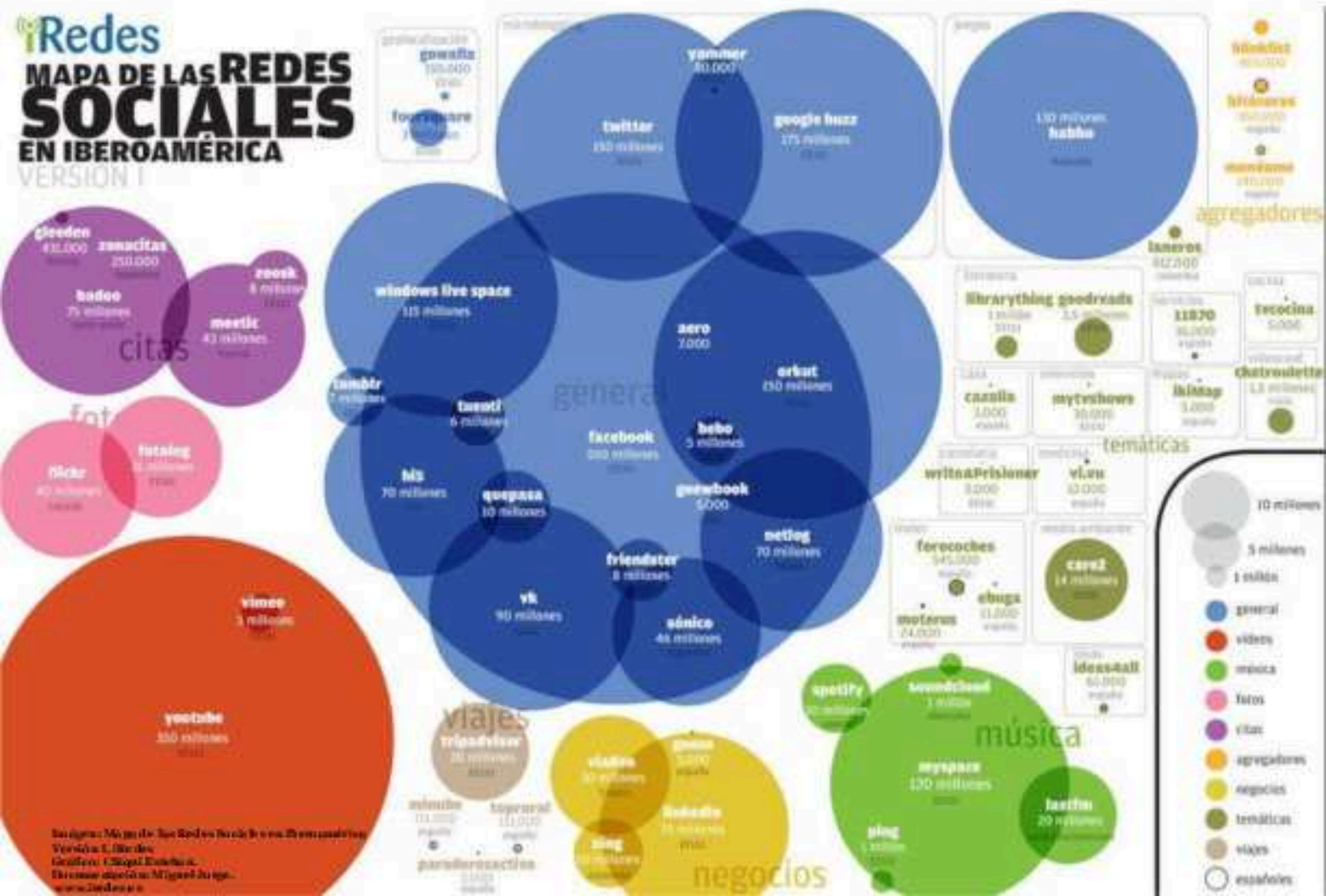
Manejo de Datos

Antonio Carrillo Ledesma
Karla Ivonne González Rosas



Introducción al Manejo de Datos

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

Una copia de este trabajo se puede descargar de la página:
http://132.248.182.159/acl/md/
http://132.248.182.159/acl/Textos/

Confinamiento 2020-2021, 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 **5**
1.1 Sobre los Ejemplos de este Trabajo . . . 6
1.2 Agradecimientos . . . 6

2 Sistemas Operativos **7**
2.1 Windows . . . 20
2.2 UNIX y BSD . . . 24
2.3 Mac OS . . . 26
2.4 GNU/Linux . . . 29
2.5 Android . . . 38
2.6 Chromebook y Chrome OS . . . 40
2.7 Otros Sistemas Operativos . . . 43

3 Seguridad, Privacidad y Vigilancia **48**
3.1 Amenazas a la Ciberseguridad . . . 62
3.2 Recomendaciones de Ciberseguridad . . . 67
3.2.1 Contraseñas Robustas . . . 68
3.2.2 Encriptar Discos o Archivos . . . 70
3.2.3 Sistema Operativo . . . 75
3.2.4 Máquinas Virtuales . . . 77
3.2.5 Navegación Segura . . . 79
3.2.6 Banca en Línea Segura . . . 82
3.2.7 Uso Seguro de las Herramientas de la Nube . . . 83
3.2.8 Teléfonos Inteligentes y Tabletas . . . 84
3.2.9 Redes Sociales . . . 88
3.2.10 Videoconferencias . . . 93
3.2.11 Seguridad de la Red . . . 95
3.2.12 Ataques con Técnicas de Inteligencia Social . . . 100
3.3 Meltdown, Spectre y lo que se Acumule . . . 101
3.4 Distribuciones de Linux «Seguras y para Penetración» . . . 107

4 Búsquedas Básicas en la Red **115**

5 Búsquedas Avanzadas en la Red **127**
5.1 Qué es un motor de búsqueda . . . 127
5.2 Tipos de buscadores . . . 127

5.3 Buscadores verticales . . . 128
5.4 Cuántos motores de búsqueda existen . . . 129
5.5 Principales motores de búsqueda . . . 130
5.6 Para qué sirven los motores de búsqueda . . . 131
5.7 Buscadores para obtener respuestas rápidas . . . 131
5.8 ¿Para qué saber sobre los motores de búsqueda? . . . 137

6 Lenguajes de Programación y Entornos de Desarrollo 143
6.1 Conceptos Transversales . . . 158
6.2 Algo de Programación . . . 164
6.3 Introducción a los Paradigmas de Programación . . . 176
6.4 Errores de Redondeo y de Aritmética en la Programación . . . 185
6.5 Lenguajes y Entornos de Desarrollo . . . 196
6.6 Documentación del Código Fuente . . . 206
6.6.1 Documentar en C, C++ y Java . . . 207
6.6.2 Documentar en Python . . . 216
6.7 Herramientas de Programación . . . 220
6.8 Programando Desde la Nube . . . 235

7 Programación Orientada a Objetos 239
7.1 Herencia y Composición . . . 245
7.2 Clase Abstracta . . . 258
7.3 Polimorfismo y Programación Genérica . . . 267
7.4 Excepciones . . . 272
7.5 Ejemplo de Altas, Bajas y Cambios . . . 279

8 Bases de Datos 293
8.1 Clasificación . . . 293
8.2 ¿Qué son las bases de datos SQL? . . . 295
8.2.1 Lenguaje de definición de datos (DDL) . . . 297
8.2.2 Lenguaje de manipulación de datos DML . . . 298
8.3 ¿Qué son las bases de datos NoSQL? . . . 307
8.3.1 SQL en comparación con NoSQL . . . 311
8.3.2 SQL en comparación con Terminología NoSQL . . . 313
8.4 Programando en Diferentes Manejadores de Bases de Datos . . . 318
8.4.1 SQLite . . . 319
8.4.2 MySQL/MariaDB . . . 321
8.4.3 PostgreSQL . . . 329

8.4.4 MongoDB . . . 332
8.5 Lenguajes de Programación y Bases de Datos SQL . . . 336
8.5.1 Java y el Acceso a Base de Datos Relacionales . . . 338
8.5.2 Python y el Acceso a Base de Datos Relacionales . . . 344
8.5.3 Pruebas de Penetración . . . 346
8.6 Programando desde la Web . . . 348
8.6.1 SQLite . . . 348
8.6.2 MySQL y MariaDB . . . 349
8.6.3 PostgreSQL . . . 350
8.6.4 MongoDB . . . 351

9 Intercambio de Información 354
9.1 Archivos de Formatos Abiertos . . . 360
9.2 Estándares Según el Tipo de Datos . . . 371
9.3 ¿Qué son los Metadatos? . . . 375

10 POO Java vs Python 377
10.1 Ejemplo de Clases Java vs Python . . . 377
10.2 Atributos de Objetos . . . 379
10.3 Métodos y Funciones . . . 390
10.4 Herencia y Polimorfismo . . . 391
10.5 Reflexión . . . 400

11 Apéndice A: Software Libre y Propietario 405
11.1 Software Propietario . . . 405
11.2 Software Libre . . . 407
11.3 Seguridad del Software . . . 409
11.4 Tipos de Licencias . . . 412
11.4.1 Nuevas Licencias para Responder a Nuevas Necesidades 418
11.5 Implicaciones Económico-Políticas del Software Libre . . . 420
11.5.1 Software Libre y la Piratería . . . 420
11.5.2 ¿Cuánto Cuesta el Software Libre? . . . 424
11.5.3 La Nube y el Código Abierto . . . 427
11.5.4 El Código Abierto como Base de la Competitividad . . . 430

12 Apéndice B: Errores en Programación 432
12.1 Errores en Java . . . 433
12.2 Errores en Pyhton . . . 438

13 Bibliografía **443**

1 Introducción

Con el aumento del acceso a internet por parte de personas y empresa 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, porqué 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 Sobre los Ejemplos de este Trabajo

La documentación y los diferentes ejemplos que se presentan en este trabajo, se encuentran disponibles en la página Web: Herramientas, para que puedan ser copiados desde el navegador y ser usados en la terminal de línea de comandos. 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.

1.2 Agradecimientos

Este proyecto fue posible gracias al apoyo recibido por la Facultad de Ciencias de la Universidad Nacional Autónoma de México (UNAM) y al tiempo robado a nuestras actividades académicas, principalmente durante el período de confinamiento de los años 2020 y 2021.

2 Sistemas Operativos

Actualmente tenemos 3 grandes sistemas operativos en el mercado:

- Windows
- Unix
- Linux

De los cuales, sus dignos representantes son: Windows 10[1], Mac OS X[2], iOS, Android, Chrome OS y GNU/Linux con todas sus diferentes distribuciones. Y sin temor a equivocarnos aseguramos que Android es la distribución de Linux más polular sobre la tierra 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 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

[1] Windows 11 será el reemplazo de Windows 10, disponible a partir diciembre de 2021.

[2] macOS Big Sur 11.0 será el reemplazo de Mac OS X, disponible a partir noviembre de 2020.

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? ¿Apple 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 OS X 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 OS X 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 OS X 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 OS X. 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) en el año 2021 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 OS X 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. Es de destacar que en el servicio de servidores Azure de Windows, el sistema predominante es Linux.

- En el segmento de supercomputadoras, Linux tiene la cuota más importante del mercado; es utilizado por los más poderosos 500 sistemas de supercómputo de alto desempeño del mundo[3].

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[4]

[3]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%).

[4]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

es mejor que Windows o Mac OS X 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:

- Windows NT 4.0: 12 millones de líneas
- Windows 2000: 29 millones de líneas (Pro $ 319 USD en 2000)
- Windows XP: 45 millones de líneas (Home $175 US y Pro $ 255 USD en 2001)
- Windows Vista: 50 millones de líneas (Home Premium $239.99 USD en 2007)
- Windows 7: 40 millones de líneas (Home Premium $199.99 USD en 2009)
- Windows 8: 60 millones de líneas (Home $119.99 US y Pro $ 199.99 USD en 2013)
- Windows 10: 60 millones de líneas sin Cortana y 65 millones de líneas con Cortana ($139.99 USD en 2019)
- Fedora 9: 204 millones de líneas
- Debian 7 ("wheezy"): 419 millones de líneas
- Debian 10 ("buster"): 1,077,110,982 líneas
- Debian Sid: 1,510,195,519 líneas (marzo 2021)
- Kernel Linux 4.12: 24 millones líneas
- Kernel Linux 5.8: 28 millones líneas

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.

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.

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 u 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 GNU/Linux Debian 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.

GNU/Linux Debian 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 esta 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 incluyen 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 susceptibles 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[5], el cual puede actualizarse y manipularse en cualquier dis-

[5]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.

tribución. 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 convergieran. 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 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, y finalmente la NT 10.0, también conocida como Windows 10.

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

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.

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[6] en cambio es bastante más cerrado, y está hecho por

