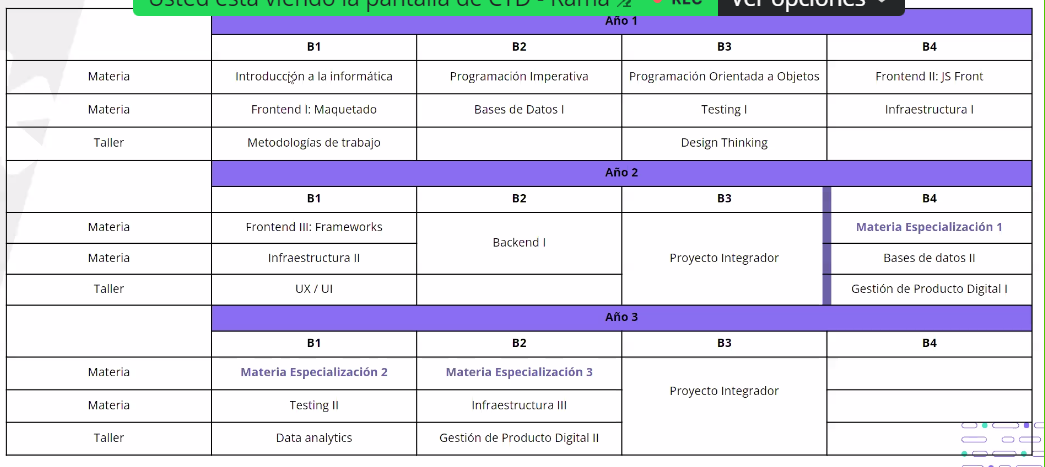
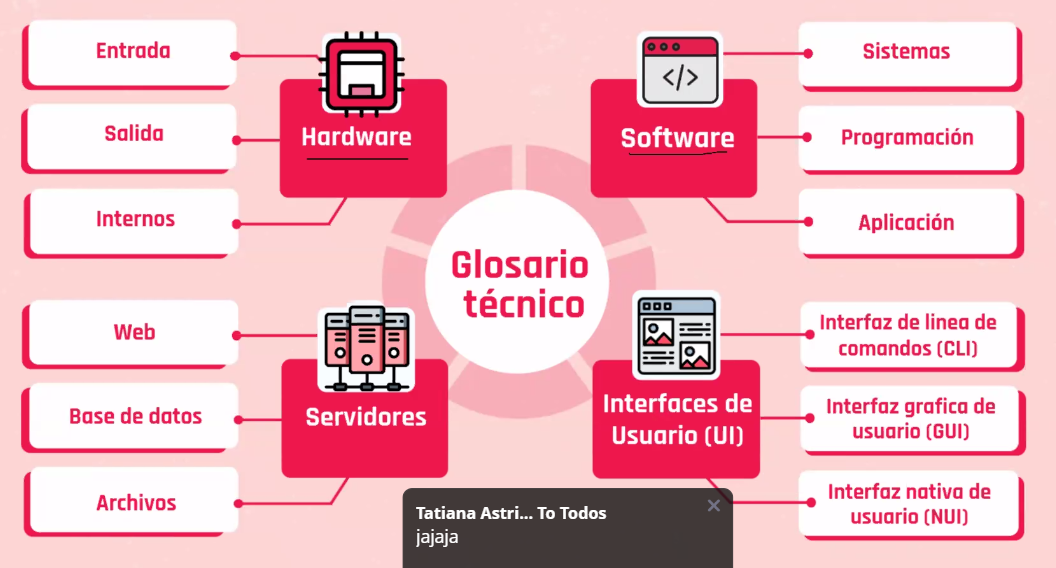
CERTIFIED TECH DEVELOPER



**INTRO A LA INFORMÁTICA:**

Todo aquello que puedo patear jaja



El software es todo aquello que me hace enojar.

