cambio climático y la defensa medioambiental del territorio. Desde imvec.tech usan herramientas de Software libre para medición y monitorización de contaminación.

Software Libre: Beneficios Más Allá de la Informática El uso de las tecnologías de código abierto supone cultivar el conocimiento y la puesta en valor de la libertad individual, lo personal y lo privado, todo ello sin menospreciar lo público y la construcción de una sociedad. El movimiento del Software libre, con Linux a la cabeza, ha capitaneado durante décadas planteamientos para un cambio en el modo de producción: el abaratamiento de costes empresariales para grandes y pequeños, el trabajo en línea, el desarrollo de Software y Hardware a pequeña escala, el replanteamiento del negocio informático, el sistema de normas éticas que rigen los grupos, la documentación abierta, etc.

Todo ello derivado de una simple idea: la libertad. Ha sido la clave que ha llevado a todo este movimiento hacia una autonomía y motivación que pocas veces se ve en otros sectores. El Open Source está lleno de alternativas con la libertad como pilar y consecuencia filosófica, de hecho muchos Forks (derivaciones) de proyectos aparecen cuando la disputa sobre la misma toma relevancia. Esta manera de hacer las cosas debería trasladarse directamente a la sociedad promoviendo esos valores para evitar caer en un mundo despótico que toma fuerza a pasos agigantados.

No estaría mal promover la comprensión de las licencias libres. Leer y entender una licencia GPL, MIT o BSD es infinitamente más sencillo y rápido que hacerlo con otras, siendo unas normas fáciles de cumplir porque encajan con un modelo ético y práctico comprensible por muchos.

Podríamos decir que GPL, BSD y MIT son las Constituciones que vertebran todo el movimiento del Software Libre, cosechando derechos de uso y logros como el ahorro o la legítima copia privada. Lo mismo podría ocurrir con las licencias Creative Commons para cierto contenido, un arma poderosa

en el sector divulgativo que está poco extendida por desconocimiento y el Status Quo de la propiedad intelectual.

La cooperación libre y voluntaria supuso un éxito hasta ahora pero está siendo amenazada por la dinámica actual de la Web, donde la centralización de las principales plataformas supone estar bajo el yugo de normas que rastrean con lupa y cambian sus términos con demasiada frecuencia. Ya ni digamos cuando eso se junta con el ansia de monetización: si quieres dinero más te vale no cabrear a los anunciantes o no tener un Strike por uso de contenido que podría ser reclamado.

Los claroscuros de este sistema hace que los creadores cada vez hagan menos de forma libre y altruista, y ello podría verse potenciado por las búsquedas con inteligencia artificial, lo que apunta a un empobrecimiento del contenido cultural fresco y renovador. Las polémicas con GitHub Copilot o ChatGPT sobre el entrenamiento de las IAs hace sobrevolar una vez más la cuestión del Copyright y el uso legítimo de los resultados brindados por éstas, lo que también condiciona la creación.

La libertad de expresión, de uso, de creación, de modificación ... esas cuestiones llevan irremediablemente a una libertad de pensamiento y acción, a una adaptación creativa que puede ser la motivación para romper moldes en todos los aspectos. El código abierto y sus licencias son, por tanto, un beneficio personal y social mucho más grande que el simple hecho de usar Linux o Software libre. El contenido despreocupado, que no prioriza el dinero y el posicionamiento/visualizaciones, se vuelve esencial para el inconformismo.

12.3 Seguridad del Software

Si bien, el Software Libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia puede estribar en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar si alguien introdujo código malicioso a una determinada aplicación.

Pero de todos es sabido, que los usuarios de Software de código abierto, como por ejemplo los que de manera habitual trabajan con equipos comandados por sistemas Linux, por regla general se sienten orgullosos de la seguridad que estos programas aportan con respecto a los sistemas cerrados propios de otras firmas, dígase Microsoft Windows o Mac de Apple.

¿Es Seguro el Software Libre? En primer lugar definiremos el concepto de "seguridad" como salvaguarda de las propiedades básicas de la información. Entre las características que debe cumplir para ser seguro, encontramos la integridad, es decir, que sólo los usuarios autorizados pueden crear y modificar los componentes del sistema, la confidencialidad, sólo estos usuarios pueden acceder a esos componentes, la disponibilidad, que todos los componentes estén a disposición de los usuarios siempre que lo deseen y el "no repudio", o lo que es lo mismo, la aceptación de un protocolo de comunicación entre el servidor y un cliente, por ejemplo, mediante certificados digitales.

Entre las diferencias de seguridad entre un Software Libre y el Software Propietario, podemos destacar:

- Seguridad en el Software Propietario: En el caso de tener "agujeros de seguridad", puede que no nos demos cuenta y que no podamos repararlos. Existe una dependencia del fabricante, retrasándose así cualquier reparación y la falsa creencia de que es más seguro por ser oscuro (la seguridad por oscuridad determina los fallos de seguridad no parcheados en cada producto).

- Seguridad en el Software Libre: Por su carácter público y su crecimiento progresivo, se van añadiendo funciones y se nos permite detectar más fácilmente los agujeros de seguridad para poder corregirlos. Los problemas tardan mucho menos en ser resueltos por el apoyo que tiene de los Hackers y una gran comunidad de desarrolladores y al ser un Software de código libre, cualquier empresa puede aportar soporte técnico.

Sin embargo esta es una pregunta sobre la que los expertos al día de hoy, tras muchos años de discusiones, siguen sin ponerse de acuerdo. ¿Es más seguro el Software de código abierto que los programas cerrados, o viceversa? Lo cierto es que, en términos generales, ambos bandos tienen sus razones con las que defender sus argumentos. Por un lado, los usuarios de las aplicaciones y sistemas de código abierto, defienden que, al estar el código fuente disponible a los ojos de todo el mundo, es mucho más fácil localizar posibles agujeros de seguridad y vulnerabilidades que pongan en peligro los datos de los usuarios.

Por otro lado, aquellos que consideran que los sistemas cerrados son más seguros en este sentido, afirman que al tener acceso tan solo los expertos al código fuente de sus aplicaciones, es más complicado que se produzcan

filtraciones o inserciones de Software malicioso en este tipo de sistemas. Hay que tener en cuenta que, por ejemplo, Google premia a las personas que descubren fallos de seguridad en su Software como Chrome, aunque no es el único gigante de la tecnología en utilizar estas tácticas.

De hecho muchas empresas están gastando miles de millones de dólares y/o euros en hacer que sus propuestas sean lo más seguras posible, argumentando que la seguridad de sus proyectos es una de sus prioridades, todo con el fin de intentar frenar que los atacantes vulneren sus sistemas. Por otro lado, otros aseguran que cuando el código fuente es público, más ojos están disponibles para detectar posibles vulnerabilidades o errores en dicho código, por lo que siempre será más rápido y sencillo poner soluciones con el fin de ganar en seguridad.

Sea como sea, en cualquiera de los dos casos, lo que ha quedado más que demostrado es que la seguridad no está garantizada en ningún momento, ya sean propuestas de código abierto, o no. Pero también es cierto que lo que se procura es que los riesgos de ser atacados se reduzcan en medida de lo posible. Los sistemas Linux son considerados desde hace mucho tiempo como un sistema operativo seguro, en buena parte debido a las ventajas que ofrece su diseño. Dado que su código está abierto, son muchas las personas que incorporan mejoras de las que el resto de usuarios de Linux se benefician, a diferencia de las propuestas de Windows o MacOS, donde estas correcciones generalmente se limitan a las que detectan Microsoft y Apple.

No obstante, en nuestra defensa del Software libre, diremos que su código abierto permite que los errores sean encontrados y solucionados con mayor rapidez, por lo que determinamos que es el Software más recomendable.

En general, puede afirmarse que el Software libre es más seguro, ya que debido a su carácter abierto y distribuido, un gran número de programadores y personas expertas pueden estar atentas al código fuente -especialmente en los grandes proyectos-, lo cual permite hacer auditorías con objeto de detectar errores y puertas traseras (Backdoor, en inglés) que pongan en riesgo nuestros datos.

Así, los grandes programas y proyectos de Software libre, con una extensa comunidad de desarrollo y usuarios que lo respalden, presentan niveles muy altos de seguridad, un alto grado de protección y una rápida respuesta a posibles vulnerabilidades.

12.4 Tipos de Licencias

Tanto la Open Source Initiative como la Free Software Foundation mantienen en sus páginas Web (véase [16], [17], y [15]) listados oficiales de las licencias de Software libre que aprueban.

Una licencia es aquella autorización formal con carácter contractual que un autor de un Software da a un interesado para ejercer "actos de explotación legales". Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del Software libre, existen distintas variantes del concepto o grupos de licencias:

Licencias GPL Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autoría (Copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del Software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL, el conjunto tiene que ser GPL.

En la práctica, esto hace que las licencias de Software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

GPL Versión 1 la versión 1 de GNU GPL, fue presentada el 25 de febrero de 1989, impidió lo que eran las dos principales formas con las que los distribuidores de Software restringían las libertades definidas por el Software libre. El primer problema fue que los distribuidores publicaban únicamente los archivos binarios, funcionales y ejecutables, pero no entendibles o modificables por humanos. Para prevenir esto, la GPLv1 estableció que cualquier proveedor de Software libre además de distribuir el archivo binario debía liberar a su vez código fuente entendible y que pudiera ser modificado por el ser humano bajo la misma licencia (secciones 3a y 3b de la licencia).

El segundo problema era que los distribuidores podían añadir restricciones adicionales, añadiendo restricciones a la licencia o mediante la combinación del Software con otro que tuviera otras restricciones en su distribución. Si

esto se hacía, entonces la unión de los dos conjuntos de restricciones sería aplicada al trabajo combinado, entonces podrían añadirse restricciones inaceptables. Para prevenir esto, GPLv1 obligaba a que las versiones modificadas en su conjunto, tuvieran que ser distribuidas bajo los términos GPLv1 (secciones 2b y 4 de la licencia). Por lo tanto, el Software distribuido bajo GPLv1 puede ser combinado con Software bajo términos más permisivos y no con Software con licencias más restrictivas, lo que entraría en conflicto con el requisito de que todo Software tiene que ser distribuido bajo los términos de la GPLv1.

GPL Versión 2 según Richard Stallman, el mayor cambio en GPLv2 fue la cláusula "Liberty or Death" («libertad o muerte»). Esta sección dice que si alguien impone restricciones que le prohíben distribuir código GPL de tal forma que influya en las libertades de los usuarios (por ejemplo, si una ley impone que esa persona únicamente pueda distribuir el Software en binario), esa persona no puede distribuir Software GPL. La esperanza es que esto hará que sea menos tentador para las empresas el recurrir a las amenazas de patentes para exigir una remuneración de los desarrolladores de Software libre.

En 1991 se hizo evidente que una licencia menos restrictiva sería estratégicamente útil para la biblioteca C y para las bibliotecas de Software que esencialmente hacían el trabajo que llevaban a cabo otras bibliotecas comerciales ya existentes. Cuando la versión 2 de GPL fue liberada en junio de 1991, una segunda licencia Library General Public License fue introducida al mismo tiempo y numerada con la versión 2 para denotar que ambas son complementarias. Los números de versiones divergieron en 1999 cuando la versión 2.1 de LGPL fue liberada, esta fue renombrada como GNU Lesser General Public License para reflejar su lugar en esta filosofía.

GPL Versión 3 A finales de 2005, la Free Software Foundation (FSF) anunció estar trabajando en la versión 3 de la GPL (GPLv3). El 16 de enero de 2006, el primer borrador de GPLv3 fue publicado y se inició la consulta pública. La consulta pública se planeó originalmente para durar de nueve a quince meses, pero finalmente se extendió a dieciocho meses, durante los cuales se publicaron cuatro borradores. La GPLv3 oficial fue liberada por la FSF el 29 de junio de 2007.

Según Stallman los cambios más importantes se produjeron en el campo

de las patentes de Software, la compatibilidad de licencias de Software libre, la definición de código fuente y restricciones a las modificaciones de Hardware. Otros cambios están relacionados con la internacionalización, cómo son manejadas las violaciones de licencias y cómo los permisos adicionales pueden ser concedidos por el titular de los derechos de autor. También añade disposiciones para quitar al DRM su valor legal, por lo que es posible romper el DRM (Digital Rights Management) en el Software de GPL sin romper leyes como la DMCA (Digital Millennium Copyright Act).

GPLv2 vs GPL v3 GPLv3 contiene la intención básica de GPLv2 y es una licencia de código abierto con un Copyleft estricto. Sin embargo, el idioma del texto de la licencia fue fuertemente modificado y es mucho más completo en respuesta a los cambios técnicos, legales y al intercambio internacional de licencias.

La nueva versión de la licencia contiene una serie de cláusulas que abordan preguntas que no fueron o fueron cubiertas de manera insuficiente en la versión 2 de la GPL. Las nuevas regulaciones más importantes son las siguientes:

- GPLv3 contiene normas de compatibilidad que hacen que sea más fácil combinar el código GPL con el código que se publicó bajo diferentes licencias. Esto se refiere en particular al código bajo la licencia de Apache v. 2.0.

- Se insertaron normas sobre gestión de derechos digitales para evitar que el Software GPL se modifique a voluntad, ya que los usuarios recurrieron a las disposiciones legales para protegerse mediante medidas técnicas de protección (como la DMCA o la directiva sobre derechos de autor).

- La licencia GPLv3 contiene una licencia de patente explícita, según la cual las personas que licencian un programa bajo licencia GPL otorgan derechos de autor y patentes, en la medida en que esto sea necesario para utilizar el código que ellos otorgan. Por lo tanto, no se concede una licencia de patente completa. Además, la nueva cláusula de patente intenta proteger al usuario de las consecuencias de los acuerdos entre los titulares de patentes y los licenciatarios de la licencia pública general que solo benefician a algunos de los licenciatarios (correspondientes al

acuerdo Microsoft / Novell). Los licenciatarios deben garantizar que todos los usuarios disfrutan de tales ventajas (licencia de patente o liberación de reclamos) o que nadie puede beneficiarse de ellos.

- A diferencia de la GPLv2, la GPLv3 establece claramente que no es necesario divulgar el código fuente en un uso ASP (Application Service Provider) de los programas GPL, siempre que no se envíe una copia del Software al cliente. Si el efecto Copyleft debe extenderse al uso de ASP, debe aplicarse la Licencia pública general de Affero, versión 3 (AGPL) que solo difiere de la GPLv3 en esta consideración.

Licencias Estilo BSD Llamadas así porque se utilizan en gran cantidad de Software distribuido junto a los sistemas operativos BSD. El autor, bajo tales licencias, mantiene la protección de Copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles. Puede argumentarse que esta licencia asegura "verdadero" Software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al Software y que puede decidir incluso redistribuirlo como no libre.

Licencia Copyleft Hay que hacer constar que el titular de los derechos de autor (Copyright) de un Software bajo licencia Copyleft puede también realizar una versión modificada bajo su Copyright original y venderla bajo cualquier licencia que desee, además de distribuir la versión original como Software libre. Esta técnica ha sido usada como un modelo de negocio por una serie de empresas que realizan Software libre (por ejemplo MySQL); esta práctica no restringe ninguno de los derechos otorgados a los usuarios de la versión Copyleft.