[6]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, CSDVersion, ServicePackMajorVersion, BuildNumber | FL
```

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 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

o

[System.Environment]::OSVersion.Version

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 tienen 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, Microsoft está integrando el Kernel de Linux a su propio Kernel y esto ha permitido usar Linux dentro de Windows 10 de forma nativa gracias al llamado Windows Subsystem for Linux (WSL, WSL2 WSLg), lo cual ha permitido mejorar la estabilidad y desempeño de Windows.

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 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 sis-

tema solo se comunica con un disco duro y un adaptador de red, es probable que tenga una estabilidad impecable que supere 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 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 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[7] 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 11.1) 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.

La versión más reciente de Windows[8] es Windows 10 para equipos de escritorio, Windows Server 2019 para servidores y Windows 10 Mobile para dispositivos móviles. Windws 10 llegó de forma oficial y gratuita a usuarios con licencia genuina de Windows 7, Windows 8 y Windows 8.1 así como a Insiders, siendo la primera versión que busca la unificación de dispositivos (escritorio, portátiles, teléfonos inteligentes, tabletas y videoconsolas) bajo una experiencia común, con lo que se espera eliminar algunos problemas que se presentaron con Windows 8.1.

Windows 10 tiene al menos doce ediciones con diferente conjunto de carac-

[7] 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.

[8] El 24 de junio de 2021 se presento la nueva versión Windows 11, pero estará disponible para los usuarios a finales del año en una actualización gratuita para los usuarios de versiones anteriores.

terísticas 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.

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 10 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.

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.

Winget Windows Package Manager (*winget*) es el gestor de paquetes de Windoes 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 10 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[9]). Reiniciando el sistema, ya podemos usar distribuciones de Linux desde Microsoft Store.

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 anuncio 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 10. 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.

Windows 11 Microsoft presentó el 24 de junio del 2021 la siguiente generación de su sistema operativo, Windows 11, basado en la nube, 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

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

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 esta 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 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.

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 certifica-

dos 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 11.4).

Algunos sistemas operativos descendientes del sistema desarrollado por Berkeley son SunOS, FreeBSD, NetBSD, OpenBSD, DragonFlyBSD y Mac OS X. 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 OS X, 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 investigación. 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 Mac OS

Mac OS[10] (véase [5]) -del inglés Macintosh Operating System, en español Sistema Operativo Macintosh- es el nombre del sistema operativo propietario

[10] macOS Big Sur 11.0 es el reemplazo de Mac OS X disponible a partir de noviembre del 2020, con el abandono de los procesadores de INTEL para usar los procesadores M1 de ARM.

(véase apéndice 11.1) 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 Big Sur 11 es el reemplazo de Mac OS X, disponible a partir de noviembre del 2020 usando los procesadores M1 de ARM que han mostrado un gran desempeño en comparación con equipos INTEL 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/Linux (véase [4]) también conocido como Linux, es un sistema operativo libre (véase apéndice 11.2) 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 supercómputo de alto desempeño del mundo[11]-, 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

[11] 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.

la GPL. Los desarrolladores de ambas partes trabajaron para integrar componentes 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[12], 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.

[12] 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. 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. 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 llego 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.

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[13] para instalar, una de las más ampliamente usadas es Debian GNU/Linux[14] y sus derivadas como

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

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

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[15]:

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/

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

Ahora, Windows 10 Build 2020 con WSL2[16] (Windows Subsystem for Linux), tiene su propio Kernel de Linux que permite instalar de manera casi

[15] 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.

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

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 11.2). 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, 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[17]:

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 usados por las aplicaciones base. La arquitectura esta 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.

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 compila totalmente al momento de instalación de la aplicación.

[17] 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.

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 anuncio 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 Crome 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 pudiera 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 sincronizarán 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[18] 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.

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 esta 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. 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

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

para la versión doméstica. Muchas aplicaciones libres como Firefox, OpenOffice o VLC han sido portadas a este sistema operativo.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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, resultarán familiares a los usuarios de KDE Plasma.

13. 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.

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. 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.

¿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[19] activa a sus ciudadanos o empleados para monitorear ideas, discusiones o disensiones no deseadas. Los infractores son

[19] ¿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 independiente, como un juez.

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 protege 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[20]. Tener acceso a grandes

¿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.

[20]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 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.

Una opción para nuestra privacidad es usar buscadores que no dejan rastro de nuestras

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 de hacer cumplir la ley quieren conservar la capacidad para detectar e inves-

búsquedas como puede ser: DuckDuckGo.

tigar 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[21] 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 claves maestras serían buscadas por ciberdelincuentes y otros actores del

[21] 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.

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[22].

Algunos gobiernos y empresas tecnológicas quieren que aceptemos que no

[22]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 a otras agencias. Otros países como China o Rusia también vigilan internet con total desprecio a la intimidad de las personas.

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.

Las amenazas, 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 *Hackers*. 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.

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.

Seguridad TIC La seguridad TIC (Information Technology Security o IT Security) está de actualidad, ya que cualquier profesional o empresa que use las Tecnologías de la Información y de la Comunicación, debe situarse en la vanguardia de la tecnología.

Pero, ¿qué significa seguridad TIC?, ¿qué es? La seguridad TIC es la responsable de implantar las medidas de seguridad necesarias para procurar la protección de la información a través de diferentes tipos de tecnología. Así, es capaz de proteger los datos personales y de las empresas, tanto de formatos electrónicos o digitales, como en papel.

Seguramente, más de una vez han oído hablar de la palabra ciberseguridad como si fuera equivalente de seguridad TIC. Pero nada más lejos de la realidad: la ciberseguridad es un subtipo de la seguridad TIC. De este modo, la ciberseguridad, es la encargada de la protección de los datos de la organización respecto a los posibles ataques que tengan como origen internet.

Pero, de todas formas, la información y todos los datos personales y de

las empresas, no sólo se enfrentan a problemas de seguridad en internet, sino que también fuera de él. El modelo de defensa debe abarcar:

- Políticas, procedimientos y concientización
- Seguridad física
- Seguridad de perímetro
- Seguridad de la red
- Seguridad del equipo
- Seguridad de las aplicaciones
- Seguridad de los datos

Amenazas Físicas y sus Repercusiones para la Seguridad TIC Estas, pueden darse por diversos motivos. Así, es posible encontrar las generadas por las radiaciones electromagnéticas debidas a un teclado inalámbrico o el mismo acceso libre a la red Wi-Fi.

Dentro de estas redes Wi-Fi están los Sniffers, que no son más que sistemas que, por decirlo de alguna manera, recopilan cualquier dato que encuentren de los usuarios. Por lo tanto, son ideales para realizar auditorías en las propias redes, para corroborar y verificar el tráfico de una organización y además, monitorizar el comportamiento que tiene dentro de la propia red y sus movimientos.

No obstante, si un Hacker encuentra un acceso, con los Sniffer, podría apoderarse de todos los datos del usuario y contraseñas, por lo que el sistema no es demasiado adecuado ni fiable. Es por ello, que no es recomendable utilizar este sistema en las empresas, ya que puede suponer un grave problema para la seguridad de la información que se esté gestionando de esta manera.

Es usual encontrarse Sniffers, por ejemplo, en las universidades, con objeto de evitar que se conecten a la red Wi-Fi personas que no sean del propio centro y dañen el sistema.

Otra de las amenazas físicas que podemos encontrar y que en realidad, no puede evitarse, es la debida a los accidentes naturales. Como es evidente, se trata de que tenga lugar un accidente natural en el lugar donde se encuentre el Hardware, de manera que se dañe a consecuencia de un incendio, humedad, apagones, etc.

Amenazas Lógicas Este tipo de amenazas, son programas que han sido diseñados explícitamente para dañar el sistema. Básicamente, se aprovechan de cualquier vulnerabilidad, fallo o debilidad que exista en un sistema, para poder atacarlo y, finalmente, acceder a él.

Así, se pueden encontrar dos tipos:

- Intencionados: se trata de programas que tienen como objetivo hacer daño, como por ejemplo los Malware o Software malicioso, los Spyware, Jokes, Virus, Backdoors, Dialers, etc. entre otros.

- No intencionados: este tipo, tiene lugar cuando un programador comete alguna clase de error o fallo en la programación, dando lugar, por ejemplo a Bugs o Exploits.

Amenazas del Propio Usuario Este punto es el más sensible. Normalmente, en cualquier empresa, los empleados, o usuarios en sí son los eslabones más débiles en lo que respecta a la seguridad TIC.

Esto es, cualquier usuario de una empresa trabaja con datos, independientemente del permiso que tenga para gestionarlos, editarlos o verificarlos, pues se trata de una cadena. Así, encontramos usuarios que, por diversos motivos, son la fuente de los principales fallos de seguridad TIC que tienen lugar en una empresa, y son a los que se les suele denominar eslabones débiles.

Entonces, es fundamental que sepan la forma en la que hay que proceder, con objeto de poder evitar cualquier tratamiento erróneo de los datos.

Este tipo de usuarios suele actuar de dos formas:

- Activa: aquí, el usuario causa daño de manera consciente, bien sea por dañar o por sustraer información o datos específicos de la empresa. Como ejemplos, tenemos a los Hackers, antiguos empleados o piratas informáticos.

- Pasiva: cuando daña al sistema sin intención de hacerlo, por falta de formación, conocimiento o por querer acceder a una información a la que no tiene acceso. Por ello, es preciso establecer niveles de permiso para los accesos a los sistemas de gestión documental.

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[23]) 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 configurar 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

[23] 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.

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 adicional 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».

Formación en Seguridad de la Información Finalmente, los cursos de formación respecto a la seguridad de la información se tornan fundamentales para poder manejarse y recordar cualquier norma que sea imprescindible. Por ejemplo, la gestión segura de las contraseñas, del correo electrónico, la gestión de la documentación, etc.

Pero, de todas formas, la información y todos los datos personales y de las empresas, no sólo se enfrentan a problemas de seguridad en internet, sino que también fuera de él. Algunas de estas amenazas, son las que van a desarrollarse en este capítulo, pero si quiere conocer la norma ISO por excelencia para procurar la seguridad de la información en las empresas, le recomendamos que visite por ejemplo la página sobre la norma ISO 27001 o Cybersecurity Framework.

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.

3.1 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:

- 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 zombis (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: Este es uno de los más sofisticados y modernos ataques, ya que lo que hace es secuestrar datos (encriptándolos) y pedir un 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.

- 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) 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 este 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 zombis, 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 que troyanos, Rootkits, virus, gusanos y otros Malwares se propaguen e infecten más equipos.

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.

3.2 Recomendaciones de Ciberseguridad

Como usuarios de equipo de cómputo tenemos poca o nula capacitación en cuanto a las normas básicas de ciberseguridad, las más importantes son: usar contraseñas fuertes y diferentes para cada caso, mantener actualizado el sistema operativo, al navegar en internet hacerlo de forma segura, hacer uso seguro de las herramientas de la Nube, generar respaldos de nuestra información, hacer correcto uso de redes Wi-Fi, conocer y no caer en ataques de técnicas de inteligencia social, al hacer uso de videoconferencias conocer las políticas de privacidad y las medidas de seguridad, entre otras tantas cosas que el usuario actual de las Tecnologías de la Información y la Comunicación (TIC) debe dominar.

Algunas recomendaciones para reducir nuestra huella en internet:

- El uso de cuentas de correo alternativas nos permite registrarnos en diferentes sitios Web sin utilizar datos y/o cuentas personales.
- Usar cuentas de correo seguro, anónimo y bidireccional como el servicio de 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 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.

- 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 cononocimiento.

3.2.1 Contraseñas Robustas

Las contraseñas protegen la información que contienen los dispositivos y cuentas de los usuarios. No obstante, ante la cantidad de claves y combinaciones que cotidianamente se deben utilizar, la mayoría de las personas opta por contraseñas fáciles de recordar por la comodidad que esto implica, o bien, por la falta de conocimiento de lo fácil que puede ser para un ciberdelincuente obtenerlas. Para asegurar la efectividad de las contraseñas[24] y evitar el robo de éstas, es recomendable poner en práctica las siguientes acciones:

- Al generar las contraseñas de los dispositivos y cuentas se deben utilizar claves largas -mínimo 15 caracteres[25]- y únicas para cada caso, evitando utilizar la misma contraseña para diferentes dispositivos o cuentas.

- Se deben evitar las combinaciones sencillas como fechas de nacimiento, secuencias consecutivas, repeticiones de un mismo dígito o palabras simples como "password" o "contraseña".

[24] Para conocer la seguridad de una clave, podemos revisarla en:
https://howsecureismypassword.net/

[25] Solo para tener una idea del tiempo necesario para encontrar la clave de 13 caracteres aleatorios de símbolos, números, letras mayúsculas y minúsculas usando fuerza bruta en un solo core en el que se ejecuten más de mil millones de intentos por segundo se tardaría 9.6 millones años. Si se aumentan el número de cores los tiempos se acortaran de forma drástica, por ejemplo si se usa una supercomputadora que intente 100 millones de contraseñas por segundo bajaría de 9.6 millones a 96 años.
https://i.imgur.com/e3mGIFY.png

- La mayor longitud de la contraseña, así como la incorporación de mayúsculas, minúsculas, números y símbolos (#$,.-_%&*!?), contribuyen a que ésta sea más segura y difícil de vulnerar[26].
- No usar una contraseña única o similar para tus cuentas
- Se debe evitar escribir contraseñas en papeles o tener archivos con esa información que sean fácilmente accesibles para otros.
- Habilitar el doble factor de autenticación o verificación en dos pasos. Esta medida es una capa adicional de seguridad disponible para cada vez más servicios en la que, además de la contraseña, durante el inicio de sesión se solicita información sobre otro medio al que sólo el usuario autorizado tiene acceso (por ejemplo, verificación para entrar al correo electrónico mediante la recepción de un código vía SMS, llamada o mensaje de WhatsApp).
- Es importante no facilitar a nadie, aunque así lo solicite, por ningún medio, contraseñas y/o códigos para el inicio de sesión.
- Es recomendable cambiar con frecuencia las contraseñas a efecto de evitar accesos no autorizados.

[26] GnuPG y OpenSSl son herramientas en línea de comandos de seguridad en comunicaciones electrónicas en donde se utiliza criptografía de clave pública para que los usuarios puedan comunicarse de un modo seguro.

Para instalar el paquete GnuPG, usamos:

```
# apt install gnupg
```

para generar clave aleatoria de por ejemplo 32 caracteres, usamos:

```
$ gpg –gen-random –armor 1 32
```

Para instalar el paquete OpenSSL, usamos:

```
# apt install gnupg
```

para generar clave aleatoria de por ejemplo 32 caracteres, usamos:

```
$ openssl rand -base64 32
```

- Es recomendable usar un gestor de contraseñas[27] (como KeePassXC, KeePass, Bitwarden, Dashlane, NordPass, LastPass, Enpass, Keeper Password Manager, Password Safe, Password Gorilla, UPM, Buttercup, Myki, Pass, 1Password, etc) para generar y almacenar las claves de acceso estrictamente necesarias. Este es un programa de seguridad que almacena de forma segura todas las contraseñas en una caja fuerte virtual cifrada, los hay locales a tu dispositivo o en línea.

3.2.2 Encriptar Discos o Archivos

La encriptación de datos o cifrado de archivos es un procedimiento mediante el cual los discos, archivos, o cualquier tipo de documento, se vuelven completamente ilegibles gracias a un algoritmo que desordena sus componentes. Así, cualquier persona que no disponga de las claves correctas no podrá acceder a la información que contiene. A menudo nos referimos a los datos encriptados como texto cifrado. El propósito de la encriptación es proteger la confidencialidad de los datos digitales.

En el mundo criptográfico, cifrar es un procedimiento que utiliza un algoritmo matemático. Estos algoritmos modifican los datos, de manera que sólo sabiendo el mismo algoritmo se puede descifrar para saber qué es lo que se dice.

Como puede imaginarse, la protección de datos se ha convertido en una preocupación importante para las personas y compañías con el significativo crecimiento del volumen de datos y la sofisticación de la amenaza. A estos hechos, hay que añadir el riesgo que supone cada uno de nosotros lleva consigo grandes cantidades de datos confidenciales en sus ordenadores portátiles y dispositivos móviles.

[27] Un administrador de contraseñas tiene dos funciones principales: (1) almacenar contraseñas, y (2) generar contraseñas seguras y únicas. Esta aplicación es esencialmente como un libro digital que almacena todas sus contraseñas usando una "clave maestra". Al ingresar esta clave, se le otorga acceso al resto de las contraseñas. Por lo tanto, dicha clave o contraseña debe permanecer altamente protegida. Pero su segundo uso es mucho más práctico. El administrador de contraseñas genera automáticamente contraseñas que contienen una combinación compleja de mayúsculas y minúsculas, números, símbolos y caracteres especiales, que puede complicar el descifrado o la detección de la contraseña por parte de piratas informáticos. Al utilizar un administrador de contraseñas, se evita el error común de usar una sola contraseña en las diversas plataformas en línea, evitando así ataques de relleno de credenciales (credential stuffing).

¿Qué es encriptar? a pesar de los complicados cálculos matemáticos involucrados, la acción de encriptar datos no es difícil de entender. Simplemente, consiste en bloquear los datos mediante un código secreto que oculta su verdadero significado. De esta forma, si alguien accede a ellos, encontrará que la información carece de sentido. Y es que, para que los datos encriptados tengan sentido, se necesita la clave del código.

Existen dos tipos de cifrado: simétrico y asimétrico. En términos simples, el cifrado simétrico es la técnica más antigua y conocida. Utiliza una clave secreta para cifrar y descifrar los datos. Tanto el remitente como el receptor conocen la clave.

El cifrado asimétrico es un método más nuevo y también conocido como criptografía de clave pública. Utiliza dos claves en lugar de una: una pública y una privada. Las claves públicas permiten que cualquiera envíe información a otra persona, pero solo cada individuo conoce su clave privada.

Para qué sirve el cifrado de mensajes y otros datos sirve para hacer las comunicaciones más seguras, y lo mismo se puede decir a la hora de aplicarlo a internet. La primera funcionalidad para conseguirlo es la de la confidencialidad de los mensajes, ya que al no ir al descubierto, cuando tú le envías algo a otra persona, los algoritmos criptográficos de la aplicación ayudan a que no se pueda leer fácilmente si alguien lo intercepta en el camino.

Siendo la confidencialidad la primera de las ventajas que ofrece el cifrado de mensajes, la segunda podríamos decir que es la integridad. El encapsular un mensaje dentro de un sobre de cifrado, dicho así para hacerse una mejor imagen mental, ayuda que todo lo que haya cifrado se mantenga correcto y completo.

También hay algoritmos criptográficos que proporcionan mecanismos para verificar la identidad de la persona que envía un mensaje. Además, hay métodos de cifrado que también ayudan a vincular un documento o transacción a una persona o sistema de gestión concretas. Pero a nivel general, para lo que más se suele utilizar es para proteger las comunicaciones. Si tú envías un SMS se envía en texto plano, sin cifrar, y si un operador o alguna agencia interfiere el mensaje, puede leer todo lo que hay escrito. Sin embargo, aplicaciones como WhatsApp, Gmail o Telegram aplican cifrados para que esto no sea tan fácil.

¿Qué se debe encriptar? en términos generales, hay dos tipos de datos que se deben encriptar:

- Información de identificación personal. En este grupo se incluye cualquier tipo de información que otra persona pueda usar para identificar a un individuo de manera única. Esto incluye la licencia de conducir, número de seguridad social, etc. Los ladrones pueden usar esta información para robar una identidad, lo que les permite cometer delitos mayores, como solicitar tarjetas de crédito y préstamos a nombre de otra persona. Combatir esta clase de ataques requiere esfuerzos en muchos frentes. La información de identificación personal reside en los teléfonos, tabletas y computadoras portátiles , por lo que esos dispositivos y su almacenamiento se deben encriptar.

- Información confidencial de Negocios y Propiedad Intelectual. Los datos a los que los empleados acceden cada día acerca de los clientes, los planes para un nuevo producto o los datos sobre la próxima campaña de marketing entrarían en este grupo. De toda esta información podrían sacar provecho los competidores y, por tanto, puede convertirse en objetivo de piratas informáticos.

Ante las dificultades para decidir si se deben o no encriptar algunos datos, basta con preguntarse si se destruirían antes de tirarlos a la basura, caso de estar en formato papel o si, en el caso de que se filtraran por accidente causarían daño a los empleados o clientes. En ambos casos, si la respuesta es afirmativa, hay que encriptar.

Encriptar Discos Los discos duros externos y los dispositivos USB son la manera perfecta de poder llevarnos en nuestros viajes todos nuestros archivos importantes. Pero cuando estos archivos son personales y privados es posible que no quieras que nadie pueda acceder a ellos en caso de que se te pierda el USB o te lo quiten de alguna manera.

Por eso, todos los sistemas operativos suelen darte la opción de aplicarles un cifrado y protegerlos con una contraseña de tu elección, un proceso diferente al de protegerlos contra escritura. Vamos a ver como hacerlo paso a paso con Windows con una de sus herramientas nativas, y también te diremos brevemente cómo hacer lo mismo en MacOS, Android y GNU/Linux pero de una forma aún más sencilla.

Cifrar en Windows la manera de cifrar unidades externas en Windows tiene un nombre: BitLocker. Se trata de una herramienta desarrollada por Microsoft que suele venir preinstalada en casi todas las versiones de su sistema operativo, pero que de no ser así puedes conseguir en su página de descarga, donde se pueden descargar sus versiones de 32 y 64 bits.

Cifrar en MacOS en el caso de que tengas un Mac el proceso es aún más fácil, ya que lo único que tienes que hacer es tener la unidad formateada, conectarla al equipo y dar Click derecho sobre su icono cuando aparezca en el escritorio. En el menú desplegable sólo tendrás que elegir la opción: "Encriptar" para introducir con qué contraseñas lo quieres bloquear.

Cifrar en GNU/Linux en las distribuciones GNU/Linux por su parte también basta con dar Click derecho sobre la unidad, sólo que en este caso la opción a elegir es: "Formatear volumen". La clave aquí la tienes en el tipo de de formateo, ya que en él tendrás la opción Cifrado -LUKS o Cryptsetup-. De esta mane-ra, formatearás el disco duro aplicándole directamente un cifrado con una contraseña que elijas.

Cifrar en Android siguiendo estos pasos podemos encriptar el disco:

1.- Ve a: "Ajustes".

2.- Selecciona: "Seguridad".

3.- Pulsa en: "Cifrar teléfono" y configura una contraseña (véase sección 3.2.1).

4.- Espera tranquilamente a que el proceso acabe, puede tardar más de media hora fácilmente.

Es importante destacar que la única forma de quitar el encriptado es reseteando de fábrica el dispositivo, lo que significa el borrado total del contenido que en el haya almacenado. Si tienes una tarjeta microSD insertada, los datos dentro también serán encriptados y no podrás acceder a ellos en otro dispositivo.

Encriptar Archivos

En Windows la herramienta que trae Windows preinstalada (ediciones Education, Pro y Enterprise) es un buen comienzo para cifrar nuestros datos.

Para ello, deberemos seguir los siguientes pasos:

1. Haz Clic con el botón derecho en un archivo o carpeta (o mantenlo presionado) y selecciona: "Propiedades".
2. Selecciona el botón: "Avanzados" y haz Clic en la casilla de verificación: "Cifrar contenido" para proteger datos.
3. Pulsa el botón: "Aceptar" para cerrar la ventana: "Atributos avanzados" y a continuación, selecciona el botón: "Aplicar" y después: "Aceptar".

Una vez cifrada la información, solo podremos acceder si disponemos de la clave de cifrado correcta.

En GNU/Linux y MacOS es posible encriptar archivos usando GnuPG, esta es una herramienta en línea de comandos de seguridad en comunicaciones electrónicas en donde se utiliza criptografía simétrica y de clave pública para que los usuarios puedan comunicarse de un modo seguro[28]. Dentro de las funciones de GnuPG se incluyen generar un par de claves, intercambiar y comprobar la autenticidad de claves, cifrar y descifrar documentos, etc. Para instalar el paquete GnuPG, usamos:

```
# apt install gnupg
```

Es posible cifrar archivos usando sólo una clave (véase sección 3.2.1) -cifrado simétrico- para cifrar el archivo. La clave que se usa para el cifrado simétrico deriva de la contraseña dada en el momento de cifrar el documento. El cifrado simétrico es útil para asegurar archivos cuando no sea necesario dar la contraseña a otros. Un archivo puede ser cifrado con una clave simétrica usando la opción *–symmetric*, por ejemplo:

```
$ gpg –symmetric doc
```

o

```
$ gpg -c doc
```

y podemos descifrar usando:

```
$ gpg doc.gpg
```

[28] Otras opciones son: ccrypt y mcrypt.

3.2.3 Sistema Operativo

Mantener actualizados los sistemas operativos y las aplicaciones de los dispositivos, incluidas las computadoras personales (PC), los teléfonos inteligentes y las tabletas[29]. Estas actualizaciones incluyen cambios importantes que mejoran el rendimiento y la seguridad de los equipos; muchos de estos programas, incluso, se actualizan de manera automática. Además de:

- Cifrar el disco (véase sección 3.2.2) donde reside el sistema operativo y nuestros datos, sin este cifrado cualquier persona con acceso físico a nuestro equipo de cómputo, tableta, teléfono o dispositivo electrónico puede copiar sus contenidos incluso si no tienen la clave de acceso a ellos.

- Instalar solo aquellos programas y complementos que realmente uses, porque además de que pueden mermar la velocidad de tu equipo, es posible que estés cediendo más permisos de los que te imaginas.

- Activar funcionalidades de protección, como el cortafuegos[30] (Firewall), incorporadas en los sistemas operativos más comunes. Un cortafuegos es la primera línea de defensa ante un ataque a tu red desde internet y permite proteger el equipo de programas maliciosos o atacantes que intenten conectarse al equipo de forma remota. Además, permite establecer reglas para indicar qué conexiones de red se deben aceptar y cuáles no. Al mismo tiempo, admite el normal intercambio de datos entre la computadora y servicios verificados de internet.

[29] En el caso del sistema operativo Android hay una gran fragmentación de versiones en el mercado coexistiendo, esto debido a que los fabricantes y proveedores tienen un modelo de mercado que prioriza la venta de equipo nuevo y no la seguridad del usuario. Forzando a que las actualizaciones del sistema operativo dependan en gran medida del fabricante del dispositivo o del proveedor del servicio de telefonía, ya que ellos personalizan el sistema operativo, dificultando la actualización de los dispositivos. Lo que ocasiona que la gran mayoría de los dispositivos no reciban las actualizaciones de seguridad y nuevas versiones del sistema operativo publicadas por Android.

[30] Un cortafuegos es un sistema tecnológico cuya misión es la de dar paso o prohibir conexiones entre dispositivos informáticos, en base a una serie de reglas que el usuario puede establecer según las necesidades de cada dispositivo. Dichos sistemas puede ser implementados tanto en Hardware (físicos) como en Software (lógicos), o bien una combinación de ambos. En ambos casos, la tarea del cortafuegos es permanecer a la escucha de qué paquetes de información tratan de salir o entrar de nuestro equipo para desechar aquellos que no deseados.

- En caso de usar el sistema operativo Windows o Android se debe instalar un antivirus[31], estos programas ayudan a proteger los dispositivos contra la mayoría de los virus, gusanos, troyanos y otros tipos de Malware que pueden infectar a los dispositivos, por ello se recomienda:

  ◦ Instalar y mantener actualizados los antivirus, prefiriendo aquéllos que incorporan funcionalidades de protección contra Malware y cortafuegos (Firewall), también conocidos como "suites de seguridad".

  ◦ Evitar tener dos antivirus en un mismo dispositivo. Tener dos antivirus activos no significa mayor protección; de hecho, puede ocasionar diferentes problemas en el sistema. Un antivirus que esté trabajando se convertirá en un "Software malicioso" a los ojos del otro, el cual intentará bloquearlo y eliminarlo, y se corre el riesgo de afectar el desempeño del sistema por el consumo extra de recursos.

  ◦ Todas las instalaciones y actualizaciones de programas y aplicaciones deben hacerse desde el sitio Web oficial del fabricante o desde las tiendas oficiales de apps -verificando la identidad del autor de la aplicación-, evitando descargar e instalar aquéllas de dudosa procedencia.

  ◦ Deshabilitar la auto ejecución de memorias USB, para evitar que, por ese medio, se instalen programas maliciosos.

  ◦ En los sistemas operativos que lo soporten, habilitar la limpieza remota del dispositivo en caso de perdida o robo.

[31] Debido a que el Internet es una red abierta, cualquier computadora o dispositivo puede conectarse a este desde cualquier lugar. El uso de software antivirus sirve como un escáner inicial de cualquier actividad sospechosa o maliciosa a la que los usuarios están expuestos a través de las redes sociales. El software antivirus puede ayudar a supervisar la entrega de noticias y puede ofrecer un nivel adicional de protección en el evento que el usuario haga clic erróneamente en enlaces sospechosos que pueden contener spam y diferentes tipos de virus, como gusanos . Pero tener instalado un software antivirus no es una protección general ya que no puede atrapar todo el malware; el dispositivo aún puede estar infectado. Sin embargo, le agrega una capa de protección que puede ser beneficiosa para el usuario. Por eso es tan importante usar el sentido común también y desconfiar de cualquier mensaje que parezca extraño o sospechoso.

Respaldos a veces, no importa cuan cuidadoso sea uno, existe la posibilidad de perder tus datos o que te puedan Hackear. Si ese es el caso, a menudo la única forma en la que puedes recuperar tu información personal es restaurar desde un respaldo. Asegúrate de realizar respaldos periódicamente -preferentemente encriptados (véase sección 3.2.2)- de cualquier información importante y verifica que puedas restaurar tu información a partir de ellos. Es posible usar servicios en red como Google Drive y dispositivos USB para guardar en ellos los 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.
- Almacenar una copia de seguridad fuera de nuestra casa u oficina.

3.2.4 Máquinas Virtuales

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[32].

Una característica esencial de las máquinas virtuales[33] es que los procesos que ejecutan están limitados por los recursos y abstracciones proporcionados por ellas. Estos procesos no pueden escaparse de esta "computadora virtual". Uno de los usos más extendidos de las máquinas virtuales es ejecutar sistemas operativos nuevos u obsoletos adicionales a nuestro sistema habitual.

[32] Tal y como puede verse reflejado en la definición de máquina virtual, en este texto nos estamos focalizando en las máquinas virtuales de sistema. Existen otro tipo de máquinas virtuales, como por ejemplo las máquinas virtuales de proceso o los emuladores.

[33] Una máquina virtual dispone de todos los elementos de un equipo de cómputo real, de disco duro, memoria RAM, unidad de CD o DVD, tarjeta de red, tarjeta de vídeo, etc., pero a diferencia de un equipo de cómputo real estos elementos en vez de ser físicos son virtuales. Así, una vez instalado un sistema operativo en la máquina virtual, podemos usar el sistema operativo virtualizado del mismo modo que lo usaríamos si lo hubiéramos instalado en nuestro equipo de cómputo.

Algunas opciones de manejadores de máquinas virtuales son: Virtualbox, Vmware Workstation Player, Parallels, Windows Virtual PC, QEMU/KVM.

De esta forma podemos ejecutar uno o más sistemas operativos -Linux, Mac OS, Windows- desde nuestro sistema operativo habitual sin necesidad de instalarlo directamente en nuestra computadora y sin la preocupación de que se desconfigure el sistema operativo huésped o a las vulnerabilidades del sistema virtualizado, ya que podemos aislarlo para evitar que se dañe.

Las máquinas virtuales son una verdadera opción para coexistir simultáneamente diferentes versiones de sistemas operativos y en un mismo sistema máquinas virtuales ejecutando las diversas versiones de un mismo programa de cómputo, además se pueden configurar para que al momento de iniciarlas siempre se ejecuten a partir de una configuración e instalación base, de tal forma que al ser lanzadas, el usuario pueda instalar, configurar e inclusive dañar la máquina virtual, pero al reiniciarse la máquina virtual en una nueva sesión, se regresa a la configuración de la versión base, de esta forma no hay posibilidad de infección de virus entre diversos lanzamientos de sesiones de la máquina virtual.

El uso de las máquina virtual es variado, flexible y permite ser usado en diversos ámbitos de la educación, del desarrollo y prueba de programas de cómputo y en general, en Ciencias e Ingeniería. Algunas de las utilidades y beneficios que podemos sacar de una máquina virtual son los siguientes:

- Para aprender a instalar, probar diversas opciones de configuración y usar múltiples sistemas operativos. El proceso de instalación de la máquina virtual no requiere crear particiones adicionales en nuestro disco ni alterar la configuración de la máquina anfitriona.

- Para usar un Software que no esta disponible en nuestro sistema operativo habitual. Por ejemplo, si somos usuarios de Linux y queremos usar Photoshop, lo podemos hacer a través de una máquina virtual.

- En ocasiones tenemos que usar Software que únicamente se puede ejecutar en sistemas operativos obsoletos -Windows XP por ejemplo-, podemos crear una máquina virtual con dicho sistema y usar el Software de forma aislada sin preocuparnos de sus vulnerabilidades.

- Podemos experimentar en el sistema operativo que corre dentro de la máquina virtual haciendo cosas que no nos atreveríamos a realizar con nuestro sistema operativo habitual, como por ejemplo, instalar Software no seguro que consideramos sospechoso, etc.

- Si se hace el adecuado aislamiento de una máquina virtual en la que se instale alguna versión de Windows, esta puede ser inmune a los virus y no requiere el uso de antivirus.
- Si eres un desarrollador de Software puedes revisar si el programa que estas desarrollando funciona correctamente en varios sistemas operativos y/o navegadores de Web.
- Podemos usar las máquinas virtuales para hacer SandBox[34] con el fin de ejecutar aplicaciones maliciosas o abrir correos sospechosos en un ambiente controlado y seguro.
- Para probar versiones Alfa, Beta y Release Candidate de ciertos programas y sistemas operativos.
- Para probar multitud de programas en Windows y evitar que se ensucie el registro mediante las instalaciones y desinstalaciones de los programas.
- Podemos navegar en sitios Web maliciosos sin poner en peligro nuestro equipo, ya que podemos configurar la máquina virtual para que se pierdan los cambios al ser reiniciada.

3.2.5 Navegación Segura

Elegir un buen navegador es fundamental para garantizar que tus datos se mantengan a salvo. A efecto de promover la navegación segura en internet, se sugiere adoptar las siguientes recomendaciones:

- No guardar contraseñas en el navegador, aunque sea muy cómodo que el navegador tenga tus contraseñas guardadas para entrar rápidamente a todas las páginas que visitas con regularidad, lo cierto es que permitir que los navegadores Web guarden estos datos tan delicados nos pone en una situación de debilidad, ya que cualquier persona que consiga acceder a nuestro ordenador tendría acceso a nuestra información[35].

[34] Un sistema de aislamiento de procesos o entorno aislado, a menudo usando como medida de seguridad para ejecutar programas con seguridad y de manera separada del sistema anfitrión.

[35] Es recomendable usar un gestor de contraseñas para almacenar las claves de acceso.

- 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.

- Utilizar cifrado de principio a fin (End-toEnd) que utiliza una combinación de algoritmos para identificar al usuario y otros algoritmos que identifican a una conversación para cifrar mensajes y evita que terceros vean la conversación.

- También es posible usar el protocolo mensajería confidencial (Off-The-Record Messaging , OTR), es un protocolo criptográfico que proporciona un cifrado fuerte para conversaciones de mensajería instantánea. La principal motivación debajo de este protocolo es ocultar la identidad de los participantes en la conversación a la vez que se mantenía la confidencialidad de la propia conversación, como si fuera una conversación privada de la vida real.

- No dejar a la vista de otras personas información relevante, como aquélla sensible o claves de acceso, ni documentos o carpetas de trabajo.

- Mantener siempre la computadora, tableta, teléfono celular o cualquier otro dispositivo para el trabajo, en un lugar seguro y con contraseña (véase sección 3.2.1), a fin de restringir el acceso de personas no autorizadas.

- Al alejarse de los dispositivos de trabajo, es importante bloquear la sesión.

- Mantener cubierta la cámara Web cuando no la estas utilizando, para limitar el acceso que pudieran llegar a tener a ésta aplicaciones o programas no autorizados.

- Sé selectivo con las extensiones y complementos que instalas y usas, a cuantas menos aplicaciones le des acceso a tus datos mejor, ya que

pueden mermar el desempeño de tu navegador y es posible que estés cediendo más permisos de los que imaginas.

- Si tu organización facilita los recursos necesarios para el teletrabajo, es indispensable realizar un uso exclusivamente profesional de los medios proporcionados. No se recomienda, en ninguna circunstancia, manipularlos, modificar su configuración, o prestarlos a otras personas.

- Realizar copias de seguridad periódicas -preferentemente encriptadas (véase sección 3.2.2)- de la información para garantizar el acceso a la información almacenada, ya sea personal o de la organización. Así, en caso de que ocurra cualquier incidente de seguridad (robo, pérdida del dispositivo, o avería, etc.), se podrá mantener el acceso a la misma. Proteger con contraseña los dispositivos donde se almacene información (memorias USB o discos externos) para proteger la información de posibles accesos malintencionados y garantizar así su confidencialidad e integridad.

- Ingresar sólo a sitios Web confiables, escribiendo uno mismo la dirección de la página a la que se quiere acceder y evitando utilizar ligas proporcionadas por terceros.

- Conocer y aplicar las funcionalidades de "navegación privada" o "navegación segura", que impiden el almacenamiento del historial en el navegador, así como imágenes, nombres de usuario y contraseñas.

- Cuando se realicen transacciones o intercambio de información sensible, asegurarse de que la dirección de la página Web comience con: "https" (no "http"), lo que contribuye a mantener segura la información transmitida.

- Desactivar la compartición de tu ubicación geográfica, a menos que sea estrictamente necesario.

- Evitar el ingreso de información personal en formularios dudosos. Si te encuentras ante un formulario que solicita información sensible (por ejemplo, nombre de usuario y contraseña), es recomendable verificar la legitimidad del sitio antes de responder.

- Usar cuentas de correo seguro, anónimo y bidireccional como el servicio de 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.

- Usar buscadores que no dejan rastro de nuestras búsquedas en vez del buscador de Google, como pueden ser: DuckDuckGo, Startpage, Yippy, Gibiru, Disconnect Search, Lukol, Metager, entre otros.

- Al terminar de navegar en internet, es importante cerrar la sesión y limpiar el Cache, sobre todo si se utiliza un equipo compartido, para evitar que otras personas tengan acceso a cuentas e información privada.

- Usar el navegador Tor, este provee una red abierta y distribuida que te ayuda a defenderte de una forma de vigilancia en la red que amenaza tu libertad y privacidad, tus actividades personales y relaciones, además de la seguridad gubernamental. También, te protege redirigiendo tus comunicaciones alrededor de una red distribuida de retransmisores realizados por voluntarios alrededor del mundo: lo cual previene que alguien observe tus comunicaciones a partir de los sitios que visitas, también evita que los sitios que navegas obtengan tu ubicación física.

3.2.6 Banca en Línea Segura

El uso de la banca en línea es una gran alternativa para todos los usuarios de dichos servicios, pero el acceso a las aplicaciones bancarias debe ser usando un navegador Web seguro o por medio de las aplicaciones del banco desarrolladas para dispositivos móviles, en ambos casos es necesario usarlas en un dispositivo de confianza, encriptado, con contraseña segura para el acceso (véase sección 3.2.1), el sistema operativo actualizado, usando las medidas básicas de protección y con un mínimo de aplicaciones instaladas para evitar Malware, además se sugiere adoptar las siguientes recomendaciones:

- Solicitar al banco que nos envíe alertas de todas las transacciones de nuestras cuentas y mantengamos estas monitoreadas de actividad no autorizada.

- Cuando se deba tener acceso a banca en línea es necesario conectarse usando redes seguras en las que uno confié, nunca desde redes públicas

- Cuando se deba tener acceso a banca en línea desde la Web es recomendable usar el modo Privado o Incógnito y no guardar contraseñas en el navegador.
- Ingresar sólo a sitios Web confiables, escribiendo uno mismo la dirección de la página a la que se quiere acceder y evitando utilizar ligas proporcionadas por terceros y asegurarse de que la dirección de la página Web comience con: "https" (no "http").
- Usar en la medida de posible contraseñas seguras del mayor tamaño permitido por la aplicación bancaria.
- No dejar a la vista de otras personas información relevante, como aquélla sensible o claves de acceso, ni documentos o carpetas de trabajo.
- Mantener siempre la computadora, tableta o teléfono inteligente usado para acceder a la banca en línea en un lugar seguro y con contraseña segura, a fin de restringir el acceso de personas no autorizadas.
- Al alejarse de los dispositivos de trabajo, es importante cerrar la sesión bancaria y bloquear la sesión del usuario.
- Ser selectivo con las extensiones y complementos que instalas y usas, a cuantas menos aplicaciones le des acceso a tus datos mejor, ya que pueden mermar el desempeño de tu navegador y es posible que estés cediendo más permisos de los que imaginas.
- Al terminar de navegar en internet, es importante cerrar la sesión y limpiar el Cache, sobre todo si se utiliza un equipo compartido, para evitar que otras personas tengan acceso a nuestras cuentas bancarias.

3.2.7 Uso Seguro de las Herramientas de la Nube

La nube permite almacenar y administrar datos, así como ejecutar aplicaciones en línea, entre muchas otras funciones. Con relación al almacenamiento, la nube permite acceder a archivos y datos desde cualquier dispositivo conectado a internet; es decir, la información está disponible en cualquier lugar en el que te encuentres y siempre que la necesites.

Para hacer uso de los servicios de la nube de manera segura y evitar el robo o mala utilización de la información almacenada, es conveniente tener en mente las siguientes recomendaciones:

- Tener conocimiento de las condiciones de uso y las políticas de privacidad antes de utilizar cualquier servicio en la nube.
- Utilizar servicios de almacenamiento que cuenten con cifrado "https" y certificado de seguridad. Esto lo puedes verificar en la barra de direcciones de tu navegador de internet.
- No subir a la nube información sensible con acceso público o abierto. Se recomienda utilizar herramientas de cifrado, como es el uso de carpetas con contraseña y acceso restringido.
- Verificar periódicamente los archivos y carpetas que tenemos compartidos desde nuestra cuenta, a fin de deshabilitar los enlaces y acceso de terceros que ya no sean necesarios.
- Utilizar contraseñas robustas para acceder al servicio y, preferentemente, activar el doble factor de autenticación o verificación en dos pasos.
- Realizar periódicamente un respaldo de la información almacenada en la nube en otro tipo de dispositivo, por ejemplo, en un disco duro externo debidamente protegido por contraseña (véase sección 3.2.2). De esa manera, se mantiene el acceso a la información en caso de cualquier contratiempo, como una conexión limitada a internet.
- Cerrar la sesión de la nube al concluir las actividades que se estén realizando.

3.2.8 Teléfonos Inteligentes y Tabletas

Estos dispositivos están omnipresentes en nuestra vida, además pueden almacenar mucha más información de la que nos imaginamos y muchos de nosotros los usamos para acceder a casi toda nuestra información digital desde ellos. Por tanto, necesitan protegerse activamente para evitar que el contenido que almacenan y al que tienen acceso termine en manos de terceras personas. En ellos se guarda todo tipo de datos personales, mensajes, imágenes y vídeos en los que aparecen los usuarios, amigos y otras personas. A veces incluso se puede haber recibido o grabado contenido íntimo o de connotación sexual[36].

[36] En el caso de los adolescentes no siempre son conscientes del riesgo que supone producirlo o guardarlo, Pero, en cualquier caso, requiere una protección especial, y como

No olvidemos que cualquier método de almacenaje de información digital puede sufrir filtraciones o ataques informáticos, pero los dispositivos móviles son aún más vulnerables. Pueden extraviarse, también pueden ser robados en diversas situaciones o, simplemente, ser interceptados en un momento de despiste. Por ello estos son algunos de los hábitos que debemos practicar a diario para mejorar la protección del dispositivo y nuestros datos:

- Todos los dispositivos móviles y tarjetas de respaldo extraíbles deben ser cifrados y se debe usar una contraseña robusta para el acceso al mismo (véase sección 3.2.1), es decir, utilizar una combinación compleja -de al menos 15 caracteres- de letras mayúsculas, minúsculas, números y símbolos es siempre la mejor opción. Además, es imprescindible no confiarla a sus amistades y cambiarla siempre que tengan dudas de su fiabilidad.

- Es fundamental que se comprenda que toda la información que guardan en el dispositivo es susceptible de caer en manos de terceras personas. Por eso, lo ideal es que no guarden imágenes, vídeos o datos de carácter sensible. Es preferible que revisen periódicamente sus archivos, eliminen de forma eficaz aquello de lo que puedan prescindir y, en el caso de querer guardar algunos contenidos, los almacenen en otros dispositivos más seguros, como un disco duro externo encriptado (véase sección 3.2.2) que mantengan en casa. Cuanta menos información lleven en su dispositivo, menor riesgo.

- Cualquier dispositivo ofrece un sistema de bloqueo para que nadie que no sea el usuario pueda acceder a él. En el caso de móviles y tabletas, tienen diferentes alternativas, como usar su huella dactilar, un sistema de reconocimiento facial, un código PIN, una contraseña o un patrón. Cualquier opción es buena, pero hay que tener en cuenta que un patrón o un código PIN siempre son más fáciles de observar y memorizar por otras personas. Obviamente, esta función pierde su eficacia si no se acostumbran a bloquear el dispositivo siempre que no lo estén utilizando.

adultos podemos mostrarles la importancia de proteger estos contenidos.

Este tipo de contenido es extremadamente sensible y, de hecho, cuando se trata de imágenes o vídeos en los que aparecen menores de edad ni siquiera es legal almacenarlo en ningún tipo de formato.

- Algunos servicios de internet, como cuentas de correo o redes sociales, ofrecen un sistema combinado de autenticación. De esta forma, además de solicitar la contraseña para acceder, utilizan un segundo paso para comprobar la identidad del usuario. Puede ser un código que llegue a su teléfono móvil, por ejemplo, y es muy útil contra el robo de contraseñas. Para ponernos en situación, si alguien averigua la clave de acceso a su cuenta de Instagram no podrá acceder porque también necesitará disponer del dispositivo para ver el código de verificación.

- Siempre es una buena costumbre cerrar la sesión de cada servicio al concluir su uso. Por ejemplo, si se entra en redes sociales o en su cuenta de correo en el ordenador público, es fundamental que cierren la sesión y limpiar el Cache antes de irse, comprobando que su contraseña no ha quedado guardada.

- Hoy en día todos los servicios digitales incorporan opciones de seguridad y privacidad que se han diseñado para proteger nuestra información. Una buena práctica es revisar la configuración de cada aplicación o servicio que se utilice, personalizando todas aquellas funciones que les sean de utilidad. Así, se pueden establecer cuentas de redes sociales como privadas o revisar los permisos que otorgan a cada aplicación.

- Existen muchas más funciones que pueden ser útiles para salvaguardar todas esos datos, imágenes y vídeos que se guardan en dispositivos. Todo es cuestión de investigar y buscar soluciones siempre que sientan que su información es vulnerable. Por ejemplo, pueden bloquear la galería de su móvil o el explorador de archivos, de forma que sea necesario utilizar su huella para acceder. También pueden crear listas de amistades en las redes sociales para que las publicaciones sean más personales y no lleguen a todos los seguidores.

- Activar la función de búsqueda y bloqueo de dispositivo frente a pérdidas o robos. Cada vez hay más opciones de seguridad disponibles, solo hay que motivar su uso en la prevención de riesgos.

- Activar el Wi-Fi y Bluetooth sólo cuando se usen, de esta forma se minimiza la posibilidad que un usuario mal intencionado intente conectarse a nuestro dispositivo.

- Es posible poner un código de acceso en su tarjeta SIM, esto puede protegerla de la suplantación de SIM. La configuración de este código se puede hacer en un iPhone yendo a Configuración > Celular > PIN de SIM. Ingrese su PIN existente para habilitar el bloqueo. Los usuarios de Android pueden ir a Configuración > Seguridad > Bloqueo de tarjeta SIM. Aquí puede habilitar la opción para bloquear su SIM.

- Se debe evitar el uso de puertos de carga USB no confiables, como los de energía públicos, ya que ellos se pueden Hackear, y el ciberdelincuente puede instalar, grabar datos, tomar video o sacar datos de nuestros dispositivos móviles. 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.

Escaneo de Códigos QR Los códigos QR cada vez están más presentes en nuestras vidas, sin embargo, éstos no están exentos de riesgos de seguridad, riesgos de los que muchos usuarios no son conscientes.