* **Hardware:** El hardware o equipo se refiere a las partes físicas, tangibles, de un sistema informático, sus componentes eléctricos, electrónicos, electromecánicos y mecánicos. Ejemplos de hardware son los cables, gabinetes, periféricos de todo tipo y cualquier otro elemento físico involucrado; contrariamente, el soporte lógico e intangible es el llamado software.
* **Entrada**: Son aquellos componentes que permiten el ingreso de información, en general desde alguna fuente externa o por parte del usuario. Proveen el medio fundamental para transferir hacia la computadora (al procesador) información desde alguna fuente, sea local o remota. También permiten cumplir la tarea esencial de leer y cargar en memoria el sistema operativo y los programas informáticos, los que a su vez ponen operativa la computadora y hacen posible realizar las más diversas tareas. Entre los periféricos de entrada se puede mencionar: teclado, mouse, escáner, micrófono, cámara web, joystick, lectoras de CD, DVD o BluRay, entre otros.
* **Salida:** Son aquellos que permiten dar salida a la información resultante de las operaciones realizadas por la CPU. Los más comunes de este grupo son los monitores, las impresoras, las consolas. y los altavoces.
* **Internos**: El hardware interno es el conjunto de componentes físicos que forman parte del dispositivo principal, siendo inseparable de este. En otras palabras, cada parte del hardware interno es una pieza fundamental de cara al funcionamiento correcto del dispositivo. Ya que si faltara alguna de las partes de este, podría bien no ejecutar alguna tarea e incluso directamente no funcionar. Ejemplos de hardware interno: Placa base, CPU, RAM, GPU, HDD, SDD.
* **Software:** software o soporte lógico al sistema formal de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware. La interacción entre el software y el hardware hace operativo un ordenador (u otro dispositivo), es decir, el software envía instrucciones que el hardware ejecuta, haciendo posible su funcionamiento.
* **Software de sistema:** Desvincula al usuario y al programador de los detalles del sistema informático en particular que se use, transparentando el procesamiento referido a las características internas de: memoria, discos, puertos y dispositivos de comunicaciones, impresoras, pantallas, teclados, etc. El software de sistema le procura al usuario y programador adecuadas interfaces de alto nivel, controladores, herramientas y utilidades de apoyo que permiten el mantenimiento del sistema global. Incluye entre otros:

**•**Sistemas operativos

**•** Controladores de dispositivos

**•** Herramientas de diagnóstico

**•** Herramientas de corrección y optimización

**•** Servidores

**•** Utilidades

* **Software de programación:** Es el conjunto de herramientas que permite al programador desarrollar programas de informática, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Incluyen en forma básica:

**•** Editores de texto

**•** Compiladores

**•** Intérpretes

**•** Enlazadores

**•** Depuradores

**•** Entornos de desarrollo integrados (IDE)

* **Software de aplicación:** Es aquel que permite a los usuarios llevar a cabo una o varias tareas específicas, en cualquier campo de actividad susceptible de ser automatizado o asistido, con especial énfasis en los negocios. Incluye entre muchos otros:
* Aplicaciones para Control de sistemas y automatización industrial
* Aplicaciones ofimáticas
* Software educativo
* Software empresarial
* Bases de datos
* Telecomunicaciones (por ejemplo, Internet y toda su estructura lógica)
* Videojuegos
* Software de diseño asistido (CAD)
* **Servidor:** es una computadora capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Se pueden ejecutar en computadoras dedicadas a las cuales se les conoce individualmente como "el servidor" o en la mayoría de los casos una misma computadora puede proveer múltiples servicios y tener varios servidores en funcionamiento.
* **Servidor web:** Almacena documentos HTML, imágenes, archivos de texto, escrituras, y demás material web compuesto por datos (conocidos colectivamente como contenido) y distribuye este contenido a clientes que lo piden en la red
* **Servidor de base de datos:** Provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio.
* **Servidor de archivos:** Es el que almacena varios tipos de archivos y los distribuye a otros clientes en la red.
* **Interfaces de usuario (UI):** La interfaz de usuario es el espacio donde se producen las interacciones entre seres humanos y máquinas. El objetivo de esta interacción es permitir el funcionamiento y control más efectivo del sistema informático desde la interacción con el humano.
* **Interfaz de línea de comandos (CLI):** Interfaces alfanuméricas (intérpretes de comandos) que solo presentan texto.
* **Interfaz grafica de usuario (GUI):** Permiten comunicarse con la computadora de forma rápida e intuitiva representando gráficamente los elementos de control y medida.
* **Interfaz nativa de usuario (NUI):** Pueden ser táctiles, representando gráficamente un "panel de control" en una pantalla sensible al tacto que permite interactuar con el dedo de forma similar a si se accionara un control físico; pueden funcionar mediante reconocimiento del habla, como, por ejemplo, Siri; o mediante movimientos corporales, como es el caso de Kinect.

**Internos:** La tarea se lleva a cabo a través de los dispositivos internos del hardware: procesador, disco duro, trabajan en equipo

Un sistema operativo es un conjunto de aplicaciones que unidas a un hardware conforman un PC

Servidor: compu que brinda un servicio

**GIT BASH**

Aplicación de terminal que se utiliza como interfaz con un sistema operativo mediante comandos escritos (CLI).

* **Terminal:** programa que está presente en todos los sistemas operativos, desde ella se pueden dar órdenes al sistema.
* ¿A qué llamamos **líneas de comando**? – Interfaz que permite personalizarla, interfaz de usuario limpia y sin distracciones.
* El comando **cd** permite navegar entre carpetas
* El comando **gedit**: permite crear y editar archivos de texto
* El comando **pwd** nos permite saber el nombre de la carpeta en el que uno se encuentra situado.
* Ls: muestra los archivos de la carpeta
* Ls -t: muestra los archivos en versión offline para que los ordene según fecha de modificación.
* Rutas absolutas: toda la carpeta, se inicia con /
* Rutas relativas: se da la ruta en base a mi posición actual, no necesita iniciar con /



CICLO DE ESTADOS DE UN ARCHIVO EN GIT:

* 1. Untracked: sin seguimiento
  2. Tracked: en seguimiento
  3. Modificado: git reconoce que hubo cambios
  4. Stage área: listo para commit

git config --global user.name nombreusuario

git config --global user.email email

git rm –cached: git pasa a ignorar a un archivo y lo deja como untracked, sin seguimiento.