Licencia estilo MIT Las licencias MIT son de las más permisivas, casi se consideran Software de dominio público. Lo único que requieren es incluir la licencia MIT para indicar que el Software incluye código con licencia MIT.

Licencia Apache License La licencia Apache trata de preservar los derechos de autor, incluir la licencia en el Software distribuido y una lista de

los cambios realizados. En modificaciones extensivas del Software original permite licenciar el Software bajo otra licencia sin incluir esas modificaciones en el código fuente.

Licencia Mozilla Public License MPL Esta licencia requiere que los archivos al ser distribuidos conserven la misma licencia original pero pueden ser usados junto con archivos con otra licencia, al contrario de la licencia GPL que requiere que todo el código usado junto con código GPL sea licenciado como código GPL. También en caso de hacer modificaciones extensivas permite distribuirlas bajo diferentes términos y sin incluir el código fuente en las modificaciones.

Licencia Código de Dominio Público Es un código que no está sujeto a derechos de autor que puede utilizarse sin restricciones.

Licencia Creative Commons Las licencias de Creative Commons son más utilizadas para cualquier creación digital que para el Software, entendiendo como creación digital desde fotos, artículos en blogs, música, vídeos, este trabajo, etc. Hay varios tipos de licencias de Creative Commons diferenciando entre permitir modificaciones a la obra original, solicitando crédito de la creación o permitiendo un uso comercial de la obra.

Licencias de Código Abierto Las licencias de código abierto son un intermedio entre las licencias privativas y las licencias de Software libre. Las licencias de código abierto permiten el acceso al código fuente pero no todas se consideran licencias de Software libre al no otorgar otros derechos que se requieren para considerar un Software como Software libre como el derecho al uso o con cualquier propósito, modificación y distribución.

Dado el éxito del Software libre como modelo de desarrollo de Software algunas empresas cuyo Software era privativo pueden decidir hacerlo de código abierto con la intención de suplir algunas carencias de Software privativo pero sin perder ciertos derechos que son la fuente de sus ingresos como la venta de licencias.

Las expresiones «Software libre» y «código abierto» se refieren casi al mismo conjunto de programas. No obstante, dicen cosas muy diferentes acerca de dichos programas, basándose en valores diferentes. El movimiento del Software libre defiende la libertad de los usuarios de ordenadores, en un

movimiento en pro de la libertad y la justicia. Por contra, la idea del código abierto valora principalmente las ventajas prácticas y no defiende principios. Esta es la razón por la que gran parte de la comunidad de Software libre está en desacuerdo con el movimiento del código abierto y nosotros no empleamos esta expresión en este texto.

Licencia Microsoft Public License La Microsoft Public License es una licencia de código abierto que permite la distribución del Software bajo la misma licencia y la modificación para un uso privado. Tiene restricciones en cuanto a las marcas registradas.

En caso de distribuir el Software de forma compilada o en forma de objeto binario no se exige proporcionar los derechos de acceso al código fuente del Software compilado o en forma de objeto binario. En este caso esta licencia no otorga más derechos de los que se reciben, pero si permite otorgar menos derechos al distribuir el Software (compilado o en forma de objeto binario).

Modelo de Desarrollo de Software Bazar y Catedral El tipo de licencia no determina qué Software es mejor o peor, si el privativo o el Software libre, la diferencia entre las licencias está en sus características éticas y legales. Aunque el modelo de desarrollo con una licencia de código abierto a la larga suele tener un mejor desarrollo y éxito que el Software privativo, más aún con un medio como internet que permite colaborar a cualquier persona independiente de donde esté ubicada en el mundo.

Comparación con el Software de Código Abierto Aunque en la práctica el Software de código abierto y el Software libre comparten muchas de sus licencias, la Free Software Foundation opina que el movimiento del Software de código abierto es filosóficamente diferente del movimiento del Software libre. Los defensores del término «código abierto (Open Source)», afirman que éste evita la ambigüedad del término en ese idioma que es «Free» en «Free Software».

Mucha gente reconoce el beneficio cualitativo del proceso de desarrollo de Software cuando los desarrolladores pueden usar, modificar y redistribuir el código fuente de un programa. El movimiento del Software libre hace especial énfasis en los aspectos morales o éticos del Software, viendo la excelencia técnica como un producto secundario de su estándar ético. El movimiento de código abierto ve la excelencia técnica como el objetivo prioritario, siendo

el compartir el código fuente un medio para dicho fin. Por dicho motivo, la FSF se distancia tanto del movimiento de código abierto como del término «código abierto (Open Source)».

Puesto que la «iniciativa de Software libre Open Source Initiative (OSI)» sólo aprueba las licencias que se ajustan a la «definición de código abierto (Open Source Definition)», la mayoría de la gente lo interpreta como un esquema de distribución, e intercambia libremente "código abierto" con "Software libre". Aún cuando existen importantes diferencias filosóficas entre ambos términos, especialmente en términos de las motivaciones para el desarrollo y el uso de tal Software, raramente suelen tener impacto en el proceso de colaboración.

Aunque el término "código abierto" elimina la ambigüedad de libertad frente a precio (en el caso del inglés), introduce una nueva: entre los programas que se ajustan a la definición de código abierto, que dan a los usuarios la libertad de mejorarlos y los programas que simplemente tienen el código fuente disponible, posiblemente con fuertes restricciones sobre el uso de dicho código fuente. Mucha gente cree que cualquier Software que tenga el código fuente disponible es de código abierto, puesto que lo pueden manipular, sin embargo, mucho de este Software no da a sus usuarios la libertad de distribuir sus modificaciones, restringe el uso comercial, o en general restringe los derechos de los usuarios.

12.4.1 Licencias Creative Commons

Las Licencias[117] Creative Commons (CC) de forma general no tienen una definición oficial, sin embargo, entre las muchas definiciones aceptadas están la de la UNESCO, la cual expresa la siguiente descripción:

Las Licencias Creative Commons (CC) son modelos de contratos que sirven para otorgar públicamente el derecho de utilizar una publicación protegida por los derechos de autor. Entre menos restricciones implique una licencia, mayores serán las posibilidades de utilizar y distribuir un contenido. Las Licencias CC permiten a cualquier usuario descargar, copiar, distribuir, traducir, reutilizar, adaptar y desarrollar su contenido sin costo alguno.

Sin embargo, en la Web oficial de la Organización Creative Commons se nos dice sobre las mismas lo siguiente:

[117] Las licencias de Creative Commons son más utilizadas para cualquier creación digital que para el Software, entendiendo como creación digital desde fotos, artículos en blogs, música, vídeos, este trabajo, etc.

Las Licencias Creative Commons (CC) brindan a todos, desde creadores individuales hasta grandes instituciones, una forma estandarizada de otorgar permiso al público para usar su trabajo creativo bajo la ley de derechos de autor. Desde la perspectiva del reutilizador, la presencia de una licencia Creative Commons sobre una obra protegida por derechos de autor responde a la pregunta: ¿Qué puedo hacer con esta obra?.

Las «Licencias Creative Commons» que hoy en día pertenecen a la organización mundial Creative Commons[118], y buscan regularizar y mantener, de forma equilibrada y satisfactoria, todo lo relacionado con el derecho de utilizar una publicación protegida por los derechos de autor a nivel mundial, han logrado un buen trabajo, sin duda alguna. Y seguramente en el tiempo, se irán adaptando a las nuevas realidades sociales y tecnológicas para poder seguir manteniendo de forma armónica las posibilidades de utilizar y distribuir cualquier contenido libre y abierto sobre la Internet, y más allá.

¿Cuáles son y cómo funcionan o para qué se usan? Las 7 distintas Licencias Creative Commons son las siguientes:

CC BY Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, siempre que se otorgue la atribución al creador. La licencia permite el uso comercial.

CC BY-SA Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, siempre que se otorgue la atribución al creador. La licencia permite el uso comercial. Si remezcla, adapta o construye sobre el material, debe licenciar el material modificado bajo términos idénticos.

CC BY-NC Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador.

[118] Una organización mundial sin ánimo de lucro que permite compartir y reutilizar la creatividad y el conocimiento mediante el suministro de herramientas legales gratuitas. Y cuyas herramientas legales (licencias) ayudan a quienes quieren fomentar la reutilización de sus obras ofreciéndolas para su uso bajo términos generosos y estandarizados; a quienes quieren hacer usos creativos de las obras; y a quienes quieren beneficiarse de esta simbiosis.

CC BY-NC-SA Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador. Si remezcla, adapta o construye sobre el material, debe licenciar el material modificado bajo términos idénticos.

CC BY-ND Esta licencia permite a los reutilizadores copiar y distribuir el material en cualquier medio o formato únicamente en forma no adaptada, y siempre y cuando se otorgue la atribución al creador. La licencia permite el uso comercial.

CC BY-NC-ND Esta licencia permite a los reutilizadores copiar y distribuir el material en cualquier medio o formato únicamente en forma no adaptada, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador.

CC0 (CC Cero) Esta licencia es una herramienta de dedicación pública que permite a los creadores renunciar a sus derechos de autor y poner sus obras en el dominio público mundial. CC0 permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, sin condiciones.

12.4.2 Nuevas Licencias para Responder a Nuevas Necesidades

El mundo de la tecnología avanza mucho más rápido que las leyes y estas tienen que esforzarse para alcanzarlo. En el caso del Software libre y de código abierto, tanto la Free Software Foundation como la Open Source Initiative (los organismos encargados de regular las diferentes licencias) enfrentan periódicamente el problema de cómo mantener sus principios y al mismo tiempo evitar que alguien se aproveche indebidamente.

En el último tiempo, la Open Source Initiative le dio el sello de aprobación a otras nuevas licencias para propósitos específicos.

Nuevas Licencias de Código Abierto

- Cryptographic Autonomy License version 1.0 (CAL-1.0)

Fue creada en el 2019 por el equipo del proyecto de código abierto Holochain, esta licencia fue desarrollada para ser utilizada con aplicaciones criptográficas distribuidas. El inconveniente con las licencias tradicionales es que no obligaba a compartir los datos. Esto podría perjudicar el funcionamiento de toda la red. Por eso la CAL también incluye la obligación de proporcionar a terceros los permisos y materiales necesarios para utilizar y modificar el Software de forma independiente sin que ese tercero tenga una pérdida de datos o capacidad.

- Open Hardware Licence (OHL)

De la mano de la Organización Europea para la Investigación Nuclear (CERN) llegó esta licencia con tres variantes enfocadas en la posibilidad de compartir libremente tanto Hardware como Software.

Hay que hacer una aclaración. La OSI fue creada en principio pensando en el Software por lo que no tiene mecanismos para la aprobación de licencias de Hardware. Pero, como la propuesta del CERN se refiere a ambos rubros, esto posibilitó la aprobación:

- CERN-OHL-S es una licencia fuertemente recíproca: El que utilice un diseño bajo esta licencia deberá poner a disposición las fuentes de sus modificaciones y agregados bajo la misma licencia.

- CERN-OHL-W es una licencia débilmente recíproca: Sólo obliga a distribuir las fuentes de la parte del diseño que fue puesta originalmente bajo ella. No así los agregados y modificaciones.

- CERN-OHL-P es una licencia permisiva: Permite a la gente tomar un proyecto, relicenciarlo y utilizarlo sin ninguna obligación de distribuir las fuentes.

Hay que decir que la gente del CERN parece haber encontrado la solución a un problema que viene afectando a algunos proyectos de código abierto. Una gran empresa utiliza ese proyecto para comercializar servicios y no solo hace ningún aporte al proyecto original (ya sea con código o dando apoyo financiero) si no que también compite en el mismo mercado.

Post Open Zero-Cost en el año 2024 se desarrolla la licencia «Post Open Zero-Cost» con la cual busca abordar los desafíos surgidos en la interacción entre desarrolladores de código abierto y empresas comerciales, especialmente en lo que respecta a la compensación justa por el uso comercial del código.

La característica distintiva de la licencia «Post-Open» en comparación con las licencias abiertas existentes, como la GPL, es la introducción de un componente contractual que puede ser rescindido en caso de violación de los términos. Esta licencia ofrece dos tipos de acuerdos contractuales: gratuitos y de pago. El acuerdo de pago permite negociar derechos adicionales para la distribución comercial de productos o modificaciones sin requerir su divulgación pública.

Las situaciones que podrían llevar a la terminación del acuerdo contractual incluyen: violación de los términos de la licencia; reclamaciones por infracción de patentes; imposición de condiciones adicionales (como sanciones en contratos con clientes por divulgación de información sensible); cambios sujetos a leyes de control de exportaciones; ocultamiento de información sobre vulnerabilidades; y uso del código para entrenamiento de modelos de aprendizaje automático bajo términos no permitidos. Las relaciones contractuales no se rescinden de inmediato, sino que se notifica la infracción y se otorgan 60 días para corregirla antes de la rescisión efectiva del acuerdo.

Uno de los problemas que la nueva licencia busca abordar está relacionado con las limitaciones de la GPL, la cual se centra en otorgar derechos sin la capacidad de revocarlos, lo que permite a las empresas encontrar formas de eludir sus requisitos, especialmente en lo que respecta al acceso al código fuente. Estas lagunas son utilizadas para restringir la disponibilidad del código subyacente en productos comerciales mediante la imposición de términos contractuales adicionales con los usuarios finales.

Un claro ejemplo es el de RHEL, la cual los clientes firman un acuerdo con Red Hat que limita la redistribución del código fuente al imponer condiciones sobre la coincidencia de las copias instaladas y compradas de RHEL. Esto coloca a los usuarios en la disyuntiva entre su libertad para disponer del software y mantener su estatus de cliente de Red Hat. Aunque la GPL permite la distribución de parches que solucionan vulnerabilidades en el código de RHEL, esto podría interpretarse como una violación del acuerdo con Red Hat y podría resultar en la terminación de los servicios por parte de la empresa.

12.5 Implicaciones Económico-Políticas del Software Libre

Una vez que un producto de Software libre ha empezado a circular, rápidamente está disponible a un costo muy bajo. Al mismo tiempo, su utilidad no decrece. El Software, en general, podría ser considerado un bien de uso inagotable, tomando en cuenta que su costo marginal es pequeñísimo y que no es un bien sujeto a rivalidad -la posesión del bien por un agente económico no impide que otro lo posea-.

12.5.1 Software Libre y la Piratería

Puesto que el Software libre permite el libre uso, modificación y redistribución, a menudo encuentra un hogar entre usuarios para los cuales el coste del Software no libre es a veces prohibitivo, o como alternativa a la piratería. También es sencillo modificarlo localmente, lo que permite que sean posibles los esfuerzos de traducción a idiomas que no son necesariamente rentables comercialmente, además:

- Porque así no se condiciona a los usuarios a usar siempre lo mismo.
- Porque así no se fomenta la piratería en los usuarios al no pagar licencias.
- Porque así no se obliga a usar una solución concreta y se ofrece libertad de elección a los usuarios.
- Porque es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.

La mayoría del Software libre se produce por equipos internacionales que cooperan a través de la libre asociación. Los equipos están típicamente compuestos por individuos con una amplia variedad de motivaciones y pueden provenir tanto del sector privado, del sector voluntario o del sector público. En los últimos años se ha visto un incremento notable de grandes corporativos (como IBM, Microsoft, Intel, Google, Samsung, Red Hat, etc.) que han dedicado una creciente cantidad de recursos humanos y computacionales para desarrollar Software libre, ya que esto apoya a sus propios negocios.