Los QR son Menos Transparentes que una URL comencemos por lo más obvio, es relativamente fácil distinguir una URL maliciosa de una legítima, pero no ocurre lo mismo con los QR, lo que facilita que seamos remitidos a Webs de Phising o con Malware. Alex Mosher, vicepresidente de soluciones globales de MobileIron, afirma que espera que "pronto veamos una avalancha de ataques a través de códigos QR" dirigidos a robar datos de nuestros dispositivos móviles o tomar nuestras credenciales y datos personales.

No Entendemos qué Puede Abrir un Código QR otro problema es que los usuarios no terminan de comprender qué es lo que pueden hacer los códigos QR, mientras que el 67% piensa (correctamente) que pueden abrir URLs, sólo una tercera parte de ellos es consciente de que eso permite también realizar toda una serie de acciones más allá de abrir una página Web, como:

- Escribir un correo electrónico o un mensaje SMS.

- Añadir un contacto a tu agenda.

- Añadir un evento a tu calendario.

- Añadir credenciales de acceso a un red WiFi.
- Realizar un pago online.
- Iniciar una llamada telefónica.
- Enviar información sobre tu localización a una app.
- Empezar a seguir a alguien en redes sociales.

Aunque todas éstas actividades son en cierto modo lícitas, pueden engendrar una gran brecha de seguridad en nuestros dispositivos móviles.

3.2.9 Redes Sociales

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 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:

- La gente tiende a inundar de información mientras completa los perfiles en redes sociales. Hay que tener en cuenta que no estamos ante una entrevista de trabajo o que sólo nuestros amigos lo van a leer. No necesitamos informar de todo y a todos, y menos con información sensible. Los expertos creen que revelar demasiada información personal crea una grieta de seguridad. Los atacantes usan información personal como la fecha de nacimiento, el nombre de los miembros de la familia, los números de teléfono, la dirección física, etc. Todo ello para ejecutar múltiples iteraciones para descifrar las contraseñas. 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. No existe una fórmula para decidir qué es seguro publicar en público y qué no, pero hay que usar el sentido común. Algunas cosas deben mantenerse entre amigos o familiares. Publicar cierta información en muros públicos pone en riesgo no solo a nosotros mismos sino a la privacidad de nuestros amigos.

- Se dice que una foto vale más que mil palabras. En el caso de las redes sociales incluso más. Además si subimos imágenes con una leyenda informativa, nos pone aún más en riesgo. Estamos dando información sobre nuestros hábitos, nuestros movimientos, a posibles atacantes. Es un aspecto importante para mantener la seguridad en redes sociales.

- Hay que asegurarse de que nuestro contenido en las redes sociales sea visible solo para amigos y familiares. Podemos crear listas de tipos de amigos y personalizar la visibilidad de cada publicación. No todo debe estar visible para esa persona que trabajó con nosotros durante unas semanas hace unos años, por poner un ejemplo. Puede que haya cosas que no queramos compartir con todos los contactos.

- El GPS es una característica común en todos los teléfonos inteligentes en la actualidad. Además de ayudarnos mientras pasamos por una ciudad desconocida, el GPS también se utiliza para el geoetiquetado. Significa que podemos adjuntar información de ubicación a cualquier contenido multimedia que enviemos o recibamos. Twitter, Facebook e Instagram usan esta característica extensivamente para ayudar a los usuarios a marcar la ubicación donde se hizo una foto, y ayudar a que el perfil sea más «social». Los atacantes cibernéticos pueden interpretar fácilmente información como nuestro estado económico, estilo de vida, lugares frecuentes y la rutina diaria a través de los medios con etiquetas geográficas.

- Una de las maneras más elementales de fortalecer nuestra cuenta de redes sociales es crear una contraseña robusta (véase sección 3.2.1), es decir, utilizar una combinación compleja -de al menos 15 caracteres- de letras mayúsculas, minúsculas, números y símbolos es siempre la mejor opción. Además, es imprescindible no confiarla a sus amistades y cambiarla siempre que tengan dudas de su fiabilidad.

- Guardar contraseñas en el dispositivo es una práctica muy habitual. Simplemente la abrimos y ya está, sin necesidad de tener que poner la clave cada vez que la usamos. Sin embargo esto no es un buen método. Si alguien accede físicamente a nuestro teléfono o al ordenador, podría entrar sin problemas a nuestras cuentas. Lo más recomendable es no guardar contraseñas.

- En las redes sociales debemos de tener cuidado con los Links que nos llegan. Incluso si éstos proceden de algún amigo. Hay que fijarse bien en el enlace. En lo que contiene. Muchas veces puede ser una trampa. Nuestro contacto lo hace de forma inconsciente, claro. Pero se trata realmente de un Malware.

- Algo que no puede faltar es contar con programas y herramientas de seguridad. Pero esto no se extiende únicamente a la seguridad en redes sociales, sino a cualquier acción que hagamos en internet. Siempre tenemos que tener algún antivirus instalado. Así podremos hacer frente a posibles amenazas que pongan en riesgo el buen funcionamiento de nuestro equipo.

- 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. También es muy frecuente subir fotografías personales donde el usuario es perfectamente identificable. 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 las cuentas de todos sus amigos.

- Es muy común que de forma periódica aparezcan solicitudes de amistad en el perfil de cada usuario, por parte de personas que en realidad no se conocen. Muchas veces el acto de aceptar una de esas solicitudes es tan automático que no se vigila si se está admitiendo a una persona conocida o no. Los amigos se pueden contar con los dedos de las manos (con una suele ser suficiente) mientras que lo que encontramos en las redes sociales son conocidos con los que debemos mantener la correspondiente distancia.

- Una red social es un lugar muy atractivo para estar; mientras más amigos se tienen, más novedades aparecen de forma constante en la página de cada uno, creándose un ciclo de interacciones que no tiene un

final concreto. Eso hace que algunas veces el usuario sienta la necesidad de estar siempre pendiente y atento a su red social, dándose casos esporádicos y extremos de dependencia total a su red.

- Quizás lo más importante de estos consejos para mantener la seguridad en redes sociales es el sentido común. La gran mayoría del Malware necesita la interacción del usuario para ejecutarse. Debemos estar siempre alerta y usar el sentido común.

Acuerdos de Privacidad La lectura de los acuerdos de privacidad 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. Estas son algunas recomendaciones generales a tener en cuenta para administrar mejor la configuración de privacidad en una cuenta de redes sociales:

- Selecciona quién tiene acceso de visualización a la actividad en redes sociales pasada, presente y futura (por ejemplo, Tuits, Me gusta, etc.).
- Revisa qué contenido se le puede agregar (es decir, etiquetar) a una cuenta cuando otras personas suban o publiquen contenido.
- Revisa, comprende y define la audiencia con la que se puede compartir contenido.
- Revisa, comprende y determina los formularios a través de los cuales otros usuarios pueden encontrar y conectarse con tu cuenta.
- Revisa, comprende y determina la cantidad de información personal que se incluye al bloguear o publicar información en línea.
- Monitorea periódicamente la seguridad y la información de inicio de sesión de las cuentas y revisa la probabilidad de que se esté realizandoalguna actividad sospechosa.
- Selecciona una copia de seguridad confiable, que pueda detectar o recibir alertas de alguna actividad sospechosa.
- Monitorea si y qué aplicaciones pueden acceder a alguno de tus datos y/o información en redes sociales, especialmente en segundo plano.

- Ten en cuenta las implicaciones de incluir la ubicación al publicar contenido en línea.
- Configura una autenticación de dos factores para iniciar sesión.
- Revisa la política de privacidad de la plataforma para saber qué datos recopilan los servicios, con quién se comparten y selecciona tus preferencias en estos dos temas.

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 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 les 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.

3.2.10 Videoconferencias

También conocidas como teleconferencias o videollamadas[37] se han convertido en una herramienta indispensable para el trabajo, estudio y placer de muchas personas, e incluso, el medio para dar continuidad a asuntos laborales, la vida cotidiana y la comunicación con familiares y amigos. Lo novedoso de estos servicios para muchos usuarios y la aparición de algunas vulnerabilidades en ciertas plataformas[38], supone para los ciberdelincuentes

[37] Algunas opciones son: Google Meet, Microsoft Teams, Jitsi Meet, Jami, Nextcloud Talk, Riot.im, BigBlueButton, Wire, Skype, Hangouts, WhatsApp, FaceTime, Google Duo, Discord, Gruveo, Instagram, Snapchat, Meet, Line, Blue Jeans, Teams, Webex, Facebook Messenger, ZooRoom, Zoom, Signal, Viber, WeChat, etc.

Si bien el equipo mínimo necesario para participar en videoconferencias es un equipo Atom o Celeron con 2 GB de RAM, casi todas las plataformas de videoconferencia recomiendan un equipo de cómputo como el i3 de Intel con 2GB de RAM -o su equivalente en AMD- para ser partícipe, en el caso de ser el anfitrión de la reunión se recomienda un equipo i5 con 4 GB de RAM -o su equivalente en AMD-. En cualquier caso, es común hacer un uso intenso de CPU, generando más calor que en su uso cotidiano. Esto ocasionará que los ventiladores del equipo trabajen a toda su capacidad para disipar el calor generado. Por ello es recomendable que el equipo esté conectado a la corriente eléctrica, cerrar las aplicaciones innecesarias y que el equipo esté bien ventilado, además de no maximizar la ventana de visualización de la videoconferencia pues esto generará mayor uso de CPU.

En el caso de usar dispositivos móviles como tabletas o teléfonos inteligentes para ser partícipe en videoconferencias, es recomendable instalar las aplicaciones de videoconferencia respectiva y tratar de no hacerlo mediante el navegador, para así optimizar el uso de Hardware y batería.

[38] Una de las aplicaciones más usadas es Zoom, la cual ha tenido múltiples problemas de seguridad y privacidad.

la oportunidad para el acceso no autorizado a información, robo de credenciales y acceso a los distintos recursos del dispositivo (como micrófono, cámara, etc.).

Por lo anterior, es necesario promover la adecuada protección de los usuarios para evitar incidentes al usar estos servicios tales como:

- Informarse sobre las políticas de privacidad y las medidas de seguridad que implementa el servicio que se desea utilizar.
- Descargar e instalar la aplicación correspondiente desde la página Web oficial del desarrollador o desde las tiendas oficiales de apps.
- Mantener actualizada la aplicación que se utilice, pues es a través de este proceso que se puede asegurar que las vulnerabilidades detectadas y corregidas por el desarrollador se están implementando.
- Al organizar una videoconferencia se recomienda tener en cuenta:

  ◦ En el caso de reuniones privadas, compartir el enlace directamente con los participantes, haciendo uso de las funciones para compartir de las propias aplicaciones, y evitando el uso de redes sociales o canales de comunicación abiertos que podrían promover accesos no deseados.

  ◦ Proteger la conferencia con una contraseña robusta, para restringir el acceso a ésta a personas no autorizadas.

  ◦ Si la plataforma la incorpora, activar la funcionalidad que permite al organizador verificar y, en su caso, aprobar el acceso de los participantes que deseen acceder a la videoconferencia.

- Los participantes en videoconferencia deben evitar compartir su escritorio de forma predeterminada ya que esto podría provocar fugas de información.
- Se debe cuidar el encendido del micrófono y la cámara de video para evitar situaciones incómodas o embarazosas.
- Si la videoconferencia es grabada, el organizador debe comunicarlo a los participantes.

3.2.11 Seguridad de la Red

Una parte importante del trabajo y estudio a distancia es la aplicación de medidas de seguridad de las redes. Es cada vez más común que los usuarios tengan acceso a un Ruteador inalámbrico (Wi-Fi) para conectar sus dispositivos a internet sin necesidad de cables.

Redes Wi-Fi casi todas las redes domesticas comienzan con una red inalámbricas (Wi-Fi), la mayoría de las redes inalámbricas están controladas por el Router de internet o punto de acceso inalámbrico separado y dedicado. Ambas trabajan de la misma manera: transmitiendo señales inalámbricas que permiten que los dispositivos de tu casa se conecten a internet.

Para evitar que usuarios no autorizados se conecten de forma inalámbrica al Ruteador[39] y tengan la posibilidad de acceder a la conexión, e incluso al resto de los dispositivos conectados y a la información que se transmite, es importante mantenerlo actualizado y asegurar que la red Wi-Fi cuente con contraseña robusta (véase sección 3.2.1) que el usuario debe introducir al conectar por primera vez un dispositivo.

Los Ruteadores ofrecen varios tipos de contraseñas y cifrados (que codifican los datos del usuario, usando un valor o clave secreta y los hace incomprensibles para terceros), como los siguientes:

- Las redes sin cifrado, o abiertas, son aquéllas que no tienen ninguna contraseña o cifrado y permiten a cualquier usuario conectarse. Una red con estas características no es recomendable su uso.

- El cifrado Wired Equivalent Privacy (WEP, por sus siglas en inglés) es considerado, hoy en día, un sistema poco seguro y no se aconseja su uso ya que, con las herramientas y conocimientos adecuados, se puede llegar a conseguir la clave de acceso a la red Wi-Fi en pocos minutos.

- El cifrado Wi-Fi Protected Access (WPA, por sus siglas en inglés), específicamente en su versión 2 (WPA2) o más actualizada, es considerado seguro y se recomienda comprobar que esté habilitado como parte de las medidas de seguridad de la red. Para comprobarlo, es necesario entrar desde la computadora a las propiedades de la red, para

[39] Si no se estas seguro de como configurar tu dispositivo de acceso a internet, consulta el sitio Web de tu proveedor de servicios de internet o consulta el sitio Web del proveedor para tu Router o punto de acceso inalámbrico.

ver el tipo de seguridad de la conexión. Se recomienda tener habilitada alguna de las variantes de WPA2, al menos. Puedes solicitar apoyo a tu proveedor de servicios de internet para más orientación.

- Se recomienda cambiar las contraseñas predeterminadas en el Ruteador por unas de elección del usuario y distinta a la utilizada para conectarse a la red inalámbrica, utilizando contraseñas robustas, se recomienda que incluya mayúsculas, minúsculas, números y símbolos. Cuanto mayor sea la longitud de la contraseña, más difícil será que un atacante pueda descubrirla.

- Es importante evitar compartir la clave de la red Wi-Fi con otras personas, pues quien tenga acceso a tu red inalámbrica podría tener acceso a todos los dispositivos conectados a ella.

- Es importante evitar la conexión a redes Wi-Fi públicas abiertas (o Hotspots Wi-Fi). Estas redes son totalmente inseguras ya que permiten que cualquier dispositivo se conecte al Ruteador sin ningún tipo de seguridad, por lo que cualquier usuario podría capturar la información que se transmita a través de dicha conexión.

- De ser posible usar filtrado por MAC para asegurar que cada dispositivo que acceda a la red sea el autorizado por el administrador de la misma.

VPN o red privada virtual (Virtual Private Network, VPN) es un servicio mediante el cual se establece una conexión segura a través de internet, entre los usuarios y los servicios o páginas Web de internet a los que éstos acceden.

Si imaginamos el internet como un río en el que fluye el agua (datos e información), la VPN es un tubo, sumergido en el río, que impide ver todo lo que pasa dentro de él, debido a que la conexión entre los dispositivos y el servidor VPN siempre está cifrada (protegida). De esa manera, si alguien interceptara tus comunicaciones, sería incapaz de interpretar la información transmitida.

En algunas ocasiones, las empresas ponen a disposición de sus empleados acceso a través de VPN; de no ser éste el caso o para añadir una capa extra de seguridad a tus comunicaciones personales, las VPN pueden contratarse

como servicio (no se recomienda utilizar servicios de VPN gratuitos, pues éstos podrían tener el efecto contrario al deseado de proteger la información). En este sentido, es esencial utilizar un servidor VPN de confianza para el teletrabajo.

Como usuario de una conexión VPN, somos capaces de establecer comunicación con servidores alojados en todo el mundo. ¿Esto qué quiere decir? entre otras cosas, la página Web que visitemos registrará la IP que ofrece el servidor VPN determinado, y pongamos que se encuentra alojado en Estados Unidos. El sitio Web identificará la comunicación como proveniente de este país americano. Y esto ofrece una serie de ventajas como:

- Consumir contenido en Streaming de otros países -por una simple cuestión de derechos de autor-, cada país cuenta con su propio catálogo de contenidos en Streaming, ya sea Netflix o Amazon Prime Video. Incluso existen ciertos servicios exclusivos de un país, imposibles de acceder desde otros. Gracias a una conexión VPN, la orden del servidor para conectarnos a una de estas plataformas llega desde un país extranjero, por lo que el servicio detectará que estamos en ese país y nos ofrecerá el contenido correspondiente.

- Seguridad en el teletrabajo, gracias a las redes VPN los distintos equipos de una compañía pueden establecer comunicación entre sí, sin necesidad de que estos se encuentren conectados a la misma red. Pueden, incluso, estar en diferentes sucursales de su empresa, situadas en distintos puntos de la geografía mundial. Permitir el acceso a la red local de empresa desde un sitio externo supone un peligro para los datos almacenados. Sin VPN se pueden establecer contraseñas para acceder a esa red local, pero seguirá siendo susceptible de ataque. A través de una red VPN el acceso y la seguridad de los datos están protegidos.

- Evitar la censura en internet, ciertos países mantienen un control férreo de lo que lo que sus habitantes pueden consumir o no en internet. Por ello, una manera sencilla y segura de saltarse la censura es habilitando en su ordenador una conexión VPN. Por ejemplo, países como China, Corea del Norte o Emiratos Árabes Unidos captan ciertos contenidos que pueden ser accesibles gracias a la conexión VPN.

- Seguridad adicional en nuestra navegación, las conexiones VPN pueden venir acompañadas de un cifrado de la información que se transmite con ellas.

- Protección en descargas P2P, algunos proveedores bloquean la descarga de Torrents a través de programas P2P. Y sin tener en cuenta que podemos estar descargando material legítimo. Gracias a las conexiones VPN esto se podrá evitar.

Prevención de Ataques a una Red para mantener un cierto grado de protección de la información conectada a la red, las organizaciones, entidades y personas en general, deben comprender que su seguridad, y las formas en que esta se trata de vulnerar, mejoran constantemente; por tanto, lo principal y primero es entender cómo pueden sucederse estos ataques y en qué consisten dichas amenazas, con el fin de poder remediarlas de la mejor forma posible.

Existen diferentes formas efectivas para lidiar y mitigar los ataques que pueden suceder en una red. En función del tipo de ataque, algunas de ellas son:

- Para prevenir Malware, Adware y Spyware: instalar en los dispositivos un Software antivirus como primera protección, mantenerlo actualizado como segunda, y realizar escaneos periódicos como tercera. Ejecutar parches de seguridad en los sistemas operativos que se empleen, realizar copias de seguridad de los datos y emplear contraseñas diferentes y bien formadas para cada servicio al que se accede, configurar adecuadamente la seguridad de los navegadores y no descargar de la red aplicaciones o archivos de procedencia desconocida.

- Para prevenir Ransomware: La primera recomendación es tener actualizado el sistema operativo y todas las soluciones de seguridad, así como el antivirus y el cortafuegos personal habilitado (Firewall); evitar los accesos administrativos desde fuera de la entidad, y en caso necesario, permitirlos sólo mediante protocolos seguros. Activar la visua-lización de las extensiones de los ficheros para evitar ejecución de código dañino camuflado como ficheros legítimos no ejecutables, deshabilitar las macros en los documentos, educar a los usuarios de la red para reconocer amenazas antes de abrir archivos adjuntos enviados por correo electrónico, y realizar copias periódicas de respaldo de información para sistemas críticos.

- Para prevenir el Escaneo de Puertos: una forma efectiva es cerrar los puertos o servicios que no se utilizan en los sistemas, siempre que sea posible, emplear puertos no conocidos para determinadas aplicaciones (y no los configurados por defecto), tener configurado un cortafuegos (Firewall), y silenciar o desinformar las respuestas a encuestas de puertos.

- Para prevenir el Phishing: se recomienda a los usuarios de la red ir despacio ante cada acción, sin importar la urgencia que se emplee en los mensajes que se reciben; investigar los hechos y sospechar de cualquier comunicación no solicitada o desconocida. Rechazar y no responder ninguna solicitud de información confidencial como contraseñas o datos de tarjetas de crédito, así como nunca descargar ni ejecutar archivos adjuntos de personas desconocidas.

- Para prevenir ataques de Botnets: es necesario cambiar regularmente las contraseñas de acceso, incluido el Router de conexión; mantener actualizado el sistema operativo y el antivirus instalado, evitar realizar descargas P2P o vía Torrent porque, en muchas ocasiones, es la principal vía de entrada para estas infecciones, así como limitar el acceso a sitios cuya seguridad resulte sospechosa.

- Para prevenir la Denegación de Servicios (DoS): se deben bloquear direcciones IP que no se empleen, así como deshabilitar puertos y servicios de red innecesarios, aplicar filtros de enrutamiento, permitiendo solo el acceso al tráfico deseado, realizar una efectiva política de contraseñas, así como establecer la cantidad de ancho de banda a emplear por los usuarios.

- Para prevenir ataques de intermediario (Man In The Middle, MITM): la variante más adecuada es encriptar el tráfico que se envía por redes abiertas. Para ello se utilizan las redes virtuales privadas (VPN) y se emplea para los documentos la Infraestructura de las Llaves Públicas (Public key Infrastructure, PKI), Certificados Digitales, y una Entidad Certificadora, lo que contribuye a proteger los documentos enviados y confirmar la identidad de los usuarios mediante el cifrado de los mensajes. Una PKI establece un esquema de confianza en el cual ambas partes de una comunicación electrónica confían en un ente emisor para que verifique y confirme la identidad de ambas.

3.2.12 Ataques con Técnicas de Inteligencia Social

Los ataques de ingeniería social buscan engañar a los usuarios para obtener nombres de usuario, contraseñas, así como otra información sensible o bien que el usuario descargue archivos y/o instale programas que vulneran la seguridad del equipo de cómputo. La capacidad de identificar un ataque de ingeniería social minimiza en gran medida, el riesgo de ser víctimas de los ciberdelincuentes y ver comprometida información personal o de la organización para la que trabajamos. Para ello, se recomienda:

- Estar alertas ante comunicaciones, como llamadas, correos electrónicos, mensajes cortos (SMS), enlaces de teleconferencias e invitaciones de calendario de remitentes desconocidos.
- Antes de abrir cualquier enlace, archivo anexo, mensaje de texto o llamada de un remitente desconocido, hay que preguntarse lo siguiente:

  ◦ ¿Espero esa información? Si el mensaje proviene de un remitente desconocido (persona u organización), analizar bien antes de responder o hacer clic y/o descargar cualquier archivo adjunto.

  ◦ ¿Reconozco al remitente? Comprobar si la dirección está bien escrita (verificar que no haga falta ninguna letra, por ejemplo) y si el dominio (la terminación del correo electrónico) es de confianza y corresponde al nombre de quien envía el mensaje.

  ◦ ¿Solicitan que haga algo? Los correos electrónicos fraudulentos (Phishing) o los mensajes de texto de este tipo (Smishing) suelen pedir que se realice alguna acción como: hacer Clic en un hipervínculo, descargar algún archivo, responder al mensaje proporcionando información personal, etc. Con frecuencia, buscan generar una sensación de urgencia y provocar una reacción inmediata e irracional. Es necesario analizar con calma antes de proporcionar cualquier información que pudiera resultar comprometedora.

  ◦ Se debe desconfiar, particularmente, de los mensajes que parecerían genéricos (como "Estimados:", "A quien corresponda:", etc.).

  ◦ En el caso de comunicaciones referentes a instituciones bancarias y financieras, se recomienda nunca dar Clic en los enlaces contenidos en un correo o mensaje y no proporcionar información

> de acceso a tus cuentas. Si tienes alguna duda, debes contactar directamente a tu institución financiera (utilizando el número telefónico que vienen atrás de tu tarjeta, por ejemplo) para más orientación.

Siempre hemos de tener presente que garantizar la protección absoluta de su información digital es difícil sino imposible, pero debemos aprovechar todas las herramientas y avances que están a nuestro alcance. De esta forma, estaremos mejorando la seguridad de nuestros datos personales y sus contenidos. Son pequeños gestos diarios que, en caso de problemas, marcarán una gran diferencia.

3.3 Meltdown, Spectre y lo que se Acumule

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.

Un ataque que explotase 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 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[40] de Google. Ellos han publicado los detalles de dos ataques (no son los únicos[41]) basados en estos

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

[41] 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,

fallos de diseño. Los nombres de esos ataques son Meltdown y Spectre. Y en un sitio Web dedicado a describir estas vulnerabilidades destacaban 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.

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
```