En Git el ciclo de vida del archivo es:

1. Creación del archivo, por ejemplo **index.html** (**aquí el archivo está en estado untracked**).
2. **Cambiamos el estado** del archivo a **seguimiento** a través del comando **git add index.html** (especificando nombre del archivo index.html para este caso). Si el archivo ya no será modificado, pasar al paso 6.
3. Si el archivo es modificado, **deja de estar en seguimiento y pasa a un estado de modificado**, ya que Git detecta que hubo cambios.
4. Guardamos el archivo con **ctrl+g** (o ctrl + s) y luego escribimos **git add index.html** (especificando nombre del archivo index.html para este caso), el archivo pasa al **stage area**.
5. El archivo está listo para realizar el **commit**, que es cuando se genera el punto histórico, a través del comando **git commit**.

**GIT:** software de control de versiones, mantenimiento de productos, permite coordinar el trabajo colaborativo, cuando hay un gran número de archivos fuente.

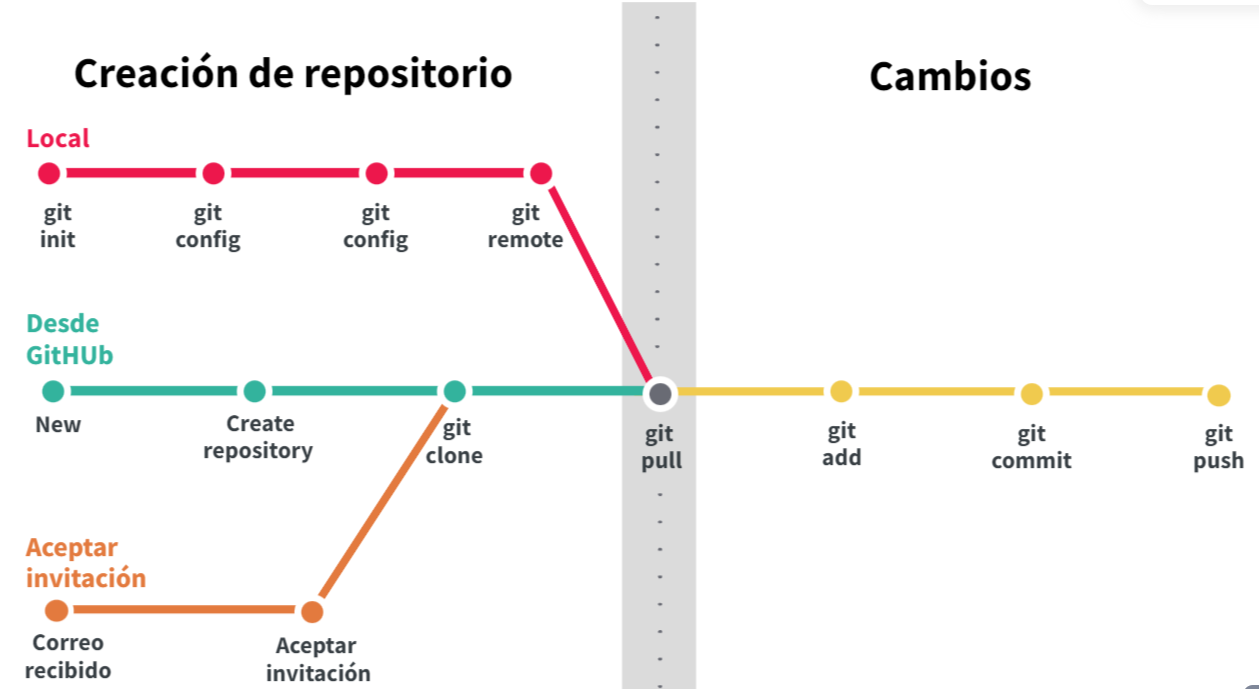
* **Repositorio local:** es el que tiene todos los archivos (que hayas guardado en él) en nuestra computadora.
* **Commits:** pequeños paquetes en donde almaceno mis archivos. son los paquetes que nos van a permitir ir haciendo un seguimiento de los cambios que vamos realizando, dado que cada uno de ellos tiene una timestamp, o fecha de creación, y un autor.