12.5.2 ¿Cuánto Cuesta el Software Libre?

En esta sección intentaremos dar una idea de cuál es el costo del desarrollo del Software Libre, por supuesto que no se tratará más que de una conjetura aproximada basada en las cifras proporcionadas por desarrolladores de Software comercial (al año 2020).

Gratis no Significa Gratuito Supongamos que todos los recursos humanos participantes en el desarrollo de un proyecto de Software libre lo hagan de forma voluntaria. De todas formas tenemos lo que los contables llaman «Costo de oportunidad» esto es, los ingresos que podrían haber generado esas personas si hubieran dedicado el tiempo y los conocimientos invertidos en el proyecto a uno en el que les pagaran. Así, el calcular el costo promedio por hora que cobra un programador, por la cantidad de horas invertidas al proyecto, nos da un razonable costo mínimo. Lo mismo puede hacerse con los voluntarios dedicados a la difusión en las redes. El costo de una campaña de marketing digital puede estimarse fácilmente.

Muchos proyectos de código abierto como una distribución Linux, son construidos a partir de la integración de otros proyectos, por los que sus costos de desarrollo también deberían sumarse.

Por otra parte, necesitamos recursos físicos. Aún cuando los voluntarios trabajen desde su casa, siguen teniendo que comprar y mantener sus equipos, además de pagar la electricidad que los hace funcionar.

Bases para el Cálculo Hay muchos factores que determinan el costo de desarrollar una pieza de Software. En un extremo tenemos una aplicación simple que requiere muy poca interacción del usuario o procesamiento del lado del servidor. Tal es el caso de un cliente de escritorio para redes sociales. Por el otro sistemas operativos que deben operar en múltiples plataformas realizando múltiples tareas (por ejemplo Debían que aspira a ser el sistema operativo universal). Sin embargo, el costo de una aplicación simple puede elevarse debido a que tiene múltiples pantallas diferentes. Por ejemplo un juego desarrollado con HTML5 y Javascript.

Los dos aspectos claves son la cantidad de horas de trabajo necesarias y las tecnologías involucradas. Para una aplicación de escritorio como un procesador de textos con las prestaciones habituales, optimizado para un determinado escritorio Linux, se estima que se tendría que contar con al menos el equivalente a 42,000 euros en trabajo voluntario. Un gestor de contenidos

para comercio electrónico con seguimiento de pedidos e integración con las principales plataformas de pago implicaría desembolsar unos 210,000 euros o su equivalente en trabajo voluntario.

Tomando en cuenta que este cálculo incluye lo que costó el desarrollo de las bibliotecas y otros proyectos libres y de código abierto incluidos, pero no los gastos que efectivamente deben desembolsarse en efectivo como la compra de equipos, Software de seguridad y desarrollo y el pago de electricidad e internet.

Por otro lado, el proceso de medición de costes del Software es un factor realmente importante en el análisis de un proyecto. Hay distintos métodos de estimación de costes de desarrollo de Software (también conocido como métrica del Software). La gran mayoría de estos métodos se basan en la medición del número de Líneas de Código que contiene el desarrollo (se excluyen comentarios y líneas en blanco de los fuentes).

Desarrollo de Fedora 9 La Linux Foundation ha calculado que costaría desarrollar el código de la distribución Fedora 9 que fue puesta a disposición del público el 13 de mayo de 2008, en el informe citado "Estimating the Total Development Cost of a Linux Distribution" se calcula que Fedora 9 tiene un valor de 10.8 mil millones de dólares y que el coste únicamente del Kernel (2.6.25 con 8,396,250 líneas de código) tendría un valor de 1.4 mil millones de dólares.

Esta distribución tiene unas 205 millones de líneas de código, el proyecto debería ser desarrollado por 1000 - 5000 desarrolladores (el trabajo invertido por una única persona desarrollándolo se alargaría durante unos 60.000 años) y esa estimación no va muy desencaminada ya que en los 2 últimos años del desarrollo de esa versión contribuyeron unos 3,200 desarrolladores aunque el número de trabajadores en la historia de la distribución es mucho mayor.

¿Qué pasa con GNU/Linux? en el año 2015 (las estadísticas más actuales que conseguimos) la Linux Foundation analizó el costo de desarrollo del núcleo. Combinando el aporte de los recursos humanos (voluntarios y de pago) y los desembolsos necesarios, la cuenta sumó 476,767,860,000.13 euros.

Todos sabemos que el hecho de tener desarrolladores asalariados no garantiza necesariamente Software de calidad. Pero, tener desarrolladores que pueden dedicar toda su atención a un proyecto en lugar de hacerlo en sus horas libres si lo hace. Lamentablemente, por el momento el único modo de

lograr eso es obtener el apoyo de corporaciones (Intel, Google, IBM, AMD, Sun Microsystems, Dell, Lenovo, Asus, HP, SGI, Oracle, RedHat, etc.) que solo lo hacen con los proyectos que son de su interés como el Kernel de Linux, hay que notar que para el Kernel de Linux un porcentaje importante (más del 10 %) lo hacen programadores independientes.

Costes Recordemos que la segunda de las cuatro libertades de un programa para ser Software libre es:

- Libre redistribución

y esta puede ser a través de un pago o sin costo. Es por ello que existen distintas empresas, organizaciones y usuarios que pueden apoyar a los usuarios finales en el desarrollo y soporte de algún programa de Software libre o una distribución personalizada de Linux por un costo determinado.

Mucha gente, en especial ejecutivos de empresas, se acercan a Linux bajo la promesa de que es una solución de bajo costo -muchos piensan que incluso es gratis-. Pero la realidad es que detrás de Linux y los programas de Software libre (y aquí la traducción correcta de la palabra inglesa, Free es libre, no gratis) pueden llevar una serie de costos ocultos que deben ser considerados al momento de decidir si se implementa una solución propietaria o una libre.

Los costos ocultos aparecen cuando se intenta instalar y capacitar en el uso de algún Software y se necesita la ayuda de un informático, al que se tiene que pagar, o alguna empresa quiere personalizar la interfaz de un programa y necesita la ayuda de un programador, que también tienen un coste, por lo que finalmente el comentario suele ser "el Software libre no es barato".

El primer punto a considerar al evaluar ambas alternativas es el costo de la licencia. Los productos de Software libre no suelen tener un costo de licencia asociado, que sí existe en los programas propietarios. De hecho es allí en donde los fabricantes de Software recargan sus costos de investigación y desarrollo, de producción e incluso sus ganancias. En este primer punto el ganador claro es el Software libre y es lo que los adeptos de este esquema publicitan: "su compañía puede ahorrar miles de dólares al año usando Software libre".

El segundo punto a considerar es el costo de instalación, configuración y capacitación. Dependiendo de su complejidad, algunos productos comerciales no contemplan costos extra por este concepto y otros -como Windows, por ejemplo- son tan populares que se puede encontrar numerosas opciones

de instalación -a través de empresas o profesionales- donde escoger en el mercado. A veces en el Software libre la configuración puede implicar recompilar el producto con algunas opciones particulares, algo que sólo pueden realizar técnicos con un nivel adecuado de conocimientos y que puede que no sean tan fáciles de encontrar. Aquí por lo general la ventaja va hacia el Software comercial.

Una vez instalado el Software, toca realizar actualizaciones de mantenimiento. Si bien es cierto lo que algunos de los fanáticos del Software libre dicen, que nadie lo obliga a mejorar el Software con que cuenta, la realidad -especialmente en lo que a seguridad se refiere- obliga a las empresas a mantener su Software actualizado. Aún así, los costos de actualizar Software libre suelen ser significativamente más bajos que los de productos comerciales y suelen ser menos exigentes con el Hardware necesario para ejecutarlos. La mayoría de las veces la ventaja es para el Software libre, pero hay que evaluar ya que varía dependiendo de cada caso.

Por último hay algunas casas que desarrollan productos de Software libre -dan el código y permiten que cualquiera lo modifique o reutilice- pero fijan contratos con cargos mensuales o anuales para mantenimiento del mismo, algo que se parece mucho a un cobro por licencia, por lo que hay que estar seguro de conocer todas las condiciones cuando una empresa nos ofrece una solución propia y la califica como Software libre.

Además de estas consideraciones hay una que debe sumarse eventualmente a esta evaluación: el costo de migrar a otra solución. En Software libre suelen usarse estándares abiertos para almacenar los datos, lo que facilita las migraciones. En cambio muchas soluciones propietarias suelen tener formatos propietarios que pueden dejar "amarrados" los datos de la empresa a una aplicación específica.

Sólo después de evaluar estos aspectos del Software, que pueden tener implicaciones importantes en el presupuesto, es que un CIO (Chief Information Officer) puede decir si una solución de Software libre le conviene más a una empresa o no, algo que va más allá de que la aplicación sea gratis o no.

12.5.3 La Nube y el Código Abierto

Desde hace años se han creado nuevos desafíos para el código abierto que plantea la nube, un término que para el usuario promedio puede significar cosas diferentes, pero que para la empresa se resume en servicios. Y es que los beneficios económicos que genera el mero Software de código abierto no

son comparables a los que se obtienen cuando se ofrece ese mismo Software a través de servicios, más allá -pero incluyendo- del soporte.

Este hecho diferencial lleva tiempo provocando fricciones entre desarrolladores y proveedores y hay quien adelantó incluso el fin del modelo de desarrollo del código abierto tal y como lo conocemos. ¿Quién tiene la razón?, ¿es para tanto la situación?.

En sus exposiciones, representantes de compañías y proyectos de código abierto muy populares en el ámbito empresarial, explican el supuesto perjuicio que les ocasiona el uso que los grandes proveedores de servicios en la nube hacen del Software que ellos desarrollan y cómo algunos han considerado y aplicado un enfoque más cerrado para sus productos con el fin de evitar lo que denominan como expolio. Hay declaraciones que merecen ser rescatadas para dotar de contexto a la discusión:

- El papel que juega el código abierto en la creación de oportunidades comerciales ha cambiado, durante muchos años les permitimos que las empresas de servicios tomaran lo que se ha desarrollado y ganasen dinero con ello.

- Empresas como Amazon Web Services, Azure de Microsoft, etc. Han ganado cientos de millones de dólares ofreciendo a sus clientes servicios basados en Software libre sin contribuir tanto a la comunidad de código abierto que construye y mantiene ese proyecto. Es imposible saber exactamente de cuánto dinero estamos hablando, pero es cierto que los proveedores de la nube se benefician del trabajo de los desarrolladores de código abierto que no emplean.

- Hay un mito ampliamente instalado en el mundo de código abierto que dice que los proyectos son impulsados por una comunidad de contribuyentes, pero en realidad, los desarrolladores pagados contribuyen con la mayor parte del código en la mayoría de los proyectos de código abierto modernos.

En resumen, todas estas voces se quejan de dos cosas: los grandes beneficios que obtienen los proveedores de servicios en la nube con su Software sin retribuirles en consecuencia, y la falta de colaboración manteniendo los productos con los que lucran. Sin embargo, no nos engañemos, el quid de la cuestión está principalmente en el dinero: la opinión generalizada de la

comunidad es que el Software de código abierto nunca fue pensado para que las empresas de servicios en la nube lo tomasen y lo vendieran.

Por otro lado, si es posible bifurcar un proyecto libre que se cierra, ¿no hubiese sido mejor colaborar con él antes y haber evitado el cierre? Si no se invierte y se mantiene con salud aquello que da beneficios, puede terminar por desaparecer. A medio camino entre el depredador y el parásito: así es como ven muchas desarrolladoras de código abierto a los proveedores de servicios en la nube.

Vale la pena retomar ahora la frase «el Software de código abierto nunca fue pensado para que las empresas de servicios en la nube lo tomasen y lo vendieran». ¿Para qué fue pensado el código abierto entonces? No hay ninguna licencia de código abierto o Software libre reconocida por la Free Software Foundation o la Open Source Initiative que prohíba hacer negocio con el Software. Lo que prohíben es la discriminación en la capacidad y alcance de su uso en función de la parte, se trate de un individuo o de la mayor multinacional imaginable.

¿Cuál es la solución a un embrollo de tamaña envergadura? Lo único claro es que no es una cuestión de blancos y negros y las consideraciones son demasiadas como para seguir ahondando: empresas que cotizan en bolsa quieren más dinero de otras compañías -que también cotizan en bolsa y por mucho más-, que además del Software ponen la infraestructura sobre la que distribuyen sus ofertas y que tienen la capacidad de clonar tu producto en un abrir y cerrar de ojos, no solo porque tienen el capital, sino porque tienen la experiencia necesaria tras contribuir técnica, pero también en muchos casos, económicamente, durante largo tiempo.

Pese a ello, esta situación está alterando el paradigma actual, en el que el modelo de desarrollo del código abierto se ha impuesto como impulsor de la innovación en el sector empresarial y ya hay quien habla de que nos acercamos al fin, o al principio del fin de la Era del Open Source, cuya preponderancia estaría sentenciada por la revolución de la nube, a la postre el mayor estímulo que haya tenido el código abierto hasta la fecha.

El futuro, pues, pasaría por el Shared Source Software, bajo el cual diferentes compañías con intereses alineados colaborarán en el desarrollo de proyectos concretos, pero limitando su explotación comercial a sí mismas. Todavía no estamos ahí, no obstante, y no parece tampoco que el relevo se vaya a dar en breve. De suceder, será muy llamativo: la muerte del código abierto por un éxito mal entendido.

12.5.4 El Código Abierto como Base de la Competitividad

En septiembre del 2021, se publicó un amplio y detallado informe llevado a cabo por el Fraunhofer ISI y por OpenForum Europe para la Comisión Europea, «The impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy», cuantifica la importancia económica del código abierto aplicado tanto al Software como al Hardware, su efecto en la contribución al producto interior bruto generado, la reducción en aspectos como el coste total de propiedad, dependencia del proveedor y autonomía digital; lanza una serie de recomendaciones específicas de políticas públicas destinadas a lograr un sector público digitalmente autónomo, una investigación y desarrollo abierto que fomente el crecimiento europeo y una industria más digitalizada y competitiva.

En las estimaciones del informe se apunta que las empresas europeas invirtieron alrededor de mil millones de euros en Software de código abierto en 2018, lo que resultó en un impacto en la economía europea de entre 65,000 y 95,000 millones de euros. El análisis estima una relación costo-beneficio superior a 1:4 y predice que un aumento del 10% de las contribuciones de a repositorios de código abierto sería susceptible de generar anualmente entre un 0.4% y un 0.6% adicional en el PIB, así como más de seiscientas nuevas empresas tecnológicas en la Unión Europea.

El análisis de las contribuciones a repositorios de Software de código abierto en la Unión Europea revela que el ecosistema tiene una naturaleza diferente frente al norteamericano, con un volumen de contribuciones que provienen sobre todo de empleados de compañías pequeñas o muy pequeñas, frente a un escenario en los Estados Unidos en el que predominan grandes compañías tecnológicas que se benefician en sus modelos de negocio de la gran cantidad y de la rápida mejora del Software disponible. En Europa, los contribuyentes individuales ascendieron a más de 260,000, lo que representa el 8% de los casi 3.1 millones de empleados de la UE en el sector del desarrollo de Software en 2018. En total, los más de 30 millones de desarrollos consolidados en repositorios en los estados miembros de la Unión Europea representan una inversión de personal equivalente a casi mil millones de euros, que han pasado a estar disponibles en el dominio público y que, por lo tanto, no tienen que ser desarrollados por otros actores.