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 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

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/

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 enlentecería los equipos de cómputo entre un 20 y un 40 por ciento dependiendo de la tarea que realizasen y el procesador del equipo.

¿Qué es Meltdown? este fallo explota una condición de carrera inherente al diseño de muchas CPU actuales. Esta condición se da entre los accesos a la memoria y la comprobación de privilegios durante el procesamiento de instrucciones. Además, en combinación con un ataque de canal lateral a la memoria caché de la CPU, esta vulnerabilidad permite que un proceso se salte las comprobaciones habituales de nivel de privilegio que normalmente aislarían al proceso maligno e impedirían que accediese a datos que pertenecen al sistema operativo y otros procesos concurrentes.

La vulnerabilidad permite que un proceso no autorizado lea información de cualquier dirección mapeada al espacio de memoria del proceso actual. Dado que la segmentación de instrucciones reside en los procesadores afectados, la información de una dirección no autorizada casi siempre se cargará temporalmente en la memoria caché de la CPU durante la ejecución fuera de orden, pudiendo posteriormente leerse desde la memoria caché. Esto puede suceder incluso cuando la instrucción de lectura original falla debido a una comprobación de privilegios que da negativo, o cuando no produce un resultado legible.

Dado que muchos sistemas operativos mapean la memoria física, los procesos del núcleo y otros procesos del espacio de usuario en el espacio de direcciones de cada proceso, Meltdown permite que un proceso maligno pueda leer cualquier memoria mapeada física, del núcleo o de otro proceso, con independencia de si debería o no poder hacerlo. Las defensas contra Meltdown exigirían evitar el uso del mapeo de memoria de un modo que resultase vulnerable a tales amenazas (una solución basada en Software), o bien evitar la condición de carrera subyacente (una modificación del microcódigo de la CPU o la ruta de ejecución).

El agujero es viable en cualquier sistema operativo en el que la información privilegiada se mapea a memoria virtual para procesos no privilegiados, una característica que incorporan muchos sistemas operativos actuales. Por lo

tanto, potencialmente Meltdown puede afectar a una gama de dispositivos mayor que la identificada actualmente, dado que apenas hay variaciones entre las familias de microprocesadores utilizados por estos sistemas.

¿Qué es Spectre? es una vulnerabilidad que permite a los programas alojados en el sistema operativo del usuario acceder a una dirección arbitraria del espacio de memoria de un programa. En lugar de una única vulnerabilidad de fácil corrección, el documento de Spectre describe una clase entera de vulnerabilidades potenciales. Todas esas vulnerabilidades se basan en explotar los efectos secundarios de la ejecución especulativa, una técnica empleada comúnmente para combatir la latencia de la memoria y acelerar así la ejecución en los microprocesadores modernos.

En particular, Spectre se centra en la predicción de saltos, un caso especial de la ejecución especulativa. A diferencia de la vulnerabilidad Meltdown hecha pública la misma fecha, Spectre no depende de una característica en particular de la gestión de memoria de un procesador en concreto o de cómo proteja el acceso a esa memoria, sino que tiene un enfoque más general.

El punto de partida del documento de Spectre es un ataque sincronizado de canal lateral aplicado al sistema de predicción de saltos de un microprocesador moderno que utilice ejecución fuera de orden. Si bien al nivel arquitectónico documentado en las hojas técnicas de los procesadores los resultados de un fallo en la predicción se especifican que quedarán anulados tras darse dicha circunstancia, la ejecución especulativa resultante puede aun así dejar efectos colaterales, como líneas de caché cargadas con determinada información.