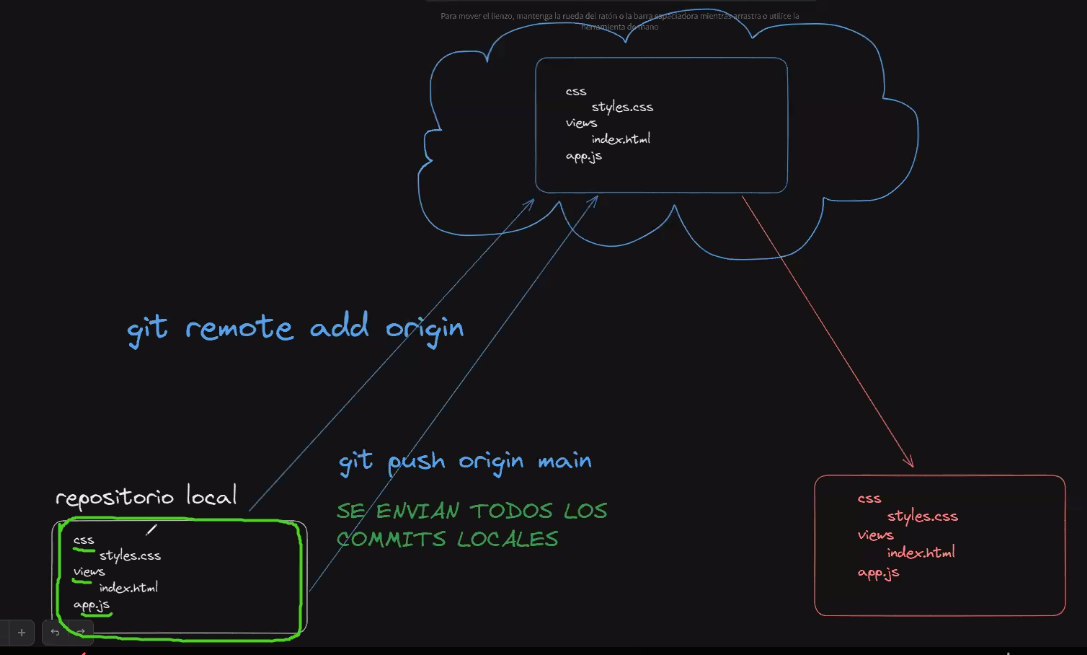
Los **commits** van a ser nuestro historial de cambios que se fueron haciendo en el proyecto.

Tienen fecha, autor y mensaje.

PASOS INICIALIZAR UN REPO LOCAL Y VINCULARSE A UNO REMOTO PARA HACER COMMITS, PUSH.

1. **Abrir una terminal de BASH**
2. **Pwr**
3. **ls**
4. **git remote rm origin: desvincular un repositorio**
5. **cd “Front-End1” para pararme en la carpeta donde quiero crear el repo local**
6. **git remote add origin** [**https://github.com/juli4888/Front-End1**](https://github.com/juli4888/Front-End1)
7. **para corroborar el repo: git remote -v**
8. **git init**
9. **git add .**
10. **git commit -m “..”**
11. **git push --set-upstream origin master**





Git reset HEAD~3: para borrar los últimos 2 commits

**Github: repo alojado en la nube**

**Git: repo local**

* **Git init: crea el repo**
* **Git config user.name “juli”**
* **Git config user.email** [**juli4888@.....**](mailto:juli4888@.....)
* **Git remote add origin http://.... : apunta al repositorio remoto.**
* **Git add .**
* **Git commit -m “……..”**
* **Git push origin master: envía los cambios al repo remoto**
* **Git status**
* **Git pull, descarga los cambios realizados desde el repo remoto**
* **Git clone http://......... : se clona el repo en nuestra máquina local**

**RAMAS**

1. git log –oneline: para ver que cambios he hecho
2. crear rama: git feature …………
3. git diff main “nombre\_rama”: diferencias entre ramas
4. git merge “mi\_rama” -🡪 para fusionarme a main
5. git push origin main
6. git checkout “mi\_rama” -🡪 pasarme a mi rama de nuevo para seguir trabajando
7. git push “mi\_rama”
8. compare and pull request

git switch o git checkout: cambiarme de rama

Fork: crea una copia de un repo en mi cuenta de usuario.

El comando para crear branch es git branch nombre\_rama

b. Para posicionarse en esa rama el comando es git checkout nombre\_rama

c. Para subir cambios a la rama creada es git push origin nombre\_rama

**GITHUB**: **Es una plataforma colaborativa que nos va a permitir llevar un control de versión  
sobre nuestro código.**

**¿Con qué comando enviar los archivos de mi repositorio a GitHub?**

git push

**En Git, ¿qué es una rama?:**

Es una línea paralela para agregar funcionalidades sin afectar a la rama principal.

**Para Git...**

el repositorio remoto se llama por defecto origin o main

**git pull origin main:** actualiza mis archivos existentes que cloné previamente en mi pc local.

Cuando se presenta un **conflicto** en el que se modificó la misma línea de código, se debe aceptar el cambio entrante (es decir el de otro compañero), sin embargo, se debe poner de acuerdo en cuánto a esa línea de código.