Según el análisis, cuanto más pequeña es la empresa, mayor es la inversión relativa en Software de código abierto (las empresas con 50 empleados o menos asumieron casi la mitad de los desarrollos en la muestra de las com-

pañías más activas). Aunque más del 50% de los contribuyentes pertenecen a la industria tecnológica (el 8% del total de sus empleados participaron en estos desarrollos), también hubo participación significativa de empresas de consultoría, científicas, técnicas y, en menor medida, de distribuidores, minoristas y empresas del ámbito financiero.

¿Puede una filosofía de desarrollo como el código abierto, disponible para todo el mundo, llegar a convertirse en una fuente de ventajas diferenciales para el resto de los países, que se ha visto tradicionalmente muy superado en su relevancia en el entorno tecnológico por los gigantes tecnológicos de Estados Unidos o de China? El informe afirma que su uso puede llegar a incidir en gran medida en el desarrollo de una independencia tecnológica superior, de una mayor competitividad y de más innovación. Veremos si llegamos a ver en el resto del mundo políticas que incentiven el uso del código abierto como una variable estratégica clave para ello. La idea, capitalizar la tecnología de una forma más orientada al procomún y al desarrollo colaborativo, suena sin duda atractiva e interesante.

12.5.5 Software Libre en Empresas y Corporaciones

En esta sección exploraremos algunas de las claves por las cuales el Software libre está hoy en el punto de mira de todo tipo de empresas y grandes corporaciones (algunas de las cuales ayer eran sus acérrimos enemigos). Pero hay que empezar destacando que corporativos como Google, Amazon Web Services, Azure de Microsoft, Microsoft, IBM, entre otras, en los últimos años han ganado miles de millones de dólares ofreciendo a sus clientes servicios y/o productos basados en Software libre y con una queja recurrente por su pobre o nula contribución a la comunidad de código abierto que construyó y mantiene esos proyectos.

Si bien es imposible saber exactamente de cuánto dinero estamos hablando, pero es cierto que empresas y corporaciones se benefician diariamente del trabajo de los desarrolladores de código abierto que no emplean.

Grandes Equipos de Programadores GNU/Linux ha demostrado que los equipos de desarrolladores grandes, distribuidos, aunque desorganizados pueden crear Software viable.

Antes de la llegada de GNU/Linux, la mayoría del Software era desarrollado por pequeños equipos de programadores que trabajaban en estrecha coordinación entre sí. Ese era el enfoque recomendado por informáticos de hace

unos decenios, que advertían que añadir más programadores a un proyecto tendía a disminuir su eficiencia. Y estaban muy equivocados.

Desde el principio, el Kernel de Linux se desarrolló con un enfoque diferente, en el que programadores de todo el mundo, que en la mayoría de los casos no se conocían, escribieron e integraron el código de forma rápida y poco organizada. Gracias a la publicación temprana y frecuente, consiguieron que funcionara y hoy día es el Kernel más usado en informática en supercomputadoras y dispositivos móviles.

Pero actualmente hay un mito ampliamente instalado en el mundo de código abierto, que dice que los proyectos son impulsados por una comunidad de contribuyentes gratuitos, pero en realidad, los desarrolladores pagados contribuyen con la mayor parte del código en la mayoría de los proyectos de código abierto modernos de los cuales las corporaciones pueden sacar provecho. Claro ejemplo es el propio Kernel de Linux, en el cual una gran cantidad de desarrolladores actuales pertenecen o son subvencionados por empresas, fundaciones o corporaciones (actualmente cientos de ellas), como en el caso de Linus Torvalds que trabaja bajo los auspicios de la Fundación de Linux.

Reutilización de Software Parte de la razón por la que Linux se hizo muy popular entre los ingenieros de Software con relativa rapidez fue que Linux -y el Software libre en general- facilita la reutilización del código escrito por otras personas.

Hoy en día, la reutilización de Software de terceros es habitual, incluso entre los equipos de desarrollo cuyos productos no son de código libre. Es difícil imaginar la construcción de una aplicación hoy en día sin hacer uso de las bibliotecas de Software de origen, las API de terceros u otros recursos externos a su propio proyecto.

Es cierto que proyectos como GNU, que precedió al Kernel de Linux en siete años, promovían la reutilización de código antes de que apareciera el núcleo. Pero, podría decirse que Linux fue el proyecto que trajo las prácticas de codificación libre a la corriente que tanto parece interesar a ciertas grandes empresas, ayudando a crear el modelo de ingeniería de Software de componentes de Software modulares y reutilizables.

Gestión Actual del Código Fuente Linus Torvalds, que creó el núcleo de Linux cuando era estudiante en Helsinki, es el más famoso por ese trabajo.

Pero un hecho a menudo olvidado es que Torvalds es también el padre de Git, el masivamente popular gestor de código fuente libre.

Torvalds creó Git para ayudar a gestionar el código fuente de Linux. Si Linux no existiera, tampoco existiría Git. Tampoco existiría GitHub, ni GitLab, ni GitOps. Y, lo que es más importante, sin la idea de Software libre y colaborativo de Richard Stallman, tampoco existiría la cultura de intercambio y colaboración abierta que sostienen estas tecnologías.

Estrategias de Despliegue de Software "App Store" Apple puede atribuirse el mérito de haber lanzado la primera App Store, un lugar donde los desarrolladores pueden compartir aplicaciones y los usuarios pueden instalarlas fácilmente, utilizando un catálogo Online centralizado.

Pero al igual que con muchas cosas que ha hecho Apple, el concepto de App Store (que ahora es una estrategia de despliegue de Software apilado como servicio, especialmente pero no sólo en el ecosistema móvil) se parece mucho a lo que los desarrolladores de GNU/Linux estaban haciendo a través de los repositorios de Software mucho antes de que las tiendas de aplicaciones se convirtieran en algo común en el mundo del Software propietario, como también lo es la Tienda de Windows.

Los repositorios de Software en GNU/Linux hacen más o menos lo mismo que las tiendas de aplicaciones: Permiten a los usuarios seleccionar las aplicaciones que quieren de una lista centralizada y en línea, y luego instalarlas con unos pocos clics o bien órdenes de terminal (como el famoso *apt* de Debian).

Es cierto que empresas como Apple parecen tener el mérito de crear tiendas de aplicaciones muy fáciles de usar de hacer clic e ir, pero no es un invento de ellos. Y la historia del concepto de tienda de aplicaciones en general implica a más actores que sólo la comunidad de Apple. Aún así, creo que se podría argumentar con fuerza que, sin GNU/Linux y los repositorios de Software de GNU/Linux, las tiendas de aplicaciones tal y como las conocemos hoy no existirían.

Formatos Abiertos para Intercambio de Información Hay una gran variedad de tecnologías disponibles para producir y almacenar datos. Como son: hojas de cálculo, bases de datos, Software estadístico más específico y más. Esto genera una enorme diversidad de formatos, a veces esto es por decir lo menos caótico.

La ventaja de los archivos de formatos abiertos, es que permiten a los de-

sarrolladores producir varios paquetes de Software y servicios utilizando esos formatos. Esto entonces reduce al mínimo los obstáculos para la reutilización de la información que contienen.

El advenimiento del Software libre ha generado algunos de los formatos abiertos más usados para el intercambio de información, pero los entes generadores de información no siempre se adecuan a los niveles de apertura deseados, algunos de estos formatos abiertos son: XML, JSON, YAML, RDF, REBOL, PDF, CSV, ODF, OOXML, TXT, HTML, HDF.

Pero, incluso si la información se proporciona en formato electrónico, formato legible por máquina y en detalle, puede existir problemas relacionados con el formato del archivo en sí (principalmente el generado por los diversos sistemas operativos). Los formatos en los cuales la información es publicada -en otras palabras, la base digital en la cual la información es almacenada- puede ser "abierta" o "cerrada".

Un formato abierto es aquel donde las especificaciones del Software están disponibles para cualquier persona, de forma gratuita, así cualquiera puede usar dichas especificaciones en su propio Software sin ninguna limitación en su reutilización que fuere impuesta por derechos de propiedad intelectual.

Si el formato del archivo es "cerrado", esto puede ser debido a que el formato es propietario y sus especificaciones no están disponibles públicamente, o porque el formato es propietario y aunque las especificaciones se han hecho públicas, su reutilización es limitada. Si la información es liberada en un formato de archivo cerrado, esto puede causar grandes obstáculos para reutilizar la información codificada en él, forzando a aquellos que deseen usar la información a comprar Software innecesario.

El uso de formatos de archivo con propiedad, para el que la especificación no está disponible públicamente, puede crear dependencia de Software de terceros o de los titulares de licencias de los formatos de archivos. En el peor de los casos, esto puede significar que la información sólo se puede leer con cierto Software específico, que puede ser caro, o que puede quedar obsoleto.

La preferencia del término Gobierno de Datos Abiertos, es que la información se publicará en formatos de archivo abiertos, los cuales son de lectura mecánica y esto es una aportación más del Software libre.

Ciencia Abierta La ciencia abierta (Open Science) es el movimiento creciente para hacer que la ciencia sea abierta. La ciencia en sí misma se utilizó como un ejemplo principal de la eficacia del movimiento de código abierto, ci-

tando prácticas como la difusión abierta de información, métodos y revisión por pares de la literatura científica. Podría decirse que la ciencia abierta comenzó en el siglo XVII con el advenimiento de la revista científica y la práctica de repetir los experimentos presentados en los artículos académicos. Estas revistas se imprimirían y distribuirían en todo el mundo, a menudo supervisadas por sociedades científicas como la Royal Society.

¿Qué impulsó la necesidad de un movimiento de ciencia abierta? La Royal Society tenía el famoso lema "Nullius in verba", traducido de forma aproximada como "no tome la palabra de nadie". Esto encarnaba un principio general en la ciencia de que todas las teorías están abiertas a ser cuestionadas y los resultados declarados deben ser repetibles. De hecho, es una práctica generalizada que fue realizada por la sociedad en esos primeros años. En los últimos años esta práctica no ha sido tan común, con más y más ciencia confiando en elementos cerrados, lo que en última instancia conduce a errores que son más difíciles de detectar sin un intercambio completo de información: datos, métodos y publicaciones.

El movimiento de ciencia abierta afirma en términos generales que la ciencia debe realizarse de manera abierta y reproducible donde todos los componentes de la investigación estén abiertos. Muchas revistas permanecen estancadas en un formato en el que se imprimían físicamente, a pesar de que en la actualidad se distribuyen en gran medida en línea. A menudo, todavía utilizan archivos PDF como una forma de "papel electrónico" con publicaciones fijas, procesos cerrados de revisión por pares y poco o ningún acceso a los datos. Este fue sin duda el modo más eficiente de difundir el conocimiento científico antes de los albores de internet, pero ahora un número cada vez mayor lo considera lejos de ser el óptimo.

La ciencia abierta encarna una serie de aspectos, en el núcleo esto incluye acceso abierto, datos abiertos, código abierto y estándares abiertos que ofrecen una diseminación sin restricciones del discurso científico. Estas cosas permiten una ciencia reproducible al brindar acceso completo a los componentes principales de la investigación científica. Hay una serie de componentes adicionales que también se están explorando, como la revisión por pares abierta, donde los revisores de publicaciones científicas publican revisiones abiertamente con su nombre adjunto y la ciencia de libreta abierta donde las libretas (tradicionalmente cerradas) se publican abiertamente en línea a medida que se realiza la investigación.

¿Por qué la ciencia abierta es tan importante en la era digital? También existe una creciente comprensión de que, dado que la investigación científica

depende cada vez más del código informático para simulaciones, cálculos, análisis, visualización y procesamiento de datos en general, es importante tener acceso a este código tal como tradicionalmente ha sido importante mostrar (y derivar) cualquier nueva técnica matemática introducida para el análisis. Hay revistas como PLOS ONE y F1000 que exploran el significado de las publicaciones, ya sea que se deben congelar en el tiempo o se pueden actualizar. Los repositorios de datos también están ganando importancia a medida que las agencias de financiación requieren la publicación y preservación de los datos generados por la investigación financiada.

En esencia, la ciencia abierta se trata de volver a esos valores fundamentales inculcados por algunos de los primeros científicos de que no debemos confiar en la palabra de nadie, que es esencial que todos los elementos pertinentes a un descubrimiento pretendido se publiquen para que los resultados puedan repetirse y validarse. El movimiento de la ciencia abierta varía en el grado en que lo requiere, pero están surgiendo patrones. Se están estableciendo recomendaciones sobre licencias, como CC0 para datos, CC-BY para publicaciones, licencias compatibles con OSI para código fuente y formatos abiertos para datos. En última instancia, se trata de empoderar a todos para que participen en la ciencia, con internet como vehículo principal para la amplia difusión de este conocimiento.

Este movimiento está cambiando la forma en que se hace la ciencia, está recibiendo el respaldo de muchas agencias de financiamiento, ya que requieren planes de gestión de datos, planes de distribución de código fuente y una mayor validación de los resultados a través del acceso abierto a estos resultados para todos. Esto también mejora la transferencia de conocimientos de la academia a la industria, ya que se brinda acceso completo en el momento de la publicación o después de un período de embargo. El movimiento de la ciencia abierta se limita en gran medida a la investigación que está financiada por las agencias de financiación nacionales de todo el mundo y exige que todos los que financian la investigación tengan acceso total e igualitario a ella.

Open Hardware El concepto de Software libre también se permeó al Hardware. El término Open Hardware u Open Source Hardware, se refiere al Hardware cuyo diseño se hace públicamente disponible para que cualquiera pueda estudiarlo, modificarlo y distribuirlo, además de poder producir y vender Hardware basado en ese diseño. Tanto el Hardware como el Software

que lo habilita, siguen la filosofía del Software libre. Hoy en día, el término "hágalo usted mismo" (DIY por sus siglas en inglés) se está popularizando en el Hardware gracias a proyectos como Arduino que es una fuente abierta de prototipos electrónicos, una plataforma basada en Hardware flexible y fácil de utilizar que nació en Italia en el año 2005.

El movimiento de Hardware abierto o libre, busca crear una gran librería accesible para todo el mundo, lo que ayudaría a las compañías a reducir en millones de dólares en trabajos de diseño redundantes. Ya que es más fácil tener una lluvia de ideas propuesta por miles o millones de personas, que por solo una compañía propietaria del Hardware, tratando así de que la gente interesada entienda cómo funciona un dispositivo electrónico, pueda fabricarlo, programarlo y poner en práctica esas ideas en alianza con las empresas fa-bricantes, además se reduciría considerablemente la obsolescencia programada y en consecuencia evitaríamos tanta basura electrónica que contamina el medio ambiente. Al hablar de Open Hardware hay que especificar de qué tipo de Hardware se está hablando, ya que está clasificado en dos tipos:

- Hardware estático. Se refiere al conjunto de elementos materiales de los sistemas electrónicos (tarjetas de circuito impreso, resistencias, capacitores, LEDs, sensores, etcétera).

- Hardware reconfigurable. Es aquél que es descrito mediante un HDL (Hardware Description Language). Se desarrolla de manera similar a como se hace Software. Los diseños son archivos de texto que contienen el código fuente.