Estos efectos colaterales pueden posteriormente afectar a los denominados aspectos no funcionales del sistema informático. Si tales efectos colaterales -incluyendo, entre otros, los tiempos de acceso a la memoria- resultan visibles para un programa malicioso, y puede hacerse que esos efectos dependan de información sensible en posesión del proceso que hace las veces de víctima, entonces estos efectos colatelares pueden hacer que la información sensible resulte deducible. Esto puede ocurrir aunque los sistemas de seguridad formales a nivel de arquitectura funcionen de forma correcta -lo que sucede es que las optimizaciones en los niveles más bajos a nivel de microarquitectura dirigidas a la aceleración la ejecución de código pueden revelar información no esencial para la correcta ejecución de un programa-.

Falla en los Procesadores Snapdragon El día 8 de agosto del 2020 se publico que los procesadores Qualcomm Snapdragon[42] de más de 3 mil millones de dispositivos Android del mundo son vulnerables a ataques que pueden convertirlos en herramientas de espionaje al explotar más de cuatrocientas vulnerabilidades y no se sabe cuando Google y los fabricantes de dispositivos móviles incorporaraán la solución de Qualcomm.

Las vulnerabilidades pueden explotarse cuando un objetivo descarga un video u otro contenido que es procesado por el chip. Los objetivos también pueden ser atacados instalando aplicaciones maliciosas que no requieren ningún permiso. Desde allí, los atacantes pueden monitorear ubicaciones y escuchar audio cercano en tiempo real y filtrar fotos y videos. Las infecciones se pueden ocultar del sistema operativo de tal manera que dificulta la desinfección.

Snapdragon es lo que se conoce como un sistema en un chip que proporciona una gran cantidad de componentes, como una CPU y un procesador de gráficos. Una de las funciones conocida como procesamiento de señal digital o DSP, aborda una variedad de tareas, incluidas las capacidades de carga y de las funciones de video, audio, realidad aumentada y otras funciones multimedia.

Los investigadores de la firma de seguridad Check Point en un breve informe sobre las vulnerabilidades que descubrieron indican que estos Chips introducen una nueva superficie de ataque y puntos débiles en estos dispositivos móviles por las múltiples vulnerabilidades de los DSP. Qualcomm ha lanzado una solución a las fallas, pero hasta el momento no se han incorporado al sistema operativo Android.

Por el momento Check Point está reteniendo detalles técnicos sobre las vulnerabilidades y cómo se pueden explotar hasta que las soluciones lleguen a los dispositivos del usuario final. Check Point ha denominado a las vulnerabilidades Achilles. Los más de 400 errores distintos se rastrean como:

> CVE-2020-11201, CVE-2020-11202, CVE-2020-11206, CVE-2020-11207, CVE-2020-11208 y CVE-2020-11209.

En un comunicado, los funcionarios de Qualcomm dijeron: "Con respecto a la vulnerabilidad de Qualcomm Compute DSP revelada por Check Point, trabajamos diligentemente para validar el problema y poner las mitigaciones

[42] Representa a más del 40% de los teléfonos inteligentes del mercado incluyendo a los teléfonos de las compañías Google, Samsung, LG, Xiaomi, OnePlus, Motorola, etc.

adecuadas a disposición de los OEM. No tenemos evidencia de que esté siendo explotado actualmente. Alentamos a los usuarios finales a actualizar sus dispositivos a medida que los parches estén disponibles y solo a instalar aplicaciones de ubicaciones confiables como Google Play Store".

No hay mucha orientación útil para proporcionar a los usuarios cómo protegerse contra estos ataques. Descargar aplicaciones solo desde Play puede ayudar, pero el historial de verificación de aplicaciones de Google muestra que los consejos tienen una eficacia limitada. Tampoco hay forma de identificar eficazmente el contenido multimedia explotable.

Falla en el Chip T2 de las Mac Expertos en ciberseguridad de la empresa belga *ironPeak* han confirmado los rumores de que los chips de seguridad T2 pueden llegar a ser hackeados: una determinada combinación de dos 'exploits' diferentes proporciona al atacante la oportunidad no sólo de cambiar el comportamiento del chip, sino incluso de instalar Malware en el mismo (como keyloggers).

Y hay tres factores que hacen que esta noticia resulte especialmente chocante... y preocupante:

- Que el objetivo de estos Chips T2 es precisamente el de salvaguardar la seguridad de los datos: se trata de un procesador independiente que actúa de intermediario entre la CPU y el sistema operativo y que permite supervisar que sólo se ejecute en el equipo Software aprobado por Apple.
- Que el Chip T2 viene instalado en todos los iMac Pro, Macbook Pro y Macbook Air vendidos desde 2018, en todos los Mac Pro desde 2019 y en todos los iMac del 2020. Es decir, afecta a un número notable de usuarios.
- Pero lo peor de todo es que, aunque el atacante puede modificar el comportamiento del Chip T2, la vulnerabilidad no puede ser parcheada preventivamente, porque ésta se basa precisamente en la escritura del apartado de sólo lectura del Chip.

¿Cómo Funciona este Ataque? los Exploits usados para esta clase de ataque son dos que ya se venían usando para aplicar el Jailbreak a dispositivos iOS (el T2 está basado en el chip A10, usado por los iPhones más antiguos). Según *ironPeak*, este método funciona porque: "Apple dejó abierta

una interfaz de depuración cuando envió los Chips T2 a los clientes, lo que permite a cualquiera entrar en el modo de actualización de Firmware del dispositivo sin autenticación".

Con este método, es posible crear un cable USB-C que puede explotar automáticamente la vulnerabilidad de su dispositivo MacOS durante el arranque. Esto permite a un atacante obtener acceso *root* sobre el Chip T2 y modificar o controlar cualquier cosa que se ejecute en el dispositivo atacado.

El lado bueno de esto es que, como habrás deducido de leer lo anterior, esta clase de ataque requiere de acceso físico para funcionar (idealmente, de varios accesos físicos). En principio, el riego para un usuario medio de Mac es bastante bajo (otra cosa son aquellos equipos integrados en redes de grandes empresas y organismos públicos).

Ante la imposibilidad de parchearlo, desde *ironPeak* se muestran convencidos de que Apple desarrollará una nueva versión del T2 para los próximos equipos que lance al mercado... la cual dejará de ser necesaria cuando desembarque la próxima generación Apple Silicon, que traerá incluidas las funciones de seguridad en el propio Chip ARM.

3.4 Distribuciones de Linux «Seguras y para Penetración»

Para muchos, Linux y Mac OS son dos sistemas operativos más seguros que Windows de Microsoft, pero con todo, hay algunas distribuciones especializadas de Linux que satisfacen las necesidades de temas relacionados con la seguridad, pruebas de penetración, análisis forense y auditorías de seguridad.

Las distribuciones seguras intentan preservar la privacidad y el anonimato, ayudan a utilizar internet de forma anónima y evitar la censura en prácticamente cualquier lugar y cualquier equipo de cómputo, pero sin dejar rastro a menos que lo solicites explícitamente.

Las distribuciones para pruebas de penetración ofrecen herramientas para penetración, análisis forense y auditorías de seguridad en donde se realizan distintos tipos de tareas que identifican, en una infraestructura objetivo, las vulnerabilidades que podrían explotarse y los daños que podría causar un atacante. En otras palabras, se realiza un proceso de Hacking ético para identificar qué incidentes podrían ocurrir antes de que sucedan y, posteriormente, reparar o mejorar el sistema, de tal forma que se eviten estos ataques.

Las distribuciones anónimas ofrecen niveles adicionales de privacidad y

seguridad y se complementan con las distribuciones para pruebas de penetración que ofrecen herramientas para penetración y auditorías de seguridad (mediante el uso de tecnologías como TOR[43], Sandbox[44], Firewall[45], herramientas criptográficas[46], etc.). De ambos tipos de Softwares hay varias alternativas diferentes, tanto comerciales como de Software libre, por lo que decidirse por una u otra, en ocasiones puede ser una tarea un tanto complicada. Es por ello que aquí listamos algunas de las distribuciones de Linux más usadas en la actualidad, apartados con los que cada vez debemos prestar más atención.

Las distribuciones seguras y para penetración son sistemas operativos Live[47] diseñados para ser usados desde un CD, DVD, memoria USB o máquina virtual independientemente del sistema operativo original de la computadora.

[43] Tor es una red abierta y distribuida que te ayuda a defenderte de una forma de vigilancia en la red que amenaza tu libertad y privacidad, tus actividades comerciales confidenciales y relaciones, además de la seguridad gubernamental. Además, te protege redirigiendo tus comunicaciones alrededor de una red distribuida de retransmisores realizados por voluntarios alrededor del mundo: lo cual previene que alguien observe tus comunicaciones a partir de los sitios que visitas, también evita que los sitios en que navegas obtengan tu ubicación física.

[44] Sandbox es un mecanismo para ejecutar programas con seguridad y de manera separada. A menudo se utiliza para ejecutar código nuevo, o Software de dudosa confiabilidad proveniente de terceros. Ese entorno aislado permite controlar de cerca los recursos proporcionados a los programas cliente a ejecutarse, tales como espacio temporal en discos y memoria. Habitualmente se restringen las capacidades de acceso a redes, la habilidad de inspeccionar la máquina anfitrión y dispositivos de entrada entre otros. En este sentido, el aislamiento de procesos es un ejemplo específico de virtualización.

[45] Un cortafuegos es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar o descifrar el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios. Los cortafuegos se utilizan con frecuencia para evitar que los usuarios de internet no autorizados tengan acceso a redes privadas, especialmente intranets.

[46] El surgimiento de redes de comunicación, en particular de internet, ha abierto nuevas posibilidades para el intercambio de información. Al mismo tiempo, son cada vez mayores las amenazas a la seguridad de la información que se transmite. Es necesario entonces, crear diferentes mecanismos, dirigidos a garantizar la confidencialidad y autenticidad de los documentos electrónicos, todo ello es parte de la Criptografía.

[47] 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.

Distribuciones de GNU/Linux «Seguras»

Tails (https://tails.boum.org/)

Para muchos esta es la primera opción a la hora de buscar una solución de seguridad en Linux. También conocida como «The Amnesic Incognito Live System», esta es una distribución basada en Debian. Es un proyecto de código abierto que se publicó por primera vez hace 8 años y que redirige todo el tráfico Web a través de Tor logrando la privacidad a través del anonimato. Además, almacena todo en la RAM y evita el uso del disco duro, por lo que borra todo una vez se apaga. La imagen tiene un tamaño menor de 1.2 GB y necesita al menos 2 GB de RAM en un equipo de 64 bits, se puede usar en formato Live, como máquina Virtual o bien instalarse en una USB, DVD o en el disco duro del equipo.

Whonix (https://www.whonix.org/)

Es una distribución que se basa en Debian y consta de dos máquinas virtuales, una es Tor Gateway que se ejecuta en Debian, mientras que la otra es una Workstation. Whonix se instala en un sistema operativo Host proporcionado por el usuario que puede ser Linux, Windows, MacOS o Qubes OS. Así al utilizar la red abierta y distribuida de transmisión de Tor, Whonix echa abajo las posibilidades de vigilancia de la Red. Además, y por motivos de seguridad, hace todo lo posible para ocultar nuestra dirección IP real.

Qubes OS (https://www.qubes-os.org/)

Se conoce como uno de los sistemas operativos más seguros del mundo y se basa en realizar la virtualización mediante el «hipervisor Xen» -un hipervisor imita el Hardware y permite ejecutar varias máquinas virtuales simultáneamente-. El entorno de usuario de Qubes OS podría ser Fedora, Debian, Whoix o Windows y, al igual que Tails. Así mismo utiliza diferentes dominios para separar los niveles de confianza, por ejemplo, un dominio de trabajo, un dominio para el ocio, etc.; los cuales se ejecutan en diferentes máquinas virtuales, esta versión requiere un mínimo de 16 GB de RAM.

Subgraph OS (https://subgraph.com/)

Nos encontramos con un sistema operativo seguro basado en Debian que promete proporcionar una experiencia digital anónima. Ha sido diseñado para evitar diferentes ataques de Malware, es capaz de ser una plataforma

de comunicación segura además de proporcionar una interfaz de usuario muy sencilla.

Discreete Linux (https://www.privacy-cd.org/)
En este caso nos encontramos con un proyecto de Software libre que puede ser utilizado por cualquier persona que desee llevar una vida digital anónima también basado en Debian. Además, protege a sus usuarios contra los ataques de vigilancia accionados por troyanos y hoy en día esta en fase Beta. Es una de las alternativas más adecuadas para los usuarios que no tienen un conocimiento muy profundo de estos sistemas pero que consideran la seguridad en internet como su principal preocupación. Hace uso de cifrados y entornos aislados para proporcionar un entorno de trabajo seguro. Así mismo no utiliza los discos duros internos del equipo, ya que almacena todos sus datos en la memoria RAM o en una unidad de disco USB externa.

Kodachi (https://www.digi77.com/linux-kodachi/)
Es un sistema operativo centrado en la seguridad y basado en Debian cuyo objetivo es proporcionar una experiencia informática segura. Ponerlo en marcha es muy sencillo y no necesita demasiado tiempo, ya que permite la opción de arrancar desde el Hardware del PC, o desde una unidad USB externa para mayor seguridad. Hace uso de elementos tales como una conexión VPN activa, el servicio TOR y DNScrypt con el que garantiza un buen nivel de privacidad. Además, todo el sistema operativo se ejecuta desde la memoria RAM del equipo, por lo que después de apagado no queda ningún rastro de actividad.

Tens (https://www.spi.dod.mil/lipose.htm)
También conocido como Trusted End Node Security, este sistema es distribuido y desarrollado por el Departamento de Defensa de los Estados Unidos. Se basa en Arch Linux y puede ejecutarse en cualquier equipo con tecnología Intel. Sólo arranca desde la RAM y viene cargado con un asistente de encriptación, un Software de encriptación simple y potente para la protección de nuestra información confidencial.

Tin Hat (https://sourceforge.net/projects/tinhat/)
Esta propuesta es una derivación de Gentoo y es un sistema operativo seguro que se ejecuta en su totalidad en la RAM del equipo, por lo que no

monta ningún sistema de archivos directamente en el dispositivo de arranque, evitando así la posibilidad de dejar expuesto cualquier dato. Como era de esperar, podremos arrancarlo desde un CD o desde una unidad flash USB. Puede ejecutarse tanto en arquitecturas de Hardware de 32 como de 64 bits y es recomendable tener conocimientos previos de Gentoo Linux.

IprediaOS (https://www.ipredia.org/os/)
Para empezar diremos que I2P es una capa de comunicación P2P anónima que se crea utilizando herramientas de código abierto, algo en lo que se basa IprediaOS, ya que orienta todo su tráfico a través de I2P y se asegura de que toda su actividad Online no pueda ser interceptada por terceros. Así hace uso de múltiples capas de cifrado y cabe mencionar que la red I2P es una red dinámica y distribuida.

Alpine Linux (https://alpinelinux.org/)
Es una distribución diseñada principalmente para los usuarios avanzados que valoran la seguridad, la eficiencia de recursos y la simplicidad. En un principio parte como bifurcación del proyecto *LEAF* aunque, a diferencia de este, Alpine mejora las características de seguridad y cuenta con un Kernel más actual. Su funcionamiento se centra en la privacidad, por lo que utiliza su propio sistema de gestión de paquetes.

Openwall (https://www.openwall.com/Owl/)
Es una pequeña distribución de Linux con seguridad mejorada (SELinux) para servidores, dispositivos y dispositivos virtuales. A diferencia de otras distribuciones, el uso de SELinux por parte de Openwall evita que se incorpore Software vulnerable a la distribución, en lugar de depender de parches para remediar vulnerabilidades de seguridad conocidas o características diseñadas para disminuir el impacto de los errores de seguridad. Mediante el uso del marco *SELinux*, Openwall eclipsa a la mayoría de sus contrapartes más grandes en este sentido.

Distribuciones de GNU/Linux «Para Penetración» Realizar pruebas de penetración, análisis forense y auditorías de seguridad resulta ser una tarea compleja e involucra un proceso en donde se realizan distintos tipos de tareas que identifican, en una infraestructura objetivo, las vulnerabilidades que podrían explotarse y los daños que podría causar un atacante. En

otras palabras, se realiza un proceso de Hacking ético para identificar qué incidentes podrían ocurrir antes de que sucedan y, posteriormente, reparar o mejorar el sistema, de tal forma que se eviten estos ataques.

Para realizar una prueba de penetración de forma profesional, es necesario sumar a los conocimientos de Hacking ético, otros aspectos fundamentales como: programación, metodologías, documentación, entre otros. No obstante, esos aprendizajes suelen venir una vez que se conocen y se saben utilizar muchas herramientas que son parte del proceso de pruebas de penetración. Las siguientes herramientas se deben conocer, instalar y probar para dar los primeros pasos en este "arte".

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, Hackers éticos e investigadores de seguridad en todo el mundo y contiene cientos de herramientas para el trabajo forense, esta distribución es basada en Debian.

Parrot OS (https://parrotlinux.org)
Puede verse como un laboratorio totalmente portátil para una amplia gama de operaciones de seguridad cibernética, desde pruebas de penetración hasta ingeniería inversa y análisis forense digital, pero esta distribución basada en Debian también incluye todo lo que necesita para proteger sus datos y desarrollar su propio Software.

BlackArch Linux (https://blackarch.org)
Esta popular distribución de pruebas de penetración proviene de Arch Linux y contiene más de 2,400 herramientas de penetración y análisis forense diferentes, lo que le permite usar lo que necesite sin tener que descargar nuevas herramientas.

BackBox (https://www.backbox.org/)
Es una distribución de Linux basada en Ubuntu destinada a ayudar a Hackers éticos y probadores de penetración en evaluaciones de seguridad. BackBox OS está diseñado con el objetivo de ser más rápido, fácil de operar y tener un entorno de escritorio mínimo. La ventaja clave de BackBox es que sus propios repositorios de software se actualizan a intervalos regulares para

mantener la distribución estable y popular para las operaciones del mundo real.