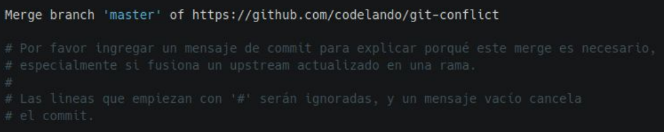
El siguiente paso después de resolver el conflicto es hacer de nuevo **git add (ese archivo), git commit -m “conflictos resueltos”, git push origin master.**

Cuando hay conflicto con otro compañero en 2 archivos diferentes, lo mejor es aceptar ambos cambios, así no se pierde el trabajo de ninguno.

**RAMAS:** Es solo una línea de desarrollo que tiene como base el repositorio, pero que no actúa directamente sobre él, o sea que no lo modifica. Al crear ramas lo que generamos son versiones del repositorio donde están los archivos del mismo más los cambios que le iremos sumando.

* git Branch <nombre de la rama>: crea una rama
* git Branch -d <nombre de la rama>: elimina la rama, no se eliminará si aun hay cambios no fusionados con main.
* git branc -D <nombre de la rama>: fuerza la eliminación aún si tiene cambios sin fusionar.
* git checkout: para moverse de una rama a otra. (no es posible moverse de rama si hay cambios pendientes, se deben eliminar deshaciendo los cambios o confirmalos haciendo un git commit).
* git push origin <Branch>
* git pull origin <Branch>

Una vez que Git termine de fusionar todos los cambios, la consola nos devolverá un mensaje como este:



Lo que nos pide Git es que agreguemos, si así lo deseamos, un mensaje que explique por qué ese merge fue necesario. Si no escribimos nada, por defecto quedará registrado el mensaje de arriba: “Merge branch...”.

Para salir de la consola sin escribir un mensaje en particular, tendremos que apretar :q , o Ctrl+x . Si nos interesa aprender cómo escribir un mensaje, podemos investigar por nuestra cuenta los editores de texto Nano y Vim.

**Evitando conflictos**

Una buena manera de evitar conflictos es mantener los commit relativamente pequeños y subir al repositorio frecuentemente. De esta manera, tenemos menos probabilidades de que ocurran conflictos y de que, si ocurren, sean pequeños.

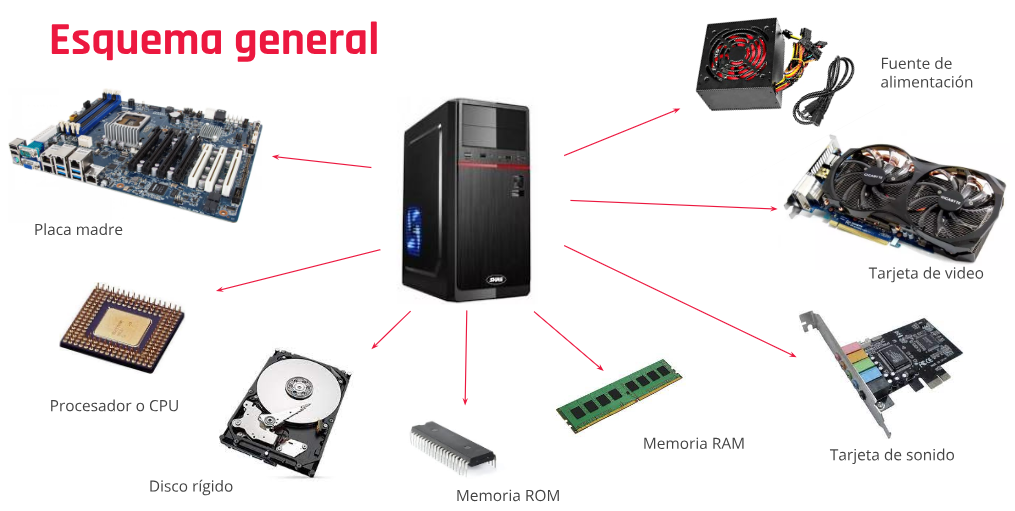
Otra manera un poco más avanzada es el uso de ramas para trabajar en paralelo a la versión principal del proyecto.

Resolvemos el conflicto tomando alguno de estos tres caminos y borramos las líneas de texto que agregó Git: <<<<<<< HEAD, ======= y >>>>>>>

3216f3fd5ca65cfd3252ae76808d8f659a715fa6

* **git remote show origin** (permite ver la información de los repositorios
* remotos)
* **git remote rename origin nombre\_nuevo** (renombra un repositorio
* remoto)
* **git fetch** (obtener los cambios, pero no aplicarlos a la branch actual)