Para tener Hardware reconfigurable debemos usar algún lenguaje de programación con licencia GPL (General Public License). La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se le denomina contrato de licencia o acuerdo de licencia. La Organización Europea para la investigación Nuclear (CERN) publicó el 8 de julio de 2011 la versión 1.1 de la Licencia de Hardware Abierto.

Existen programas para diseñar circuitos electrónicos y aprender de la electrónica como EDA (Electronic Design Automation) y GEDA (GPL Electronic Design Automation), son aplicaciones de Software libre que permiten poner en práctica las ideas basadas en electrónica.

Es posible realizar el ciclo completo de diseño de Hardware reconfigurable desde una máquina con GNU/Linux, realizándose la compilación, simulación,

síntesis y descarga en una FPGA (Field Programmable Gate Arrays). Para la compilación y simulación se puede usar GHDL (https://ghdl.free.fr) junto con GTKWave (https://gtkwave.sourceforge.net) y para la síntesis el entorno ISE de Xilinx. Este último es Software comercial pero existe una versión gratuita con algunas restricciones.

Sabemos que tanto en el caso del Software como el Hardware, libre no es lo mismo que gratis. Específicamente, en el caso del Hardware, como estamos hablando de componentes físicos que son fabricados, la adquisición de componentes electrónicos puede ser costosa. Aun así, es un campo que no solo es apasionante sino que también tiene mucho futuro y representa grandes oportunidades.

Entusiasmo de la Comunidad Por último, pero no menos importante, probablemente el mayor impacto duradero de GNU/Linux en el modelo de ingeniería de Software se reduce a lo que podría llamarse entusiasmo de la comunidad. Me refiero a la forma en que GNU/Linux en particular, y el Software libre en general, ha animado a los desarrolladores de todo tipo a considerar las contribuciones a la comunidad como uno de sus objetivos finales y esto ahora es notorio no solo en Software, sino en Hardware abierto, obras literarias, escritos técnicos (como este trabajo), imágenes, vídeo, música y un largo etc.

En un mundo de código libre en el que las contribuciones a los proyectos de código pueden ser aceleradores de carrera y el código de licencia libre se reutiliza ampliamente, los desarrolladores entienden que hay un valor real en la construcción de Software que puede beneficiar a tantos usuarios como sea posible.

Tal vez los desarrolladores valorarían a la comunidad en su conjunto si GNU/Linux y el código libre nunca hubieran aparecido. Pero me cuesta imaginar un mundo en el que corporaciones como Microsoft y Google trabajarán juntos en la construcción de Software para GNU/Linux si GNU/Linux no hubiera popularizado el concepto de proyectos de Software impulsados por la comunidad que nadie posee realmente, pero que todos pueden utilizar.

Si bien, es innegable que todo lo anterior puso en la mira de las empresas de todos los tamaños y de las grandes corporaciones el Software libre, la principal razón es el poder utilizar una gran cantidad Software funcional, depurado y ampliamente usado para ofrecer servicios y/o productos basados en Software libre y así beneficiarse económicamente de ello.

Por otro lado, se ha visto a través de múltiples estudios, el impacto y la cuantificación de la importancia económica del código abierto aplicado tanto al Software como al Hardware, su efecto en la contribución al producto interior bruto generado, la reducción en aspectos como el coste total de propiedad, dependencia del proveedor y autonomía digital. Además de generar políticas públicas destinadas a lograr un sector público digitalmente autónomo, una investigación y desarrollo abierto que fomente el crecimiento de los países y una industria más digitalizada y competitiva.

Retomando la frase «el Software de código abierto nunca fue pensado para que las empresas de servicios lo tomasen y lo vendieran». ¿Para qué fue pensado el código abierto, entonces? No hay ninguna licencia de código abierto o Software libre reconocida por la Free Software Foundation o la Open Source Initiative que prohíba hacer negocio con el Software. Lo que prohíben es la discriminación en la capacidad y alcance de su uso en función de la parte, se trate de un individuo o de la mayor multinacional imaginable.

Pese a ello, esta situación está alterando el paradigma actual, en el que el modelo de desarrollo del código abierto se ha impuesto como impulsor de la innovación en el sector empresarial, a la postre el mayor estímulo que haya tenido el código abierto hasta la fecha.

¿Puede una filosofía de desarrollo como el código abierto, disponible para todo el mundo, llegar a convertirse en una fuente de ventajas diferenciales para el resto de los países, que se ha visto tradicionalmente muy superado en su relevancia en el entorno tecnológico por los gigantes tecnológicos de Estados Unidos o de China? Se afirma que su uso puede llegar a incidir en gran medida en el desarrollo de una independencia tecnológica superior, de una mayor competitividad y de más innovación. La idea, capitalizar la tecnología de una forma más orientada al procomún y al desarrollo colaborativo, suena sin duda atractiva e interesante pero no libre de inconvenientes para algunos sectores de desarrolladores de Software libre.

12.6 Código Abierto y las Organizaciones Internacionales

Aunque la Organización de las Naciones Unidas (ONU) ha hablado previamente bien del desarrollo del código abierto, varios eventos recientes muestran que la ONU está tomando medidas definitivas para presentar al mundo entero el camino del código abierto. En julio del 2021, el Consejo Económico y Social de la ONU (ECOSOC) adoptó un proyecto de resolución presentado por el representante de Pakistán titulado: Tecnologías de fuente abierta para

el desarrollo sostenible.

12.6.1 Las Naciones Unidas y el Código Abierto

El ECOSOC destacó la disponibilidad de tecnologías de código abierto que pueden contribuir a los Objetivos de Desarrollo Sostenible (ODS). El consejo invitó al Secretario General a "desarrollar propuestas específicas sobre formas de aprovechar mejor las tecnologías de código abierto para el desarrollo sostenible basadas en las aportaciones de los Estados Miembros interesados y otras partes interesadas".

El desarrollo de tecnología de código abierto puede ser una herramienta rápida y eficaz para la innovación. Aplicarlo a tecnologías apropiadas para ayudar a alcanzar los ODS es extremadamente prometedor. Las "tecnologías apropiadas" abarcan opciones y aplicaciones tecnológicas que son a pequeña escala, económicamente asequibles, descentralizadas, energéticamente eficientes, ambientalmente racionales y fácilmente utilizadas por las comunidades locales para satisfacer sus necesidades.

Existe un caso particularmente fuerte para las tecnologías apropiadas de código abierto OSAT (Outsourced Semiconductor Assembly and Test). OSAT podría ayudar a todos a salir de la pobreza y alcanzar un estado sostenible aprovechando el mismo tipo de desarrollo que hace que el Software de código abierto sea un éxito rotundo.

La Declaración Ministerial del Foro político de alto nivel sobre desarrollo sostenible también destacó la importancia de "tecnologías no patentadas que pueden contribuir a los Objetivos de Desarrollo Sostenible, a través de diversas fuentes de acceso abierto". Pidió "el desarrollo y la puesta en funcionamiento de una plataforma en línea en el marco del Mecanismo de facilitación de la tecnología para establecer un mapeo integral y servir como puerta de entrada a la información sobre iniciativas, mecanismos y programas de ciencia, tecnología e innovación existentes, dentro y fuera de las Naciones Unidas".

Es un pequeño paso, pero muy emocionante, porque las Naciones Unidas no se demoran una vez que ven formas de ayudar a sus Estados Miembros y a las personas que los integran. Ahora, el Departamento de Asuntos Económicos y Sociales de las Naciones Unidas (DESA) está trabajando para que esto suceda. DESA está utilizando una Nota sobre una base de datos centralizada propuesta de las Naciones Unidas de tecnologías apropiadas de código abierto publicada por la Conferencia de las Naciones Unidas sobre Comercio

y Desarrollo (UNCTAD) para hacerlo.

La Nota de la UNCTAD aboga por una base de datos centralizada de OSAT para acelerar el descubrimiento y la innovación en todos los sectores asociados con los ODS al tiempo que se minimizan los obstáculos legales o financieros. Esto es importante para la difusión del acervo mundial de conocimientos, especialmente en los países en desarrollo.

Actualmente, no existe un repositorio completo o una base de datos central de OSAT y Appropedia.org, quizás sea el mejor ejemplo. Sin embargo, la Nota de la UNCTAD dice: "Muchas organizaciones, organizaciones sin fines de lucro y empresas con fines de lucro están desarrollando OSAT y manteniendo bases de datos existentes a pequeña escala. Si bien hay muchos OSAT disponibles, se encuentran dispersos en varias bases de datos para tecnologías particulares. Mientras tanto, sigue existiendo una clara necesidad de aumentar la tasa de uso de OSAT.

Por lo tanto, existe una necesidad urgente de una base de datos de código abierto centralizada global (COSD) confiable. Al tener un alcance global, un repositorio de COSD proporcionaría una ventanilla única a la que todos pueden acceder para resolver los desafíos locales".

Concluye: "La ONU está bien posicionada para liderar el establecimiento de un COSD dado su papel bien establecido en la promoción de la tecnología de código abierto a través de varios foros y publicaciones intergubernamentales. En particular, 2030 Connect es una plataforma tecnológica en línea de la ONU que se desarrolló como parte del trabajo del Equipo de Trabajo Interinstitucional de la ONU. El COSD podría mejorarlo".

Con el liderazgo de la ONU, quizás no estemos demasiado lejos de cuándo, sí tiene un problema local (sin importar en qué parte del mundo se encuentre), pueda descargar una solución de código abierto examinada y probada. Quizás, esta es la potencia de fuego que necesitamos para alcanzar los ambiciosos Objetivos de Desarrollo Sostenible.

12.6.2 La Comisión Europea se Compromete a Liberar Todo el Software que Pueda Beneficiar a la Sociedad

A finales del 2021, la Unión Europea (UE) y su órgano legislativo, la Comisión Europea siguen avanzando en su estrategia digital con el Software de código abierto como uno de los pilares fundamentales. En esta ocasión ha sido esta última la que anuncia novedades para con la distribución del Software desarrollado para cubrir necesidades internas de la organización.

De acuerdo a la información publicada, la Comisión Europea ha aprobado una nueva regulación que favorece el libre acceso al Software que producen siempre y cuando existan beneficios potenciales para «los ciudadanos, las empresas u otros servicios públicos», lo que de la teoría a la práctica bien puede abarcar todo lo que se desarrolle bajo su tejado.

Esta nueva disposición se apoya a su vez en un reciente estudio realizado también por la Comisión sobre el impacto del Software de código abierto en áreas como la independencia tecnológica, la competitividad y la innovación en la economía de la Unión Europea. El objetivo, hallar evidencias sólidas con las que conformar las políticas europeas de código abierto para los próximos años.

En términos económicos, de hecho, los cálculos son de lo más optimistas y apuntan un impacto económico contundente, de miles de millones de euros de ahorro al año -a modo de ejemplo, se estimó entre 65 y 95 mil millones de euros solo en 2018- y con un incremento mínimo en la apuesta, se podría dar un crecimiento del PIB de la UE de en torno a los 100,000 millones de euros.

Con semejante escenario, no es de extrañar que la misma Comisión Europea esté interesada en promover las soluciones de código abierto dentro y fuera de las instituciones y no solo se basan en el beneficio económico directo: son muchas otras las ventajas del modelo también recogidas en el informe, tal y como se ha mencionado: independencia, competitividad, innovación... y en el caso de las administraciones públicas, colaboración, reutilización y transparencia.

En palabras de Johannes Hahn, comisario de Presupuesto y Administración: «El código abierto ofrece grandes ventajas en un ámbito en el que la UE puede desempeñar un papel de liderazgo. Las nuevas normas aumentarán la transparencia y ayudarán a la Comisión, así como a los ciudadanos, las empresas y los servicios públicos de toda Europa, a beneficiarse del desarrollo de Software de código abierto. Poner en común los esfuerzos para mejorar el Software y la creación conjunta de nuevas funciones reduce los costes para la sociedad, ya que también nos beneficiamos de las mejoras realizadas por otros desarrolladores. Esto también puede mejorar la seguridad, ya que especialistas externos e independientes comprueban los fallos y las deficiencias de seguridad de los programas informáticos».

La comisaría de Innovación, Investigación, Cultura, Educación y Juventud, Mariya Gabriel, ha declarado: «La Comisión pretende, con su ejemplo, estar al frente de la transición digital en Europa. Con las nuevas normas, la

Comisión aportará un valor significativo a las empresas, también las emergentes, a los innovadores, a los ciudadanos y las administraciones públicas, poniendo a su disposición el código abierto de sus soluciones informáticas. Esta decisión también ayudará a estimular la innovación, gracias al código de la Comisión disponible públicamente».

Como muestra del Software desarrollado bajo el amparo de la Comisión Europea que va a ser liberado se incluyen proyectos como eSignature, «un conjunto de normas, herramientas y servicios gratuitos que ayudan a las administraciones públicas y a las empresas a acelerar la creación y verificación de firmas electrónicas jurídicamente válidas en todos los Estados miembros de la UE»; o LEOS (Legislation Editing Open Software), «el Software utilizado en toda la Comisión para elaborar textos jurídicos. LEOS, escrito originalmente para la Comisión, se está desarrollando en estrecha colaboración con Alemania, España y Grecia».

Esta nueva iniciativa de la Comisión Europa contempla asimismo la creación de un repositorio centralizado para facilitar el descubrimiento, el acceso y la reutilización del Software incluido, el cual se sumará a todos los proyectos realizados por las diferentes administraciones públicas comunitarias en base al mismo modelo de desarrollo. Y viene de lejos este impulso, aun cuando comienza a unificarse ahora.

Sin ir más lejos, hace años que la propia Comisión Europea puso en marcha el programa Interoperable Delivery of European eGovernment Services to Public Administrations, Businesses and Citizens que dio origen al observatorio JoinUp (https://joinup.ec.europa.eu/), en cuyas páginas se recogen casi 3,000 soluciones de Software abierto, 133 colecciones de recursos y cuantiosa información relacionada.

Más tarde, de 2014 a 2017 se inició la «primera fase» en la estrategia de código abierto de la Unión Europea, especialmente dentro de la propia Comisión, estableciendo determinados requisitos en materia de Software de código abierto; actualmente se está desarrollando la nueva «estrategia de código abierto 2020-2023», con la que la Comisión Europea pretende ampliar y afianzar los objetivos de la estrategia digital y la contribución al programa Europa Digital.

13 Apéndice B: Algunos Consejos para Programadores

Hay muchos consejos para programadores novel, el más importante probablemente es:

13.1 ¿Cómo Hago mi Programa?

En su artículo "How to program it", Simon Thompson plantea algunas preguntas a sus alumnos que son muy útiles para la etapa de diseño:

- ¿Han visto este problema antes, aunque sea de manera ligeramente diferente?.
- ¿Conocen un problema relacionado? ¿Conocen un programa que pueda ser útil?.
- Fíjense en la especificación. Traten de encontrar un problema que les resulte familiar y que tenga la misma especificación o una parecida.
- Acá hay un problema relacionado con el que ustedes tienen y que ya fue resuelto. ¿Lo pueden usar? ¿Pueden usar sus resultados? ¿Pueden usar sus métodos? ¿Pueden agregarle alguna parte auxiliar a ese programa del que ya disponen?.
- Si no pueden resolver el problema propuesto, traten de resolver uno relacionado. ¿Pueden imaginarse uno relacionado que sea más fácil de resolver? ¿Uno más general? ¿Uno más específico? ¿Un problema análogo?.
- ¿Pueden resolver una parte del problema? ¿Pueden sacar algo útil de los datos de entrada? ¿Pueden pensar qué información es útil para calcular las salidas? ¿De qué manera se pueden manipular las entradas y las salidas de modo tal que estén "más cerca" unas de las otras? ¿Usaron todos los datos de entrada?.
- ¿Usaron las condiciones especiales sobre los datos de entrada que aparecen en el enunciado? ¿Han tenido en cuenta todos los requisitos que se enuncian en la especificación?.