DEFT Linux (http://www.linuxandubuntu.com/home/deft-linux-a-linux-distribution-for-computer-forensics)

DEFT (Digital Evidence and Forensics Toolkit) se basa en GNU Linux y DART (Digital Advanced Response Toolkit), un sistema forense que comprende algunas de las mejores herramientas para la respuesta forense y de incidentes. DEFT Linux está especialmente diseñado para llevar a cabo tareas forenses y se ejecuta en vivo en los sistemas sin alterar el disco duro o cualquier otro medio de almacenamiento. Se compone de más de 100 herramientas forenses y de piratería de alta calificación.

Network Security Toolkit (https://www.networksecuritytoolkit.org)

El Network Security Toolkit (NST), basado en Fedora, es una versión Live que consta de las 125 principales herramientas de seguridad de código abierto proporcionadas por *insecure.org* para validar la seguridad de la red, pruebas de penetración, diagnósticos de red y monitoreo del día. El objetivo principal detrás del desarrollo de NST es proporcionar a los administradores de red/sistemas un conjunto combinado de herramientas de seguridad de código abierto para llevar a cabo operaciones como análisis de tráfico de red, detección de intrusos, escaneo de red y parches de seguridad.

Otras Distribuciones Útiles Existe una gran variedad de distribuciones Live de Linux (The LiveCD List https://livecdlist.com/) que permiten hacer una gran cantidad de cosas útiles, a continuación damos una lista de algunas de ellas:

- KNOPPIX (http://www.knoppix.org/)
- GNOME Partition Editor (https://gparted.org/)
- System Rescue (https://www.system-rescue.org)
- Parted Magic (https://partedmagic.com/)
- Ultimate Boot (https://www.ultimatebootcd.com/)
- Super Grub2 (https://www.supergrubdisk.org/)

- Rescatux (https://www.supergrubdisk.org/rescatux/)
- Rescue (https://en.altlinux.org/Rescue)
- Ddrescue (https://www.gnu.org/software/ddrescue/)
- INSERT (http://www.inside-security.de/insert.html)
- Boot-repair (https://sourceforge.net/p/boot-repair/home/Home/)
- Rescuezilla (https://rescuezilla.com/)
- Clonezilla (https://clonezilla.org/)
- Redo Backup (ww12.redobackup.org)
- Mondo Rescue (www.mondorescue.org)
- Live Wifislax (https://www.wifislax.com/)
- Puppy Linux (http://puppylinux.com/)
- Tiny Core Linux (http://tinycorelinux.net/)
- Debian Live (https://www.debian.org/CD/live/)
- Ubuntu (https://ubuntu.com/)

4 Búsquedas Básicas en la Red

Buscar en la red es principalmente saber usar al 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 hoy voy a explicar algunas formas de realizar búsquedas avanzadas en Google utilizando comandos. En concreto, voy a explicar los comandos que utilizo en mi primer acercamiento a una página web cuando tengo que realizar una consultoría SEO (Search Engine Optimization).

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 incluyo todos los que me 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 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 necesita 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

Excluir Palabras digamos que quieres buscar contenido acerca demarketing 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 el 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 cuantas palabras pueden estar separadas de los dos términos.

Ejemplo de búsqueda: content marketing around(10) video

Buscar fraces cercanas a otras el operador AROUND(X) busca fraces cercanas a otras, donde x indica cuantas 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

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 Info el atributo info sirve para obetener 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"). Búsqueda info de la Wikipedia.

Ejemplo de búsqueda: info:www.wikipedia.org

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 titulo de la pagina, por ejemplo

Ejemplo de búsqueda: intitle:index.of + mp3 + "Radiohead" -html -htm -php -asp apache

Búsqueda de Peliculas 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 citen 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 loc:paris

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 busquedas

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.

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

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 matchups? 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 imagenes por tamaño podemos usar imagesize para acotar la búsqueda de imagenes.

Ejemplo de búsqueda: imagesize:500x400
Ejemplo de búsqueda: anillo imagesize:50x40

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, solo 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 todo poderoso 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 esta 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. ¡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 de Discover, un recurso que interpreta el comportamiento del usuario mobile 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.

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 innúmeras publicaciones científicas mediante la subscripción 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 búqueda 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 nºISBN. 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 sacer 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 categoría: 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 kewywords, 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.

6 Lenguajes de Programación y Entornos de Desarrollo

Hay muchas aplicaciones a las herramientas computacionales, pero nos interesan 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[48], y repetir el proceso para calcular el siguiente estado. Este código o programa se escribe en un lenguaje de programación que sigue algún paradigma de programación[49], que posteriormente puede ser ejecutado por una unidad central de procesamiento —computadora—.

[48] Un algoritmo es un conjunto preescrito de instrucciones o reglas bien definidas, ordenadas 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.

[49] Este representa un enfoque particular o filosofía para diseñar soluciones e implementarlas en algún lenguaje de programación.

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—.

Los Tipos más comunes de paradigmas de programación son[50]:

- 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 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.

[50] En general la mayoría son variantes de los dos tipos principales: imperativa y declarativa.

- 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[51] 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[52].

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 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

[51] 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.

[52] 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.

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[53] 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.
- Else: Siempre precedido de "If" y en ocasiones de "Else If". Indica que debe ejecutarse cuando no se cumplan las condiciones previas.

[53] 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").

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[54] 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[55] de la referencia de implementación—.

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[56] para la estructura gramática.

[54] 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.

[55] 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.

[56] La notación de Backus-Naur, también conocida por sus denominaciones inglesas Backus-Naur form (BNF), Backus-Naur formalism o Backus normal form, es un meta-

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, dependiendo de su naturaleza —espacio en disco que utiliza, tráfico de red que genera, etc.—.

- 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.

lenguaje usado para expresar gramáticas libres de contexto: es decir, una manera formal de describir lenguajes formales.

- 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, en la Facultad de Ciencias se estudian algunas decenas de ellos, entre los que destacan: C, C++, Java y Python.

¿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 mas 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 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[57], 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 decisón 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

[57] 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. 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 lo que 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
```

Uso de Espacios o Tabuladores en Fuentes Los programadores siempre han debatido entre el uso de espacios y ta-bulaciones 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 sub-bloques, 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 el 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 convertir los espacios en tabuladores usamos:

```
$ unexpand prog.c
```

o redireccionando la salida usando:

```
$ unexpand prog.c > progTab.c
```

Para convertir los tabuladores en espacios (1 tabulador en 10 caracteres) usamos:

```
$ expand prog.c
```

o redireccionando la salida usando:

```
$ expand –tabs=10 prog.c > progTab.c
```

6.1 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 responsabilidad 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 procedimientos 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.

6.2 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
```

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:

```
while (z < 20) {
  z = z + x;
}
```

Ejemplo en Python:

```
while z < 20:
  z = z + x
```

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. 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. La estructura básica es:

Pseudocódigo:

```
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*[62] 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 (véase **??**) 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
```

[62] 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 cribaEratostenes(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:[63]
      xi = xi + 1
      if (n % i) == 0:
        break
    if xi == np:
      p.append(n)
      np = np + 1
    n = n + 1
   # Visualiza los primos encontrados
```

[63] 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 = cribaEratostenes(1000)
print(P)
```

Otro ejemplo en Python:

```
def cribaEratostenes(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):[64]
        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 = cribaEratostenes(1000)
print(P)
```

[64] 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*.

6.3 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[65], cuando los ejemplos crecen se empieza a usar programación estructurada[66] —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[67].

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.

[65] 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.

[66] 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.

[67] 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.

6.4 Errores de Redondeo y de Aritmética en la Programación

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 6.2), 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 comunmente 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 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 aproximación a la raíz buscada y x_n es la aproximación a la raíz (sí se converge a ella).

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:

```
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 (véase 6): 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 metodoNewtonRapson(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.metodoNewtonRapson(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.metodoNewtonRapson(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()
```

6.5 Lenguajes y Entornos de Desarrollo

Para programar, es necesario como mínimo contar con un editor de texto —como *vi* o *nano*— 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.

Instalación de Java e IDEs en Debian GNU/Linux Existen diversas versiones de Java para Linux, la más usada es JDK de Oracle pero también esta una versión abierta llamada OpenJDK, para instalar por ejemplo OpenJDK 8 en Debian GNU/Linux es necesario hacer:

```
# apt install openjdk-8-jre openjdk-8-jdk openjdk-8-doc
```

y después actualizar la versión por omisión de Java:

```
# update-java-alternatives -s java-1.8.0-openjdk-amd64
```

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
# apt install eclipse eclipse-cdt eclipse-pydev netbeans blue-
fish codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim vim-
python-jedi vim-tlib vim-latexsuite vim-nox
# apt install kdiff3 meld diffuse dirdiff kompare numdiff col-
ordiff wdiff xxdiff tkdiff ndiff
# apt install alleyoop astyle c2html java2html code2html c2html
autodia txt2html html2text
```

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:

http://132.248.182.159/acl/Herramientas/Lenguajes/Java/

Instalación de C y C++ e IDEs en Debian GNU/Linux 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 c++-annotations-pdf c++-compiler
c-compiler g++ cppcheck splint cccc autoconf automake make
cmake scons jam ohcount
```

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
# apt install eclipse eclipse-cdt eclipse-pydev netbeans blue-
fish codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim vim-
python-jedi vim-tlib vim-latexsuite vim-nox
# apt install kdiff3 meld diffuse dirdiff kompare numdiff col-
ordiff wdiff xxdiff tkdiff ndiff
# apt install ddd-doc ddd gdb xxgdb kdbg valgrind valgrind-
mpi kcachegrind cppcheck time nemiver
# apt install ggcov lcov gcov
# apt install alleyoop astyle c2html java2html code2html c2html
autodia txt2html html2text
```

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:

http://132.248.182.159/acl/Herramientas/Lenguajes/CyC++/

Instalación de Python e IDEs en Debian GNU/Linux Existen diversas versiones de Python para Linux, para instalar Python 2 en Debian GNU/Linux es necesario hacer:

```
# apt install ipython python-matplotlib python-numpy python-scipy python-pandas python-sympy python-nose python-dev spyder python-rpy2 python-rpy python-axiom python-pip python python-tk idle python-pmw python-imaging python-pip bpython python-pandas python-sklearn python-sklearn-doc wxglade python-rpy2 pypy pypy-tk python-gnuplot pypy-six jython boa-constructor pyragua dreampie winpdb mypy python-mpi4py
```

para instalar Python 3 en Debian GNU/Linux es necesario hacer:

```
# apt install ipython3 python3 idle3 python3-pip python3-matplotlib python3-rpy2 python3-numpy python3-scipy spyder3 bpython3 python3-pandas python-sklearn python-sklearn-doc pyzo pyzo-doc spe eric python-wxgtk3.0 python-wxgtk3.0-dev python3-mpi4py
```

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
```

Además 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
# apt install eclipse eclipse-cdt netbeans-platform netbeans-ide codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe
```

```
# apt install kdiff3 meld diffuse dirdiff kompare numdiff colordiff wdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html c2html autodia txt2html html2text
```

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 Distribution es una distribucción 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 esta SageMath, una Suite de código abierto bajo la licencia GPL de Software matemático como: *NumPy, 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/.

Compiladores y Mejora del Rendimiento en Python Python es un lenguaje interpretado, pero es posible compilar el código (no aumenta la velocidad de ejecución) para generar Byte Code para el interprete. 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 interprete de Python.

El módulo py_compile puede compilar manualmente cualquier módulo. Una forma de usaer la función *py_ compile.compile* en ese modulo de forma interactiva es:

```
>>> import py_compile
>>> py_compile.compile('modulo.py')
```

esto escribirá el archivo *modulo*.pyc.

En la linea 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:

```
$ 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. 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[68] a lo largo de los años, incluidos[69] *Nuitka, PyPy, Cython, cx_FreezeIron, Pythran, Python y Jython.*

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*:

[68] 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.

[69] 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.

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 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*. 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 tomo 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:

$ 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).

Conclusión Una buena opción es *PyPy* por el rendimiento, la compilación fue muy rápida y produjo los resultados menos de un segundo después de presionar la tecla Intro. Si quieres un exe, sin embargo, te recomiendo Nuitka; fue una compilación sin complicaciones y se ejecuta más rápido que CPython. Experimente con estos compiladores de Python y vea cuál funciona mejor para tus necesidades particulares.

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:

http://132.248.182.159/acl/Herramientas/Lenguajes/Python/.

6.6 Documentación del Código Fuente

La documentación dentro del código fuente (véase 6) tiene como objetivo que los que lo lean, mantengan y reparen el código, lo entiendan.

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 menos de un minuto, 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 a continuación.

6.6.1 Documentar en C, C++ y Java

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:
http://132.248.182.159/Herramientas/Programacion/DOXYGEN/EjemploUso/JerarquiaDeCla
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
```

Hay varios estilos de documentación (véase 6), 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 ver-
sion of the class
v0.0.2 March 2011 Gerardo Cisneros add HYPRE errors con-
trol
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

6.6.2 Documentar en Python

En Python el uso de acentos y caracteres extendidos esta soportado por la codificación UTF-8 (véase 6), 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.

6.7 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:

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[70]

[70] 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/

Programas para control de proyectos

- Git https://git-scm.com/
- Mercurial https://www.mercurial-scm.org/
- Subversion https://subversion.apache.org/

Generadores de documentación[71]

- Doxygen http://www.doxygen.org/

Formateador de código fuente para C, C++, Java y C#

- Astyle http://astyle.sourceforge.net/

Lenguaje unificado de modelado (Unified Modeling Language UML)[72]

- UML http://www.uml.org/

En este apartado, solo tocaremos las más usadas, pero abunda la documentación de estas y otras importantes herramientas en línea de comandos

[71] Otras opciones son: Natural Docs, ROBODoc, etc.

[72] 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.

(véase **??**). Iniciaremos por las de compilar[73] y depurar[74] programas compilables en C, C++, Fortran, entre otros.

En Linux existe una gran variedad de herramientas para programación (por ejemplo para C, C++, Fortran, etc.), para instalar algunas de ellas en Debian GNU/Linux podemos usar en línea de comandos:

```
# apt install g++ valgrind valgrind-mpi kcachegrind cppcheck
astyle
```

Para usar muchas de estas herramientas (en línea de comandos), primero debemos conocer como compilar fuentes[75], sin perdida de generalidad trabajaremos en C++ solicitando que el archivo ejecutable[76] tenga el nombre *ejemp*:

```
$ g++ *.cpp -o ejemp
```

para ejecutar el programa ya compilado:

```
$ ./ejemp
```

Compilar usando diversas optimizaciones[77] (–O0, O1, -O2 o -O3) usar:

[73] 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.

[74] Un depurador (en inglés, debugger), es un programa usado para probar y depurar (eliminar) los errores de 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 más lento que ejecutando el código directamente en el apropiado (o el mismo) procesador.

[75] Compilador para C es gcc, para C++ es g++, para Java es javac, para Fortran es f77 o f95, en Python es py3compile o pycompile, etc.

[76] Un archivo ejecutable es tradicionalmente un archivo binario con instrucciones en codigo de máquina cuyo contenido se interpreta por el ordenador como un programa. Además, suele contener llamadas a funciones específicas de un sistemas operativo.

[77] 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.

```
$ g++ -O1 *.cpp
```

para ejecutar el programa ya compilado:

```
$ ./a.out
```

Para compilar y ver todos los avisos usar:

```
$ g++ -pedantic -Wall -Wextra -O *.cpp
```

o de forma alternativa:

```
$ g++ -Weffc++ *.cpp
```

Para hacer análisis de rendimiento, hacer:

```
$ 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:

http://132.248.182.159/acl/Herramientas/Programacion/GPROF/

Para conocer el tiempo de ejecución[78] de un programa, podemos usar el comando básico *time*, mediante:

```
$ time ejecutable
```

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
   Average total size (kbytes): 0
   Maximum resident set size (kbytes): 2360
   Average resident set size (kbytes): 0
   Major (requiring I/O) page faults: 0
```

[78] 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).

```
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
```

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* cada uno tiene sus pros y contras, depende del usuario cual es el más adecuado para usar.

Para el rastreo de problemas con la manipulación de memoria y punteros desbordados:

```
$ 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:

http://132.248.182.159/acl/Herramientas/Programacion/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

Para hacer una revisión estática del código en C++ usar:

$ cppcheck –enable=all *.?pp

mostrará los avisos de análisis estático del código indicado.

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:

http://132.248.182.159/acl/Herramientas/SistemasOperativos/Linux/

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. Fué diseñado por Linus Torvalds y es usado para controlar los cambios de diversos proyectos como los fuentes del Kernel de Linux (véase **??**) 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 que se realizan sobre los elementos de algún producto o una configuración del mismo es decir a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración, y para los que aún no les queda claro del todo, control de versiones 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*.

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, 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

Para instalar *Git* completo en el servidor o en la máquina de trabajo:

```
# apt install git-all
```

Para instalar lo básico de *Git*, si no esta 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
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs
```

Configuración: 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 "John Doe"
$ git config –global user.email johndoe@example.com
```

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
```

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 "John Doe"
$ git config user.email johndoe@example.com
```

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'
```

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 via *ssh*, por ello es mejor cambiar en */etc/passwd*, la línea del usario 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 "Nombre"
$ git config –global usr.email usr@direccion
```

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:

```
$ 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:

http://132.248.182.159/acl/Herramientas/Programacion/Git/Además de Git usado de forma local, existen diversos servicios en la nube[79] 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:

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:

[79] Algunos de estos proyectos gratuitos son: Gitlab, Github, Bitbucket, Beanstalk, Launchpad, SourceForge, Phabricator, GitBucket, Gogs, Gitea, Apache Allura, entre otros.

```
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
```

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 push -u origin master
```

6.8 Programando Desde la Nube

Existen diferentes servicios Web[80] 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/

[80] 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 al navegador.

- 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/
- 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 servicio son:

- http://collabedit.com (edita código, tiene chat, no compila)
- 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 para:

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.

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, 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 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 parametros 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 empezamos 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[81] 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.

[81] 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.

> Ejemplo: En POO, podemos considerar una *Persona* como un objeto que tiene propiedades (como nombre, altura, peso, color 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 traves 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 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 parametros 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[82].