**HARDWARE Y SOFTWARE**



* **Procesador:** unidad central de procesamiento CPU, cerebro de la pc, su función es interpretar y ejecutar instrucciones a través de operaciones básicas: aritméticas y lógicas, se ubica en la placa o tarjeta madre y posee un disipador de calor.
* **Memoria RAM:** almacena info de manera temporal, el contenido se elimina cada vez que se apaga el pc. En la memoria de acceso aleatorio se cargan todas las instrucciones que ejecuta la CPU y otras unidades del computador, además de contener los datos que manipulan los distintos programas.
* **ROM:** almacena info de forma permanente, guarda todo lo relacionado a la configuración inicial para el arranque de la máquina y funcionamiento básico.
* **Disco duro:** dispositivo de almacenamiento secundario, almacena datos de forma permanente. Es un tipo de almacenamiento masivo y permanente con mayor capacidad para almacenar datos e información que la memoria primaria (RAM) que es volátil, aunque la memoria secundaria es de menor velocidad.
* **GPU**: Una unidad de procesamiento gráfico es un coprocesador dedicado al procesamiento de gráficos para aligerar la carga de trabajo del procesador central en aplicaciones, como los videojuegos o aplicaciones 3D interactivas.
* **CPU:** La unidad central de procesamiento es el hardware dentro de un ordenador u otros dispositivos programables, su trabajo es interpretar las instrucciones de un programa informático.
* **Puente Norte:** Es el chip que controla las funciones de acceso desde y hasta el CPU, PCI-Express, memoria RAM, vídeo integrado (dependiendo de la placa) y el puente sur.
* **USB:** El bus universal en serie es utilizado como estándar para conexión de periféricos. Se puede conectar con el teclado, el mouse, la memoria USB, el joystick, el escáner, la cámara digital, el celular, el reproductor multimedia, la impresora, el módem, la grabadora de DVD externa, el disco duro externo, entre otros.

**¿Qué es una computadora?**  Es esencialmente una máquina que recibe datos, los procesa y muestra los resultados. Los mismos pueden ser almacenados, transmitidos o impresos .

**Los componentes internos de una computadora son:** Son todos los elementos físicos inseparables de la computadora. Si faltara alguno de ellos, la misma podría funcionar de manera incorrecta o no hacerlo.

**¿Qué son los periféricos?** Son dispositivos que pueden añadir funciones u operaciones, pero no son parte esencial de la misma.

**Los siguientes son dispositivos de salida:** Impresora 3D. Proyector. Monitor

**Representar información como pulsos eléctricos es lo que le permite a las computadoras procesar la información y transformarla.** Verdadero

**CPU:** ejecuta instrucciones almacenadas como números binarios organizados en la memoria principal, por lo que, cuanto más potente sea tu procesador, más rápido podrá hacer las operaciones y más rápido funcionará tu dispositivo en general. Entre otras cualidades, es también el encargado de leer, interpretar y procesar las instrucciones primero del Sistema Operativo, y después de los programas o aplicaciones que tenés instalados en el ordenador.

* Frecuencia: numero de cambios realizados en 1 segundo.
* Núcleos: single core(1 tarea), dual core(2 tareas al mismo tiempo), quad core(4 tareas…), octa core
* Subprocesos o hilos (threads): procesos simultáneos, son el flujo de control de programa, ayudan de forma directa a la manera en la que un procesador administra sus tareas, su función es que los tiempos de espera entre procesos se aprovechen mejor.
* Memoria caché: almacena temporalmente info en la RAM.

**Zócalo de CPU:** se usa para fijar y conectar el procesador sin soldarlo, lo cual permite quitar y poner diferentes modelos y familias sin tener que cambiar de placa base.

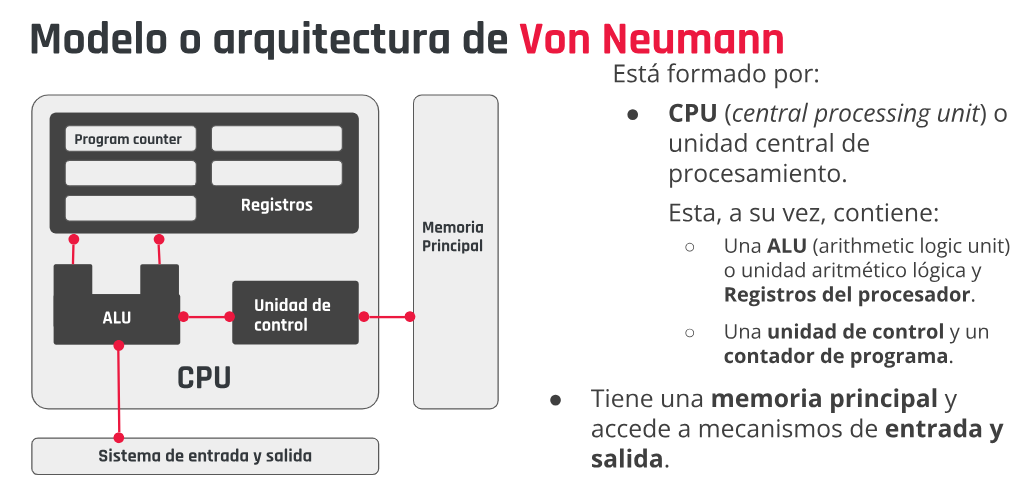
**¿Qué es el CPU?** Es el encargado de procesar todas las tareas de nuestra PC, a través de la resolución de instrucciones lógicas y matemáticas que están almacenadas en la memoria RAM.