Otros conjunto de consejos es conocido como SOLID de la programación orientada a objetos y el segundo grupo de consejos son los 10 Principios de Diseño para POO.

13.2 Diez Principios de la Usabilidad

Cuando desarrollamos un programa o aplicación informática hay algunos principios que nos pueden guiar para su usabilidad, estos son:

1. Visibilidad del estatus del sistema: *Le da certeza a los usuarios de lo que ocurre en cada paso.*
2. Compatibilidad entre el sistema y la vida real: *Entre la experiencia de la vida real y la experiencia digital.*
3. Dale libertad al usuario: *Genera una vía de escape.*
4. Consistencia y estándares: *Mantenlo simple.*
5. Prevén el error: *El usuario seguro encontrará la forma de cometer un error.*
6. Reconocer vs. Recordar: *Cuanta más información se le proporcione al usuario, se le brindan los elementos para reconocer el proceso.*
7. Flexibilidad y eficiencia: *Los usuarios experimentados deben tener prácticas que le permitan navegar más rápido en la plataforma. Y a los usuarios nuevos que les permita acercarse fácilmente y adoptar esas medidas para navegar.*
8. Diseño minimalista: *Diseño simple que no permita perderse entre un montón de elementos.*
9. Ayuda a los usuarios a reponerse de los errores: *Proveer una solución.*
10. Ayuda y documentación.

13.3 Principios SOLID de la Programación Orientada a Objetos

SOLID[119] es un acrónimo acuñado por Robert C.Martin para definir los cinco principios básicos de la programación orientada a objetos:

- Single responsibility
- Open-closed
- Liskov substitution
- Interface segregation
- Dependency inversion.

Estos principios no dejan de ser una filosofía, una manera de hacer las cosas que a la larga, hará que tus proyectos sean más estables, mantenibles, robustos y escalables en el futuro. Muchas veces ni los plazos ni el presupuesto te dejan tiempo para aplicar todas las buenas prácticas que se te puedan ocurrir, pero aplicar esos principios debería ser un deber en todos tus proyectos.

> Los buenos programadores usan sus cerebros, pero unas buenas directrices nos ahorran de tener que hacerlo en cada caso
> Francis Glassborow

Aplicando estas guías o directrices eliminarás problemas en un futuro, tendrás todo mejor organizado, y evitarás refactorizar clases para adaptarlas a nuevas funcionalidades futuras.

S: Single Responsability Principle (SRP) El principio de responsabilidad única se basa en que cada clase o método sólo debe hacer una cosa, sencilla y concreta. Si un objeto tiene un sólo cometido, éste será más fácil de mantener.

> A Class should have only one reason to change
> Robert C. Martin

[119] SOLID tiene bastante relación con los patrones de diseño.

Si un módulo o una clase agrupa funcionalidad muy dispersa, como por ejemplo la típica clase de utilidades que se utiliza a modo cajón de sastre, se dice que su cohesión es baja. Por el contrario, tiene una única responsabilidad, su cohesión es alta.

Una alta cohesión es deseable para mejorar características de nuestro código como la mantenibilidad y la reusabilidad.

Se viola este principio cuando por ejemplo en un mismo método se mezcla la lógica de negocio con la lógica de presentación, para ésto es mejor dos clases distintas, cada una para manejar su responsabilidad.

Otro ejemplo muy bueno es la típica función para leer los datos de un JSON, e importarlos a tu base de datos. Si hacemos todo el proceso en una única función o método, estaremos inclumpliendo SRP. Lo ideal sería que dividas la clase en diferentes métodos:

- Un método que obtenga el JSON
- Otro que compruebe que la estructura es la que deseamos
- Al recorrer el JSON, un método que compruebe si el dato ya existe en nuestra BBDD
- Otro que sí existe actualiza, de lo contrario lo crea

De este modo, cada método tendrá una única responsabilidad, y será más fácil realizar test unitarios

O: Open / Closed Principle El segundo principio SOLID es el Open/Closed. Está estrechamente relacionado con el SRP que vimos en el punto anterior.

El principio Open/Closed dice que una clase/método debe estar abierto a extensiones pero cerrado a modificaciones.

> Software entities (classes, modules, functions, etc...) should be open for extension but closed for modification
>
> Bertrand Meyer

Probablemente represente el mayor desafío cuando te pones a desarrollar una clase o método: pensar en el mañana. ¿Te ha pasado alguna vez que te toca modificar algo que desarrollaste hace unos meses, y que al modificar el código de un método afecta a otra parte de la aplicación sin darte cuenta?

Si el día de mañana el cliente solicita un nuevo requisito para algo que ya tienes desarrollado, si tienes en cuenta este principio, el comportamiento de esa clase debería ser extendido, nunca modificado. En caso contrario la probabilidad de que te encuentres con daños colaterales es muy alta.

L: Liskov Substitution Principle Este principio recibe el nombre (más bien el apellido) de la persona que lo acuñó: Barbara Liskov.

> Subtype Requirement: Let ℓ(x) be a property provable about objects x of type T. Then ℓ(y) should be true for objects y of type S where S is a subtype of T.
>
> Barbara Liskov

Este principio trata sobre que los objetos de un desarrollo deberían ser reemplazables por instancias de sus subtipos sin alterar el correcto funcionamiento del desarrollo. Dicho de otro modo: cualquier subclase debería poder ser sustituible por la clase padre.

De este modo, nuestros desarrollos mantendrán una integridad. De lo contrario, será un cochinero del que sólo tendremos constancia nosotros y luego el día de mañana lo heredará otra persona o tu yo del futuro, y lo que vendrá a continuación te suena, ¿no?

I: Interface Segregation Principle Este principio trata sobre algo parecido a SRP. Es mejor definir una serie de métodos abstractos a través de una serie de interfaces para que implementen nuestras clases.

Cada interface tendrá una única responsabilidad. Es preferible tener muchas interfaces que contengan pocos métodos que tener un interface con muchos métodos.

> Clients should not be forced to depend on methods they do not use
>
> Robert C. Martin

El objetivo no es otro que poder reutilizar estas interfaces en otras clases. Cada clase implementará las interfaces que necesite y use, ninguna más. No contendrá métodos que no utilice. De nuevo con el objetivo de aplicar sentido común y no ensuciar una clase con métodos o propiedades que no se utilizan.

D: Dependency Inversion Principle El objetivo de este principio es conseguir desacoplar las clases de nuestro desarrollo. Que una clase pueda funcionar por sí sola sin depender de otra. Es difícil, pero en todo diseño de Software al final suele existir un acoplamiento, pero hay que evitarlo en la medida de lo posible. Un sistema altamente acoplado es muy difícil de mantener.

> High-level modules should not depend on low-level modules. Both should depend on abstractions
> Abstractions should not depend upon details. Details should depend upon abstractions
> Robert C. Martin

El objetivo es conseguir que una clase interactúe con otras clases sin que las conozca directamente.

13.4 10 Principios de Diseño para POO

Empezando a programar con el paradigma de Objetos es un tema muy particular. Mucha gente cree que con el hecho de crear objetos y clases ya está aplicando el paradigma, pero en realidad no. El paradigma consiste no en un aspecto técnico, sino en un aspecto funcional, que puedas adaptarte a una forma de programar, en vez de utilizar formas de programación que cualquier lenguaje puede incluir.

Aquí les compartimos estos 10 principios de diseño para la programación orientada a objetos, esperando que les sirva para mejorar la forma en que hacen código y lo usan:

1. DRY (Don't repeat yourself) El principio de la POO es la de no repetir el código, hacer código reusable y abstracto, para que puedas usarlo de nuevo sin necesidad de volver a escribir desde cero. Si tu código es imposible de reusarlo en otra aplicación, o tienes que programar ciertas partes de nuevo porque no se puede adecuar, es un síntoma de que necesitas hacer funciones más abstractas, más genéricas para que puedan ser extendidas con requerimientos más puntuales.

2. Encapusla No hay mucha gente que entienda la importancia de la encapsulación, pero es muy simple. Aprende a encapsular el alcance de tus clases. La forma más fácil de hacerlo es poniendo privadas las variables y métodos que son exclusivos de tu clase, y poniendo públicos los métodos que pueden ser usados por otros objetos. Hay más formas de encapsular el código, pero esos dos son los básicos.

3. Principio de Diseño Abierto-Cerrado Este principio dice que las clases, métodos y funciones deben estar abiertos para extender su funcionalidad, pero cerrados para ser modificados. Esto significa que tu código debería ser modificado casi nunca pero sí debería servir como base para extenderlo, es decir, para usar herencia, para usar interfaces, para ser la base de nuevas clases, a fin de que nunca llegues a modificar tu código existente para crear nuevo, sino crear código sobre la base que ya tienes.

Esto podemos verlo como si fuera una caja negra, donde tu debes hacer que tu código sea imposible de modificar y solo pueda ser usado desde sus métodos públicos para poder ser extendido.

4. Principio de Responsabilidad Única Este principio hace referencia a que nuestras clases y métodos deben tener una sola tarea o actividad. Una clase que se encarga de operar datos no debería administrar componentes gráficos, ni un método que tiene que mandar un correo electrónico debería manejar funciones adicionales. Esto hace que tus funciones cumplan un solo propósito y hace más fácil que las puedas utilizar más adelante.

5. Inyección de Dependencias Las dependencias es cuando en una clase necesitas otra para poder trabajar. Piensa en una clase que toma como base otra para mandar un mensaje de texto. La clase original tiene una dependencia y si quisieras usar esa clase pero no quisieras mandar un mensaje de texto hace que tengas que modificar el código para ello. La mejor forma de resolver un problema así es a través de la inyección de dependencias en donde en vez que que la clase original mande a llamar a la segunda clase, se le inyecte a través del uso de Frameworks o interfaces para mandar a llamar a algo que después tiene que ser completado, y de esa forma no dependa de otras clases.

6. Composición sobre Herencia La composición es justamente lo opuesto a la herencia, que es mandar a llamar objetos en vez de heredarlos. Esto es bueno en la medida en que a partir de una misma relación en vez de estar heredando clases y hacer una jerarquía grande, muchas veces para simplificar trabajar con objetos de una misma base en mejor hacerlo a través de la composición o creación de objetos. Para saber más sobre composición y asociación pueden leer este artículo.

7. Principio de Sustitución Liskov Este principio dice que si tienes un objeto de subtipo S que deriva de un tipo T deberías tener la posibilidad de usar el objeto de tipo T de la misma forma que uses al subtipo S. Como ejemplo, tenemos que un objeto de tipo Perro deriva de un objeto de tipo Animal, y si tienes una función que necesita un tipo Perro como parámetro, si colocas un objeto de tipo Animal debería funcionar la clase de la misma forma, sin errores.

8. Principio de Segregación de Interfaces Este principio refiere a que no incluyamos métodos que no va a utilizar el programador en una interfaz. Cuando creamos interfaces podemos colocar métodos que queremos que el programador use para extender su código, pero si metemos funciones que no va a usar o no siempre va a usar, estamos cayendo en este error de POO.

9. Programación de Interfaces no Implementaciones Tu código siempre debe estar enfocado a definir interfaces, no implementación de las mismas. Las interfaces ya nos dan una forma abstracta de poder hacer funciones genéricas que puedan ser usadas en muchos escenarios. Hay que mantener esa estrategia para crear código que puedas reusar.

10. Principios de Delegación Este último principio dice que no hay que hacer todo el código nosotros, ya hay clases a las que le podemos delegar esa tarea, como por ejemplo comparar dos objetos. Todos los objetos tienen la clase equals() por lo cual es innecesario implementar un nuevo método para esa funcionalidad. Es un ejemplo básico pero nos dice que hay que conocer los métodos de los objetos para saber qué ya hay código hecho que podemos usar y no todo a fuerza lo tenemos que implementar desde cero.

13.5 Errores en Programación

Al programar es necesario revisar nuestro código por un compilador y los errores son inherentes al proceso de programación. Los errores de programación responden a diferentes tipos y pueden clasificarse dependiendo de la fase en que se presenten. Algunos tipos de errores son más difíciles de detectar y reparar que otros, veamos entonces:

- Errores de sintaxis
- Advertencias
- Errores de enlazado
- Errores de ejecución
- Errores de diseño

Errores de sintaxis son errores en el código fuente. Pueden deberse a palabras reservadas mal escritas, expresiones erróneas o incompletas, variables que no han sido declaradas, etc. Los errores de sintaxis se detectan en la fase de compilación. El compilador, además de generar el código objeto, nos dará una lista de errores de sintaxis. De hecho nos dará sólo una cosa o la otra, ya que si hay errores no es posible generar un código objeto.

Advertencias además de errores, el compilador puede dar también advertencias (Warnings). Las advertencias son errores, pero no lo suficientemente graves como para impedir la generación del código objeto. No obstante, es importante corregir estos errores la mayoría de las veces, ya que ante un aviso el compilador tiene que tomar decisiones, y estas no tienen por qué coincidir con lo que nosotros pretendemos hacer, ya se basan en las directivas que los creadores del compilador decidieron durante la creación del compilador. Por lo tanto en ocasiones, ignorar las advertencias puede ocasionar que nuestro programa arroje resultados inesperados o erróneos.

Errores de enlazado el programa enlazador también puede encontrar errores. Normalmente se refieren a funciones que no están definidas en ninguno de los ficheros objetos ni en las bibliotecas. Puede que hayamos olvidado incluir alguna biblioteca, o algún fichero objeto, o puede que hayamos olvidado definir alguna función o variable, o lo hayamos hecho mal.

Errores de ejecución incluso después de obtener un fichero ejecutable, es posible que se produzcan errores, durante la ejecución del código. En el caso de los errores de ejecución normalmente no obtendremos mensajes de error muy específicos o incluso puede que no obtengamos ningún error, sino que simplemente el programa terminará inesperadamente. Estos errores son más difíciles de detectar y corregir (pues se trata de la lógica como tal de nuestra aplicación). Existen herramientas auxiliares para buscar estos errores, son los llamados depuradores (Debuggers). Estos programas permiten detener la ejecución de nuestros programas, inspeccionar variables y ejecutar nuestro programa paso a paso (instrucción a instrucción). Esto resulta útil para detectar excepciones, errores sutiles, y fallos que se presentan dependiendo de circunstancias distintas. Generalmente los errores en tiempo de ejecución se dan por situaciones no consideradas en la aplicación, por ejemplo, que el usuario ingrese una letra en vez de un número y esto no se controle.

Errores de diseño finalmente los errores más difíciles de corregir y prevenir. Si nos hemos equivocado al diseñar nuestro algoritmo, no habrá ningún programa que nos pueda ayudar a corregirlos, pues es imposible que un programa pueda determinar qué es lo que tratamos de conseguir o un programa que realice aplicaciones cualquiera por nosotros. Contra estos errores sólo cabe practicar y pensar, realizar pruebas de escritorio, hacerle seguimiento y depuración a la aplicación hasta dar con el problema (una mala asignación, un valor inesperado, olvidar actualizar una variable, etc.), también es útil buscar un poco de ayuda de libros o en sitios y foros especializados.