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() +
```

[82] En Java no se permite herencia múltiple, pero sí en Paython y C++, permitiendo diseños más complejos.

```
" especialidad: " + getEspecialidad() + " cedula: " + getCedula());
  }
}
```

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 donde 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 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[83], la cual

[83] 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

plantea que donde uso una clase, debería poder usar cualquier subclase de ella y que siga funcionando todo correctamente.

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 si 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 definida en ella.

A esto se lo 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).
- 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.

- Un 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:

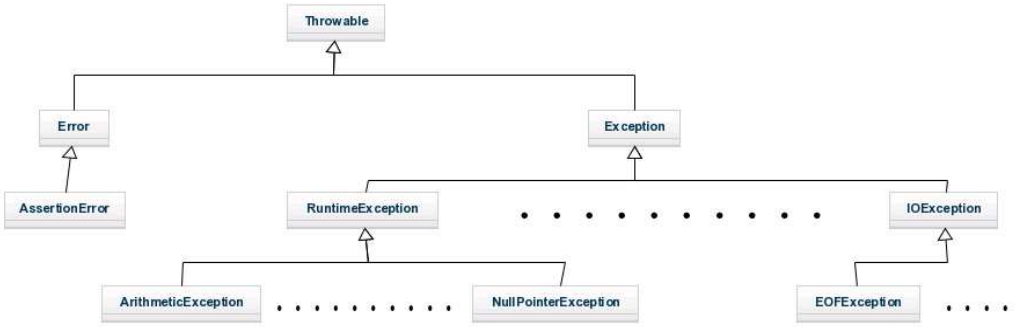


Figura 1: Jerarquía de clases para el manejo de excepciones en Java.

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 Ejemplo de Altas, Bajas y Cambios

En programación, crear, leer, actualizar y borrar (con el acrónimo CRUD[84]) 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[85] 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.

En esta sección, desarrollaremos un sistema a modo de ejemplo de Altas, Bajas y Cambios (ABC) además de Leer y Grabar los datos capturados

[84] 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").

[85] El término fue popularizado por primera vez por James Martín en su libro del año 1980 Managing the Data-base Enviroment.

—para manipular la lista de los registros usaremos un *ArrayList*[86] — 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 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;[87]
/// 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;[88]
/// Clase para manipular registros telefonicos
   public class RegTelefonico extends Registro {
   private String Nombre;
   private String Direccion;
```

[86] 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.

[87] 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;

[88] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
    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);
      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
    }
  };
```

O del catálogo de CDs, mediante:

```
import java.util.Scanner;[89]
/// 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() {
      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);
```

[89] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
        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.*;[90]
import java.util.*;[91]
/// Clase base para manipular Altas-Bajas-Cambios.
public abstract class EstructuraABC {
  /// Lista
  public ArrayList<Registro> lista;
  public EstructuraABC() {
    lista = new ArrayList<Registro>();
```

[90] Para permitir leer y grabar archivos.
[91] Para definir *ArrayList*.

```
}
/// 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;
  }
  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;[92]
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");
```

[92] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
        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();
        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;[93]
    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;
```

[93] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
      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);
   }
}
```

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;[94]
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");
    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();
```

[94] En nuestro ejemplo, usaremos la clase *Scanner*, para solicitar en consola la captura de los datos en modo texto.

```
            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 Catalogo 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.

8 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.

8.1 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.

8.2 ¿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's son:

- Varchar: Recibe cadena de palabras compuestas de letras, números y caracteres especiales.
- Int: es el principal tipo de datos de valores enteros de SQL Server. Con números enteros con o sin signo
- Date: una fecha de calendario que contiene el año (de cuatro cifras), el mes y el día.
- Time: La hora del día en horas, minutos y segundos (el valor predeterminado es 0).

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.

8.2.1 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.

Ejemplo (crear una tabla):

```
CREATE TABLE clientes;
```

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 alumnos 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 alumnos):

```
DROP TABLE alumnos;
```

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 Manipulación 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 alumnos):

```
TRUNCATE TABLE alumnos;
```

8.2.2 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 desamos 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;
```

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 solo 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 solo los datos del coche con matrícula para que la consulta devuelva solo 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 ( VAL-
UES (1) ) AS t(c)
UNION ALL
SELECT 'Peter Doe','555-2323' FROM LATERAL ( VAL-
UES (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 inserta. Lógicamente, el SELECT se evalúa antes 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';
```

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 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 devuelve la clave generada.
- Utilizando una sentencia SELECT especifica 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
```

8.3 ¿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.

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 constate 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.

8.3.1 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 deban 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.

8.3.2 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 se Datos* estará formada 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: Numericos:

- Double: punto flotante de 64 bist.
- 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 da 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 ejecucuón 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

8.4 Programando en Diferentes Manejadores de Bases de Datos

Existen diferentes paquetes para que permiten trabajar con bases de datos, en esta sección mostraremos como trabajar con algunos de ellos:

- Sqlite
- MySQL/MariaDB/Microsoft SQL Server
- PostgreSQL
- MongoDB

A continuación, mostraremos el mismo ejemplo en cada uno de estos manejadores:

8.4.1 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 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.

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 pgcli 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;
```

8.4.2 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 multiples 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 interactuación 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 multiples 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
```

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
```

además podemos instalar la interface para PHP, mediante:

```
# apt install php libapache2-mod-php php-mysql php-gd ph-
pmyadmin
```

Crear un Nuevo Usuario Para esto usamos de 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, seria 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 facilida 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
```

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 esos 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 ahors 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;
```

8.4.3 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
```

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;
```

8.4.4 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 puede que nos sientamos 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
```

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 (we will use the
same database throughout the tutorial
    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));
        }
    }
);
```

8.5 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.

8.5.1 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:data
"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 IDEN-
TITY 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), resultSet.getString(2),
    DecimalFormat.getCurrencyInstance(new Locale("es", "ES")).format(resultSet.getBi
}
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
```

sql-injection-1.sql

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;"
```

sql-injection-2.sql

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();
```

sql-injection-prepared-statement.java

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.

8.5.2 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).

8.5.3 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 especifica. 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[95] 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[96] para instalar el ambiente de pruebas como por ejemplo Kali Linux[97], dentro

[95] 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.

[96] Existen una gran variedad de manejadores de máquinas virtuales como: QEMU/KVM, VirtualBox, Vmware, Parallels, Windows Virtual PC, Gnome Box, Xen, LXC, Docker, etc.

[97] 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[98] que soporte el sistema a probar, dentro de esta máquina virtual además instalaremos el proyecto abierto para la seguridad de aplicaciones OWASP-BWA[99] (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 sistema 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.

8.6 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:

8.6.1 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 que bases de datos hay, ni seleccionar alguna de las existentes para inicar 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 es basada en Debian.

[98] Puede ser por ejemplo Linux, Apache, MySQL, PHP, Perl, Python, etc.

[99] 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;
```

8.6.2 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 que bases de datos hay, ni seleccionar alguna de las existentes para inicar 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;
```

8.6.3 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 que bases de datos hay, ni seleccionar alguna de las existentes para inicar 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;
```

8.6.4 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 inicar 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));
                }
            }
        );
```

9 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 este 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 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 necesita 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 hecho (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 documentos de texto (. Txt) son muy fáciles para leer en los ordenadores. 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 donde 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 para mostrar los datos en formato libre, 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 (http://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.

9.1 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 pueden 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 uso 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.

Esta sección se centra sólo en cómo las superficies de corte son los más dispuestos de modo que las máquinas puedan acceder a ellos directamente. Asesoramiento y orientación acerca de cómo las soluciones de sitios web y la web debe ser diseñado se puede encontrar en otros lugares.

Servicios Web cuando los datos cambian frecuentemente, y el tamaño de cada descargas es limitado, es muy importante exponer la información a través de web services. Hay muchas maneras de crear web services, pero algunos de más usados son SOAP y REST. Generalmente SOAP antes que REST. Los servicios REST son fáciles de desarrollar y entender, por eso es un standard 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.

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 caracter 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 caracter 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 caracter 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 caracter y no con un caracter 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_
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 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 observacion 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 incluyan 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 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.
- Caracter separador (separator character, en inglés): siempre debe ser "," (coma).
- Caracter calificador (quote character o enclosing character, en inglés): siempre debe ser " " " (comillas dobles).

9.2 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 contexto del dataset de que se trate.

Nombres propios Se capitalizan (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 caracter ".".
- 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.

9.3 ¿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 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.

10.1 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.

10.2 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 declara 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 últiuma 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, 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.

10.3 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.

10.4 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.

10.5 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.

11 Apéndice A: Software Libre y Propietario

Hoy en día los usuarios disponemos de dos grandes opciones para adquirir y usar el Software necesario para que nuestra computadora funcione, 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 una sola máquina y/o usuario simultáneamente.

- Por otro lado, existe el Software libre, 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.

11.1 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 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 de 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 del término "con propietario" 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.

11.2 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», y 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 en particular.

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. La Wikipedia es el ejemplo más conocido.

11.3 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 reparalos. 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 a 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 hagamos referencia a 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 la medida de lo posible. Sirva como ejemplo que 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 desarrolladores/as y personas usuarias que lo respalden, presentan niveles muy altos de seguridad, un alto grado de protección y una rápida respuesta a posibles vulnerabilidades.

11.4 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 autor (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,19 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, ya fuera 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»), como la llama en la sección 7 de ese documento. 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 en el Software de GPL sin romper leyes como la DMCA.

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 y 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 solo 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 efectividad en la práctica de las regulaciones contractuales en la GPL aún está por verse.

- 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 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 este considerar.

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 al distribuir 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 de 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.

11.4.1 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 4 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 ori-ginalmente 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.

11.5 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-.

11.5.1 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.

En México el Software libre nació en las Universidades y los Centros de Investigación. Es por eso que, desde hace cuatro décadas, los estudiantes y los profesores usan Software libre para fines didácticos y de investigación. Las universidades suelen optar por el uso de Software libre en vez de utilizar Software privativo porque satisface de una mejor manera sus necesidades de cómputo, dada su naturaleza de apertura del código y la libertad de compartir los resultados obtenidos. De forma colateral, no se tienen gastos adicionales derivados del pago de licenciamientos.

Computólogos, físicos, químicos, matemáticos, otros profesionistas y científicos utilizan Software libre como herramienta de investigación y creación. Un claro ejemplo de ello es la llamada Delta Metropolitana, que es una red de supercomputadoras que están en varios puntos de la Ciudad de México, en el CINESTAV, el IPN, la UAM y la UNAM. Esa red de supercómputo utiliza Software libre para consolidar sus recursos, hacer investigación y generar conocimiento.

Por otro lado, dadas las características del Software de código cerrado, un usuario común ignora absolutamente el contenido del mismo y por tanto si existe dentro de las líneas del código alguna amenaza contra su equipo o su información, además el usuario no sólo tiene prohibido el intentar eliminar o cambiar esa parte del código sino que puede ser perseguido por la ley por el hecho de intentar conocer si existe tal amenaza en dicho Software.

Además, en una sociedad de la información, el Software se ha convertido en una herramienta importante de productividad, y una licencia de Software privativo constituye un acuerdo o contrato entre dos sujetos jurídicos que voluntariamente acuerdan las condiciones de uso de un programa, pero el costo económico de dicha licencia es cada vez más alto y en el caso de instituciones educativas, gubernamentales y sociedades civiles es en la mayoría de los casos -por decir lo menos- prohibitivo.

Ya hace un tiempo, en varios periódicos de circulación nacional (véase [49]) fue publicado el siguiente anuncio:

> El Instituto Mexicano de la Propiedad Industrial (IMPI) anunció que en las próximas semanas dará inicio una serie de clausuras a

> negocios que utilicen Software licenciado de manera ilegal; esto como parte del acuerdo que tiene la dependencia con The Software Alliance (BSA) desde el 2002, el cual busca fomentar el uso de programas informáticos legales y disminuir el índice de piratería en el país.
>
> De acuerdo a la BSA, el porcentaje de Software ilegal utilizado en el territorio aún se ubica en un 56 por ciento, cifra considerablemente menor a lo visto en el 2005, cuando el número ascendía a más del 65 por ciento. Sin embargo, México continúa siendo uno de los mayores compradores de piratería a nivel mundial, y lo que se busca con este tipo de acciones en el 2013 es disminuir, al menos, un punto porcentual.
>
> "Si como consecuencia de una visita de inspección completa se encuentra la existencia de Software ilegal, se procede a la sanción. En el 2012 incrementaron hasta un 200% las sanciones por el uso ilegal de Software", dijo Kiyoshi Tsuru, director general en México de la BSA.
>
> Aquí es donde algunos se preguntarán, ¿y qué autoridad tiene The Software Alliance para ejecutar estas determinaciones? Pese a que cuenta con el apoyo de empresas como Microsoft, Apple, Autodesk, Adobe, Aveva, AVG, CISCO, Dell, Hewlett Packard, IBM, SAP y Symantec, lo cierto es que la BSA no puede clausurar legalmente ningún negocio. La verdadera autoridad llega en su acuerdo con el IMPI, el cual sí tiene las facultades para aplicar sanciones.

Además, la UNAM, desde junio 9 del 2009 firmó un acuerdo (véase [50]):

> Con el objetivo de fomentar la cultura de la legalidad en lo que se refiere al uso del Software entre los estudiantes, la Universidad Nacional Autónoma de México y la Business Software Alliance (BSA) firmaron un convenio general de colaboración.
>
> Mediante este acuerdo, se promoverá el uso ético de las tecnologías de la información y comunicación, y se compartirán conocimientos en materia de propiedad intelectual y Software, a fin de apoyar el desarrollo y explotación de bienes digitales en la UNAM, así como definir programas para contribuir al avance de

> un mundo digital seguro, informaron ambas organizaciones en un comunicado.
>
> El secretario general de la máxima casa de estudios, Sergio M. Alcocer Martínez de Castro, reconoció que la UNAM necesita hacer un esfuerzo en el ámbito institucional y en cada una de las entidades que la conforman, para brindar educación a sus alumnos, profesores y trabajadores en este campo.
>
> "Se pretende", destacó, "que el convenio sea operativo y que se desarrolle en cada una de las entidades con la impartición de cursos y capacitación y en una rendición de cuentas para que el Software que se utilice sea legal".
>
> El funcionario reconoció a los miembros de Business Software Alliance en Latinoamérica, y apuntó que la Universidad Nacional hará lo necesario para facilitar el uso responsable, ético y seguro del Software.
>
> Informó también que ambas partes se reunirán seis meses después del inicio de este convenio de colaboración para analizar los avances.
>
> En tanto, el director General de BSA-México, Kiyoshi Tsuru Alberú, resaltó que con la firma de este convenio podrán impulsar un plan conjunto relacionado con alta tecnología, ética y legalidad "Estamos seguros que en el mediano plazo se tendrán resultados importantes y se podrá hacer la diferencia", señaló.
>
> Por su parte, el abogado general, Luis Raúl González Pérez, comentó que la UNAM busca difundir estos valores entre su comunidad, y llegar especialmente a los estudiantes que inician el bachillerato, porque desde edad temprana es importante fomentar la cultura de la legalidad.

Ante este escenario, una alternativa viable podría ser optar por el Software libre, aunque, pese a su incipiente desarrollo es seguro que en un futuro podría alcanzar a suplir todas las necesidades básicas de los usuarios, dejando la adquisición de paquetes especializados sólo para los cursos avanzados que justifique el uso de Software privativo.

11.5.2 ¿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.

Gratis no Significa Gratuito Supongamos que todos los recursos humanos participantes en el desarrollo de un proyecto de Software libre lo hagan en 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 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 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 necesitan 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 si 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 a viva voz: "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, de 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 aquí 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 re utilice- 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 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.

11.5.3 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 rescatar 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 en la nube.

Vale la pena retomar ahora la frase esa de que «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.

11.5.4 El Código Abierto como Base de la Competitividad

En septiembre del 2021, se publico 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; y lanza una serie de recomendaciones específicas de políticas públicas destinadas a lograr un sector público digitalmente autónomo, una I+D abierta 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 coste-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 proviene 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 de la 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 Apéndice B: 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 ésto no es 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.

12.1 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.
- 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.
- 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. Si una Clase está a un paquete, entonces pertenece a un subdirectorio debajo del directorio actual de trabajo o a un subdirectorio debajo del directorio donde apunta el flan sourcepath del javac. Por ejemplo, si existe una Clase llamada com.hello.HelloWorld entonces su nombre es HelloWorld y su

paquete es com.hello. Si el flan sourcepath está configurada a c:src, entonces el archivo HelloWorld.java debe estar ubicado en el subdirectorio: c:srccomhello.

- 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.

- 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.

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.

12.2 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.

13 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:

http://mmc.geofisica.unam.mx/acl/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 20

[4] http://es.wikipedia.org/wiki/Linux 29

[5] http://es.wikipedia.org/wiki/Mac_OS 26

[6] http://es.wikipedia.org/wiki/Android 38

[7] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n) 164

[8] https://es.wikipedia.org/wiki/Python 164

[9] https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n) 164

[10] https://es.wikipedia.org/wiki/C%2B%2B 164

[11] http://www.gnu.org/philosophy/free-sw.es.html 407

[12] http://es.wikipedia.org/wiki/Software_libre 407

[13] http://www.hispalinux.es/SoftwareLibre 407

[14] http://es.wikipedia.org/wiki/Software_propietario 405

[15] Diferentes Tipos de Licencias para el Software, http://www.gnu.org/licenses/license-list.html 407, 412

[16] FSF, Free Software Fundation, http://www.fsf.org/ 407, 412

[17] GNU Operating System, http://www.gnu.org/ 407, 412

[18] LaTeX, A Document Preparation System, http://www.latex-project.org/

[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

[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/

[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

[45] Tinn-R Edit code and run it in R, http://nbcgib.uesc.br/lec/software/editores/tinn-r/en

[46] RStudio Software, Education, and Services for the R community, http://www.rstudio.com/

[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 421

[50] PCworld, http://www.pcworld.com.mx/UNAM-y-BSA-promueven-el-uso-de-software-legal/ 422

[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 2021, 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