**Seleccioná características atribuibles al CPU:** Cantidad de núcleos. Tamaño de la memoria caché. Frecuencia del reloj.

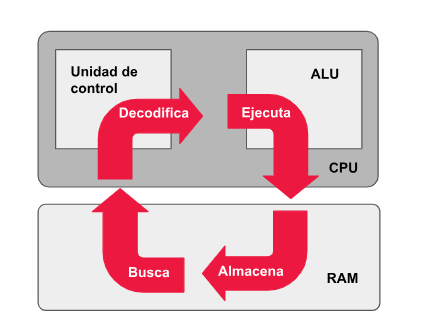
**Seleccionar el componente desde donde toma los datos y las instrucciones el CPU para realizar sus tareas: memorra RAM.**

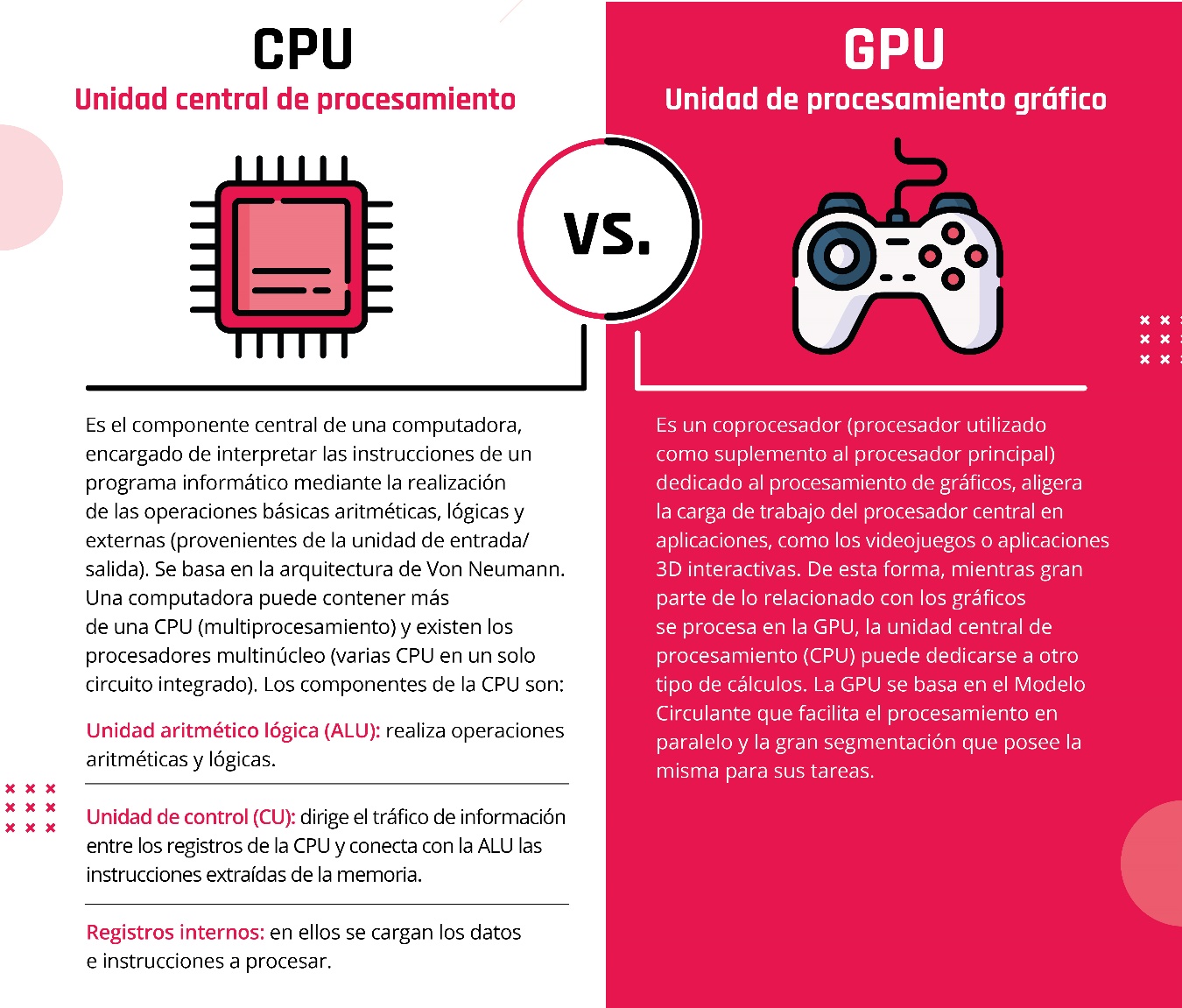
**La potencia del procesador depende de..** La frecuencia. Los núcleos. Los hilos.