En este resumen, se enumera algunas de las cosas que han aportado muchos programadores para todos aquellos que se inician en el arte y ciencia de la programación (recordando que se puede pasar más tiempo encontrando y corrigiendo errores que en el desarrollo de nuevas características en el código). Se leen rápido, pero aplicarlos bien puede llevar toda la vida.

1. Tome el código y divida grandes partes de código en pequeñas funciones.

2. Si para cuando sale del trabajo no ha resuelto el problema. Apague la computadora y déjela para el día siguiente. No pienses más en el problema.

3. Principio YAGNI: "No lo necesitará" o no codifique más de lo que se le pidió. No anticipe el futuro y simplemente cree algo que funcione

lo antes posible. Codifique solo las partes necesarias para resolver el problema actual.

4. No es necesario saberlo todo, ni todos los marcos existentes. Lo más significativo es tener una buena base. Conozca el lenguaje en profundidad antes de comenzar con un Framework y aprenda cosas fundamentales como los principios SOLID o cómo escribir código limpio.

5. KISS: "Keep it simple, stupid" o "keep it stupid simple" es un principio de diseño que establece que la mayoría de los sistemas funcionan mejor si se mantienen simples en lugar de complicados. Y si bien esto es lógico, a veces es difícil de lograr.

6. No lo pienses demasiado.

7. Si tiene un problema o un error durante demasiado tiempo, aléjese y vuelva más tarde. A menudo, las mejores soluciones a los problemas se me ocurren en el camino a la oficina, al baño. También es aconsejable alejarse cuando está enojado con un cliente o con un compañero de trabajo, especialmente si deseas conservar tu trabajo.

8. Aprenda a escribir pruebas útiles y aprenda a hacer TDD. TDD es un proceso de desarrollo de Software que se basa en la repetición de un ciclo de desarrollo muy corto: escriba una prueba, ejecute todas las pruebas y vea si la nueva falla, escriba algún código, ejecute pruebas, refactorice el código, repita.

9. Primero resuelva el problema y luego escriba el código. No empiece a codificar sin saber qué hacer.

10. No memorice el código, en su lugar, comprenda la lógica.

11. Si copia y pega una solución de desbordamiento de pila, asegúrese de comprenderla. Aprenda a usar Stack Overflow de una buena manera.

12. Si quieres aprender algo, practica. Haz ejemplos y haz que funcionen porque leer sobre algo no es suficiente.

13. Estudie el código de otras personas y deje que otros estudien su código de vez en cuando. La programación en pareja y las revisiones de código son una buena idea.

14. No reinvente la rueda.
15. Tu código es la mejor documentación.
16. Sepa cómo googlear cosas. Para ello, es necesario tener experiencia y leer mucho para saber qué buscar.
17. Tu código deberá ser mantenido por ti mismo en el futuro o por otros, así que escribe el código con el lector en mente, no tratando de ser la persona más inteligente. Haz que se lea como si estuvieras leyendo una historia.
18. La mejor manera de resolver un error con Google es copiarlo y pegarlo.
19. Nunca te rindas, al final, de una forma u otra lo resolverás. Hay días malos, pero pasarán.
20. Descanso, descanso y descanso. La mejor manera de resolver un problema es tener una mente tranquila.
21. Aprenda a utilizar los patrones de diseño de Software. Los patrones de diseño son soluciones a problemas comunes en el diseño de Software. Cada patrón es como un plano que puede personalizar para resolver un problema de diseño común en su código. (No reinvente la rueda)
22. Utilice herramientas de integración y automatice tanto como pueda.
23. Hacer katas de código. Un kata de código es un ejercicio de programación que ayuda a los programadores a mejorar sus habilidades mediante la práctica y la repetición.
24. Programe en una interfaz, no en una implementación. La inyección de dependencia es un requisito. Consulte los principios SOLID.
25. Refactor -Test-Refactor. La refactorización es una técnica para reestructurar un código existente, alterando y mejorando su estructura interna sin cambiar su comportamiento externo.
26. Pide ayuda cuando la necesites. No pierdas el tiempo.
27. La práctica hace la perfección.

28. Aunque a veces los comentarios pueden ayudarte, no les prestes demasiada atención. Probablemente estén desactualizados.

29. Conozca su entorno de desarrollo e invierta en uno lo suficientemente potente.

30. Reutilizar componentes.

31. Al desarrollar una aplicación Web, piense primero en los dispositivos móviles y en las restricciones de ancho de banda y energía asociadas.

32. No optimice ni refactorice antes de tiempo. Es más importante tener un producto mínimo viable lo antes posible.

33. Nunca elija una forma de atajo ineficiente para ahorrar unos minutos. ¡Cada vez que codifique, dé lo mejor!

34. Siga los estándares documentados.

35. Los usuarios no son personas técnicas. Piense en ello cuando desarrolle su interfaz de usuario.

36. Utilice siempre un sistema de control de código fuente como GitLab, Github o bitbucket y realice confirmaciones de git pequeñas y frecuentes.

37. Es mejor usar registros que depurar el código. Registre todas las partes críticas.

38. Sea consistente al codificar. Si usa un estilo, use siempre el mismo. Si trabaja con más personas, use el mismo estilo con todo el equipo.

39. No dejes de aprender, pero más que nuevos lenguajes o marcos, céntrate en los conceptos básicos del desarrollo de Software.

40. Y finalmente, paciencia y ama lo que haces.

13.5.1 Errores en Java

Ejemplo en Java:

```
class hola {
   public static void main(String[] args) {
      System.out.println("Hello world!");
   }
}
```

para compilarlo usamos *javac* en línea de comandos mediante:

```
$ javac hola.java
```

y lo ejecutamos con:

```
$ java hola
```

Errores en tiempo de compilación Los errores de compilación ocurren porque la sintaxis del lenguaje no es correcta, este tipo de errores no permiten que la aplicación se ejecute, por ejemplo:

- Olvidarnos de un punto y coma al final de una sentencia.
- No cerrar llaves en algún bloque de código, método, clase o en alguna estructura de control.
- Repetir variables con el mismo nombre aunque sean de diferente tipo.
- No distinguir bien los diferentes tipos primitivos (boolean, byte, short, int, long, float y double) y no saber elegir el más adecuado a cada problema.
- Asignar en una variable otra variable con tipo de dato diferente: es decir si tengo una variable String y su valor lo asigno en una variable de tipo int, en este caso el compilador me va dar un error de conversión de tipos y esto no va dejar que la aplicación compile, incluso un error parecido se puede dar en variables de grupos del mismo tipo, por ejemplo asignar el valor de un tipo int en un tipo short.

- Error en mayúsculas y minúsculas, Java es un lenguaje case sensitive, esto es que debemos respetar las mayúsculas y minúsculas. (miVar no es igual a mivar). Si hacemos referencia a un identificador que no está bien escrito entonces obtendremos un error de sintaxis.

- No llevar una cuenta clara y ordenada de los paréntesis (), llaves {} y comillas "" abiertos, lo cual conduce a que algunas ovejas vayan con parejas que no son las suyas o queden del todo desparejadas.

- No tener en cuenta el orden en que se realizan las operaciones en las expresiones de Java, de modo que en la práctica se obtiene un resultado distinto del que se espera.

- No poner bien las llaves en las sentencias if, while y for. Olvidarse por ejemplo de que una sentencia for(int i=0; i<n; i++); repite n veces la sentencia vacía (por la presencia del punto y coma final) en lugar de la sentencia siguiente al for.

- Usando equals contra la asignación (== versus =), en Java este error puede ser incluso un error de lógica o sintaxis. Para comparar 2 referencias para igualarlas se usa el operador == (el operador de igualdad). Para asignar el valor de la derecha a la variable de la izquierda se usa el operador = (operador de asignación). Los programadores novatos a veces escriben: if (miValor = valorEsperado) Este código intenta evaluar el valorEsperado como un valor booleano en lugar de intentar la evaluación de igualdad entre miValor y valorEsperado. Si valorEsperado es del tipo booleano, entonces el código tendrá un error de lógica y probará si el valorEsperado es verdadero o falso. Si valorEsperado no es del tipo booleano, entonces el código lanzará un error de compilación debido a que la estructura if requiere un valor booleano que sea retornado de la comparación (miValor = valorEsperado), pero en Java el operador = siempre retorna el valor de la derecha.

- No tener la idea clara de cómo funcionan los métodos, de lo que son los argumentos y de cómo se pasan, y de lo que es el valor de retorno y cómo se devuelve.

- Definir en un método una variable local con el mismo nombre que una variable miembro de su clase, y olvidarse de poner la referencia this cuando se quiere acceder a la variable miembro.

- Crear la referencia a un objeto y olvidarse de crear el objeto. Se obtiene un error cuando se intenta acceder a alguna variable miembro o método del objeto.
- No entender bien los métodos gráficos fundamentales (paint(), update() y repaint()) y no saber redefinirlos de acuerdo con el problema que se desea resolver.
- No saber encontrar en la documentación de Java lo que se necesita para resolver una determinada dificultad o problema de programación.
- No saber interpretar los mensajes de error del compilador (o no intentarlo siquiera), para obtener pistas que ayuden a su detección y corrección.
- No saber qué hacer cuando el problema compila y arranca bien, pero da errores en tiempo de ejecución o simplemente da un resultado incorrecto. "¿El debugger? Sí, me suena que era para algo de esto..."
- No tener una idea clara de lo que es un cast y de cuándo es imprescindible utilizarlo para que un programa funcione.
- No saber qué hacer cuando se desea utilizar un método de Java que lanza una excepción y nos da error al compilar por no haber capturado y gestionado dicha excepción.
- Tener un proyecto activo con una enorme confusión de ficheros y directorios, por ejemplo de modo que se utilicen y modifiquen ficheros del proyecto anterior destruyendo lo que ya funcionaba, etc., etc.
- No guardar con frecuencia los ficheros del proyecto, arriesgándose a perder en cualquier momento todo el trabajo realizado.
- No programar de una manera limpia, clara y ordenada; no introducir ningún comentario; elegir nombres de variables que no mantienen ninguna semejanza o relación con la magnitud a la que representan.
- No detenerse a pensar -preferiblemente sobre papel- el esquema general del programa a realizar. No intentar definir primero un esqueleto y luego todos los detalles de terminación. Conceder a todos los puntos exactamente la misma importancia.

- No desarrollar progresivamente, comprobando cada paso antes de pasar al siguiente. Olvidarse de que más vale un programa incompleto que lo que hace lo hace bien, que un programa que tiene todo pero que todo o casi todo lo hace mal.

- El nombre de la Clase pública no coincide con el nombre del archivo, cada archivo .java puede contener sólo una Clase pública. El nombre de esa Clase pública debe coincidir exactamente con el nombre del archivo antes de la extensión .java, respetando incluso las mayúsculas y minúsculas. Por ejemplo, una Clase pública llamada MyClass debe estar en un archivo MyClass.java y no en myclass.java. Este es un error de sintaxis.

- Una Clase no está en el directorio correcto, este error de sintaxis ocurre si el comando javac no puede encontrar un archivo .java en el directorio esperado. Si una Clase está en el paquete por defecto (no tiene la declaración del package en la primera línea), entonces esta Clase pertenece al directorio actual de trabajo o al directorio donde apunta el flag sourcepath del javac.

Errores en tiempo de ejecución Estos errores ocurren cuando la aplicación se está ejecutando, imagínate que cuando estás haciendo la presentación o probando tu aplicación en público o frente a tu profesor y de repente pummm la aplicación explota (se cuelga) por lo general a este tipo de errores se los conoce como errores de compilación, ahora, por qué ocurren estos errores?, bueno, hay muchos factores desde los más básicos como por ejemplo:

- El usuario ingresa valores diferentes a los que la aplicación recibe

- Acceder a una posición en un arreglo la cual no existe.

- Almacenar cadenas donde se debe almacenar números

- Divisiones por cero.

- Digamos que en una aplicación móvil consuma datos de un servicio web y que al momento de consumir esos datos no haya conexión a internet, esto hace que la aplicación se cuelgue.

- Olvidar que los índices en Java empiezan en 0, los índices de los arreglos de Java y las listas empiezan en 0, myArray[0], o myList.get(0). Asegurarse que su loop for no cause errores por este motivo. Si hacemos más loops de los que son posibles, entonces obtendremos el error: ArrayIndexOutBounds exception. Si hacemos menos loops de los requeridos, entonces tendremos un error de lógica.

- NullPointerException, el NullPointerException, la maldición de los programadores en Java. Los NullPointerExceptions son errores de lógica causados cuando un programa intenta acceder a métodos o atributos en una referencia que está nula. Si nuestro objeto no ha sido inicializado o ha sido establecido a null con el operador =, entonces la llamada el método o el acceso a uno de sus atributos no es válido.

Errores lógicos Estos son los más difíciles de detectar y corregir y pues bueno porque digo que son difíciles, es que con este tipo de errores la aplicación compila y se ejecuta de forma normal, pero, y entonces dónde está el error? El error se da porque la aplicación no muestra los resultados esperados digamos que ya hice el algoritmo y el resultado que debo obtener es 10, pero en la aplicación obtengo 5 o lo peor de todo que de seguro te a pasado, que escribes el código haces pruebas y los resultados son correctos subes a producción y resulta que para un caso particular no se obtuvo el resultado esperado. Pues en estos casos el problema son errores lógicos que a fin de cuentas estos ocurren por el mal diseño del algoritmo de la aplicación

Para terminar también mencionar que otro error de tipo lógico también ocurre cuando colocamos punto y coma después de una sentencia if o for y aunque parezcan errores tan obvios, se dan cuando estamos empezando a programar.

Por ello, estos son algunos consejos del buen programador:

- Tener muy cuenta las divisiones por cero, en cuanto a los arreglos no acceder a posiciones que no existen.

- Hacer uso de excepciones y validaciones cuando se tenga en cuenta que el usuario puede ingresar valores diferentes a los que se puede recibir la aplicación.

- Cuando diseñemos un algoritmo tener en cuenta todos los escenarios posibles que pueda tomar la aplicación durante su ejecución.

- Finalmente como dice el dicho divide y vencerás: una buena forma de contrarrestar los errores de tipo lógico es dividir algoritmos grandes y complejos en tareas pequeñas de forma que el código se más legible y cualquier error de tipo lógico se lo pueda depurar fácilmente.

Lista de errores comunes

- Incompatyble types: unexpected return value:

  (Tipos incompatibles: valor de retorno no esperado) El mensaje de error te indica que existen tipos incompatibles. ¿Qué hacer? Revisa el tipo de método y la instrucción return si la incluye.

  Recuerda que:

  Declaración del método void →Sentencia return no debe de existir en este tipo de método.

  Declaración del método int, float, double, char, boolean→Sentencia return debe existir, regresando un valor del tipo apropiado

- Cannot find symbol

  (No se puede encontrar el símbolo) El mensaje de error te indica que existe un símbolo que no se reconoce por que no está declarado. ¿Qué hacer? Revisa que la variable esté declarada como variable local del método, verifica que el nombre del método invocado sea el correcto, o asegúrate que el nombre de la variable global o local esté bien escrita.

- Invalid method declaration; return type required

  (Declaración inválida del método; tipo de regreso requerido) El mensaje de error te indica el método no ha sido declarado apropiadamente y que se requiere un tipo. ¿Qué hacer? Revisa que el método tenga especificado un tipo de método, ya sea alguno de los tipos primitivos de datos o incluso en tipo void si no regresará valores, la declaración le falta el tipo de valor que el método regresará, ya sea int, float, char, double, boolean o void.

- Ilegal start of expression

(Comienzo ilegal de expresión) Este mensaje de error aparece al inicio de un nuevo método, ya sea uno definido por el usuario o el mismo main. Puedes pasar horas revisando la sentencia donde el error se posiciona, y encontrarás que no hay error. Y eso es debido a que un método anterior no se ha cerrado, cualquier inicio de otro método generará error de sintaxis. ¿Qué hacer? Revisa que el método anterior tenga su llave de cierre.

- Missing return statement

  (Falta la sentencia return) Este error se genera simplemente porque se ha olvidado la sentencia return, la cual es obligatoria en los métodos de tipo int, char, float, double y boolean; además de cualquier tipo definido por el usuario. ¿Qué hacer? Asegúrate que, si el método no es de tipo void, exista la sentencia de return.

- class, interface or enum expected

  (Se esperabla class, interface o enum) El mensaje explica que se espera el inicio de una clase, interface o enumeración en esa parte del código. ¿Por qué? Muy sencillo, porque la clase actual ya ha sido cerrada. ¿Qué hacer? Revisa que en el método anterior no sobre una llave de cierre.

Esta es una lista muy pequeña de los errores más comunes que se pueden presentar cuando se trabaja con un programa modular.

13.5.2 Errores en Pyhton

Python es un lenguaje potente y flexible con muchos mecanismos y paradigmas que pueden mejorar significativamente el rendimiento. Sin embargo, como con cualquier herramienta de software o lenguaje, con una comprensión o evaluación limitada de sus capacidades, pueden surgir problemas imprevistos durante el desarrollo.

Ejemplo en Python 2:

```
print "Hello World\n"
```

para compilarlo y ejecutarlo usamos *python2* en línea de comandos mediante:

```
$ python2 hola.py
```

Ejemplo en Python 3:

```
print ("Hello World\n")
```

para compilarlo y ejecutarlo usamos *python3* en línea de comandos mediante:

```
$ python3 hola.py
```

Errores en tiempo de compilación Los errores de compilación ocurren porque la sintaxis del lenguaje no es correcta, este tipo de errores no permiten que la aplicación se ejecute, por ejemplo:

- Python distingue entre las mayúsculas y las minúsculas ("case sensitive").

  MapSize = "34x44"

  mapsize = "22x34"

- Indentación incorrecta

  Atención a la alineación del código. Esto es, a la hora de escribir funciones, etc., existen una sangrías que deben respetarse para que no nos de error. Mientras que la mayoría de los lenguajes utilizan la sangría para mejorar su legibilidad, pero no dependen de la práctica, Python ha tejido la sangría directamente en el tejido de su lenguaje. Esto significa que no puede permitirse cometer errores cuando se trata de formatear su código. Para Python, la regla es tener 4 espacios para sangrar. No mezcles con 2,3,5 o una cantidad diferente de espacios ya que Python simplemente no ejecutará tu programa. Necesitarás aprender a usar Python para separar tus bloques de código para que todo fluya bien. Es importante asegurarse de que lo haga bien la primera vez. No sangrar correctamente puede llevar a un error en su código si no tiene cuidado. Debugging ya es bastante pesado por sí solo, pero buscar un espacio que falta o un espacio extra te volverá loco. Por

ejemplo en el caso de la sentencia "except" debe estar a la altura de "try" ya que funcionan como un bloque. Todas las sentencias tabuladas a la misma distancia pertenecen al mismo bloque de código hasta que no se encuentre una línea con menor tabulación

- Hay que tener cuidado a la hora de escribir las rutas para encontrar tus datos. La barra que entiende Python es / ó \\ (en esta posición \ posee otro significado para este lenguaje de programación).
- Los espacios como guión bajo. Hay ocasiones que escribimos las carpetas con espacios o recibimos datos que los incluye, es una mala costumbre al trabajar. A la hora de escribir tenemos que tenerlo muy en cuenta y poner dicho guión bajo.
- Al realizar una comparación entre dos objetos o valor, se tiene que utilizar el operador de igualdad (==), no el operador de asignación (=). El operador de asignación coloca un objeto o valor dentro de una variable y no compara nada.
- Olvidar poner los ":"al final de las sentencias tipo: try, if, elif, else, for, while, class, o def.
- Identificados con el código SyntaxError, son los que podemos apreciar repasando el código, por ejemplo al dejarnos de cerrar un paréntesis:

  ```
  print("Hola"
  ```

- Errores de nombre, se producen cuando el sistema interpreta que debe ejecutar alguna función, método... pero no lo encuentra definido. Devuelven el código NameError:

  ```
  pint("Hola")
  ```

La mayoría de errores sintácticos y de nombre los identifican los editores de código antes de la ejecución, pero existen otros tipos que pasan más desapercibidos.

Errores semánticos, Estos errores son muy difíciles de identificar porque van ligados al sentido del funcionamiento y dependen de la situación. Algunas veces pueden ocurrir y otras no. La mejor forma de prevenirlos es programando mucho y aprendiendo de tus propios fallos, la experiencia es la clave. Veamos un par de ejemplos:

- Ejemplo pop() con lista vacía

  Si intentamos sacar un elemento de una lista vacía, algo que no tiene mucho sentido, el programa dará fallo de tipo IndexError. Esta situación ocurre sólo durante la ejecución del programa, por lo que los editores no lo detectarán:

  ```
  l = []
  l.pop()
  ```

  Para prevenir el error deberíamos comprobar que una lista tenga como mínimo un elemento antes de intentar sacarlo, algo factible utilizando la función len():

  ```
  l = []
  if len(l) > 0:
      l.pop()
  ```

- Ejemplo lectura de cadena y operación sin conversión a número

  Cuando leemos un valor con la función input(), éste siempre se obtendrá como una cadena de caracteres. Si intentamos operarlo directamente con otros números tendremos un fallo TypeError que tampoco detectan los editores de código:

  ```
  n = input("Introduce un número: ")
  print("{}/{} = {}".format(n,m,n/m))
  ```

  Como ya sabemos este error se puede prevenir transformando la cadena a entero o flotante:

  ```
  n = float(input("Introduce un número: "))
  m = 4
  print("{}/{} = {}".format(n,m,n/m))
  ```

Sin embargo no siempre se puede prevenir, como cuando se introduce una cadena que no es un número:

```
n = float(input("Introduce un número: "))
m = 4
print("{}/{} = {}".format(n,m,n/m))
```

Como podemos suponer, es difícil prevenir fallos que ni siquiera nos habíamos planteado que podían existir. Por suerte para esas situaciones existen las excepciones.

14 Bibliografía

Este texto es una recopilación de múltiples fuentes, nuestra aportación —si es que podemos llamarla así— es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado —en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa— múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
http://132.248.181.216/Herramientas/

Referencias

[1] Carreras en la Facultad de Ciencias, UNAM, http://www.fciencias.unam.mx/licenciatura/resumen/

[2] Aulas y Talleres en el Tlahuizcalpan de las Carreras de Matemáticas de la Facultad de Ciencias, UNAM, http://www.matematicas.unam.mx/tlahuiz/

[3] http://es.wikipedia.org/wiki/Microsoft_Windows 67

[4] http://es.wikipedia.org/wiki/Linux 81

[5] http://es.wikipedia.org/wiki/Mac_OS 78

[6] http://es.wikipedia.org/wiki/Android 95

[7] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n) 194, 308

[8] https://es.wikipedia.org/wiki/Python 194, 317

[9] https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n) 194, 337

[10] https://es.wikipedia.org/wiki/C%2B%2B 194, 341

[11] http://www.gnu.org/philosophy/free-sw.es.html 574

[12] http://es.wikipedia.org/wiki/Software_libre 574

[13] http://www.hispalinux.es/SoftwareLibre 574

[14] http://es.wikipedia.org/wiki/Software_propietario 572

[15] Diferentes Tipos de Licencias para el Software, http://www.gnu.org/licenses/license-list.html 309, 574, 584

[16] FSF, Free Software Fundation, http://www.fsf.org/ 574, 584

[17] GNU Operating System, http://www.gnu.org/ 574, 584

[18] LaTeX, A Document Preparation System, http://www.latex-project.org/ 350

[19] MATLAB, MathWorks, http://www.mathworks.com/products/matlab/

[20] SCILAB, Scilab Open Source for Numerical Computation, http://www.scilab.org/

[21] Julia, The Julia Programming Language, https://julialang.org 334

[22] FreeMat, FreeMat Open Source for rapid engineering and scientific prototyping and data processing, http://freemat.sourceforge.net/

[23] OCTAVE, GNU Octave, https://www.gnu.org/software/octave/

[24] SCIPY Open Source Library of Scientific Tools, http://www.scipy.org/

[25] Python Programming Language, http://www.python.org/

[26] SAS, Business Analytics and Business Intelligence Software, http://www.sas.com/

[27] IBM SPSS Software, http://www-01.ibm.com/Software/analytics/spss/

[28] GNU PSPP, http://www.gnu.org/software/pspp/

[29] The R Project for Statistical Computing, http://www.r-project.org/ 350

[30] LINPACK, http://www.netlib.org/linpack/

[31] EISPACK, http://www.netlib.org/eispack/

[32] Scicos Block Diagram Modeler/Simulator, http://www.scicos.org/

[33] Simulink Simulation and Model-Based Design, https://es.mathworks.com/products/simulink.html

[34] MATLAB to Scilab conversion tips, http://help.scilab.org/docs/6.0.1/en_-US/m2sci_doc.html

[35] EViews, http://www.eviews.com/home.html

[36] Gretel, http://gretl.sourceforge.net/

[37] Statgraphics, http://statgraphics.softonic.com/

[38] Statistica, https://www.tibco.com/es/products/tibco-statistica

[39] Systat, https://www.systatsoftware.com/

[40] Vensim, http://vensim.com/vensim-Software/

[41] Maple, http://www.maplesoft.com/

[42] Mathematica, http://www.wolfram.com/mathematica/

[43] Maxima, http://maxima.sourceforge.net

[44] RWeka, https://cran.r-project.org/Web/packages/RWeka/index.html 350

[45] Tinn-R Edit code and run it in R, http://nbcgib.uesc.br/lec/software/editores/tinn-r/en 350

[46] RStudio Software, Education, and Services for the R community, http://www.rstudio.com/ 350

[47] Suite de código abierto con más de 700 paquetes para el desarrollo de la ciencia de datos con Python, https://www.anaconda.com/

[48] Suite de código abierto de Software matemático como: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R, entre otros, basada en Python o directamente vía interfaces o Wrappers, http://www.sagemath.org/

[49] El economista, http://eleconomista.com.mx/tecnociencia/2013/01/22/clausuraran-negocios-mexico-uso-ilegal-Software

[50] PCworld, http://www.pcworld.com.mx/UNAM-y-BSA-promueven-el-uso-de-software-legal/

[51] L. F. Pavarino, A. Toselli; *Recent Developments in Domain Decomposition Methods*. Springer, 2003.

[52] M.B. Allen III, I. Herrera & G. F. Pinder; *Numerical Modeling in Science And Engineering.* John Wiley & Sons, Inc . 1988.

[53] R. L. Burden y J. D. Faires; *Análisis Numérico*. Math Learning, 7 ed. 2004.

[54] S. Friedberg, A. Insel, and L. Spence; *Linear Algebra*, 4th Edition, Prentice Hall, Inc. 2003.

[55] Y. Saad; *Iterative Methods for Sparse Linear Systems*. SIAM, 2 ed. 2000.

[56] Y. Skiba; *Métodos y Esquemas Numéricos, un Análisis Computacional*. UNAM, 2005.

[57] I. Herrera; *Un Análisis del Método de Gradiente Conjugado*. Comunicaciones Técnicas del Instituto de Geofísica, UNAM; Serie Investigación, No. 7, 1988.

[58] G. Herrera; *Análisis de Alternativas al Método de Gradiente Conjugado para Matrices no Simétricas*. Tesis de Licenciatura, Facultad de Ciencias, UNAM, 1989.

[59] M. Diaz, I. Herrera; *Desarrollo de Precondicionadores para los Procedimientos de Descomposición de Dominio*. Unidad Teórica C, Posgrado de Ciencias de la Tierra, 22 pags, 1997.

[60] A. Toselli, O. Widlund; *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2005.

[61] C.T. Keller, *Iterative Methods for Linear and Nonlinear Equations*, Societe for Industrial and Applied Mathematics, 1995.

[62] W. Gropp, E. Lusk, A. Skjellem, *Using MPI, Portable Parallel Programming Whit the Message Passing Interface*. Scientific and Engineering Computation Series, 2ed, 1999.

[63] I. Foster; *Designing and Building Parallel Programs*. Addison-Wesley Inc., Argonne National Laboratory, and the NSF, 2004.

[64] Jorge L. Ortega-Arjona, *Patterns for Parallel Software Design*, Wiley series in Software Design Patterns, 2010.

[65] S.J. Pennycook, S.D. Hammond, S.A. Jarvis and G.R. Mudalige, *Performance Analysis of a Hybrid MPI/CUDA Implementation of the NAS-LU Benchmark*. ACM SIGMETRICS Perform. Eval. Rev. 38 (4). ISSN 0163-5999, (2011).

[66] XMPI, *A Run/Debug GUI for MPI*.
http://www.lam-mpi.org/software/xmpi/

[67] VAMPIR, *Performance Optimization*.
http://www.vampir.eu/

[68] QEMU, http://wiki.qemu.org/Main_Page

[69] KVM, http://www.linux-kvm.org/page/Main_Page

[70] Oracle MV VirtualBox, https://www.virtualbox.org

[71] VMware, https://www.vmware.com

[72] Virtual PC,
https://www.microsoft.com/es-mx/download/details.aspx?id=3702

[73] Hyper-V,
https://msdn.microsoft.com/es-es/library/mt16937(v=ws.11).aspx

[74] Parallels, https://www.parallels.com

[75] Máquinas Virtuales, http://es.wikipedia.org/wiki/Máquina_virtual

[76] Algunos usos de máquinas Virtuales,
http://www.configurarequipos.com/doc747.html

Declaramos terminado este trabajo sufrido, ideado y llevado a cabo entre los años 2017 al 2026, aún y a pesar de impedimentos tales como: la mala suerte, la desventura, el infortunio, la incomprensión, la gripe, el COVID-19, la migraña, las horas de frío y calor, la tristeza, la desesperanza, el cansancio, el presente, el pasado y nuestro futuro, el que dirán, la vergüenza, nuestras propias incapacidades y limitaciones, nuestras aversiones, nuestros temores, nuestras dudas y en fin, todo aquello que pudiera ser tomado por nosotros, o por cualquiera, como obstáculo en este tiempo de mentiras, verdades, de incredulidad e ignorancia o negación de la existencia real y física de la mala fe.

Atentamente

Antonio Carrillo Ledesma
Karla Ivonne González Rosas