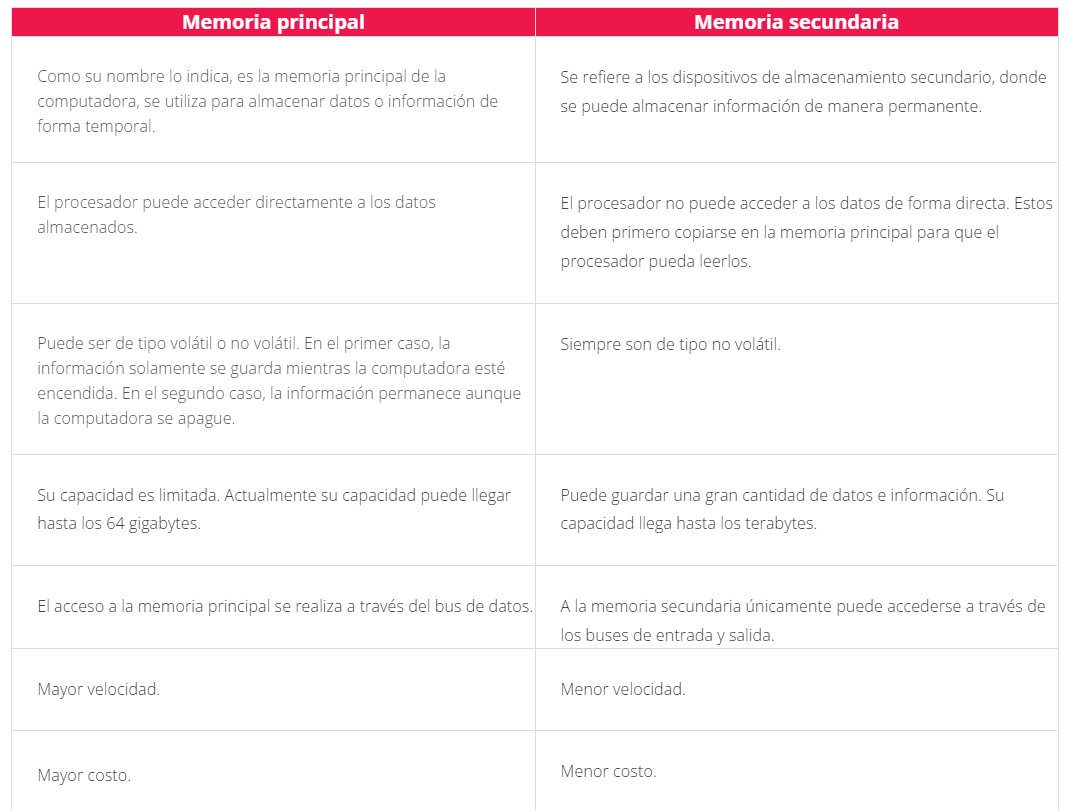
John Von Neumann fue un físico y matemático que diseñó la arquitectura básica de las compu que usamos hoy en día, la arquitectura funciona como un modelo conceptual que muestra cómo funciona una computadora.



**Ciclo de la máquina:**







MEMORIA PRINCIPAL:

* Rom: Es el acrónimo de read only memory o memoria de solo lectura. Como el nombre lo sugiere, solo puede ser leída, no escrita. Guarda las instrucciones necesarias para que la computadora pueda iniciarse.
* Caché: se sitúa entre la CPU y la memoria RAM. La CPU copia en ella los datos más relevantes que va a utilizar de la memoria RAM para acceder a ellos más rápidamente.

MEMORIA SECUNDARIA:

* Magnéticos: emplea un sistema de grabación magnética para almacenar información. Está formado por uno o más discos que giran a velocidad constante. De este tipo son los discos rígidos o disquetes.
* estado sólido: dispositivo de almacenamiento que no posee partes móviles y que permiten la escritura y lectura en múltiples posiciones en la misma operación mediante pulsos eléctricos. Tipos: discos de estado sólido y memorias.
* Ópticos: Los datos almacenados en una unidad óptica, pueden ser guardados o leídos a través de un láser. Son dispositivos ópticos los CD y DVD.

RAM: Es el acrónimo de random access memory o memoria de acceso aleatorio. Es la memoria utilizada por la CPU, en donde accede a las instrucciones y guarda los resultados inmediatos. Es volátil.

* Velocidad: Es el tiempo que tarda la RAM en recibir una solicitud del procesador y acceder a la información. La velocidad de una memoria está medida en megahertz (Mhz) o millones de ciclos por segundo.
* Capacidad: Es la cantidad de datos que se pueden almacenar en una RAM. La capacidad se mide en gigabytes (GB).
* Latencia: Es la cantidad de ciclos de reloj que transcurren entre una petición y su respuesta.
* Voltaje: El voltaje hace referencia a la energía consumida por el módulo de RAM.

|  |
| --- |
| **Single channel** |
| Para el acceso a la información en la RAM se utiliza una única señal a un ancho de banda y frecuencia determinada. |

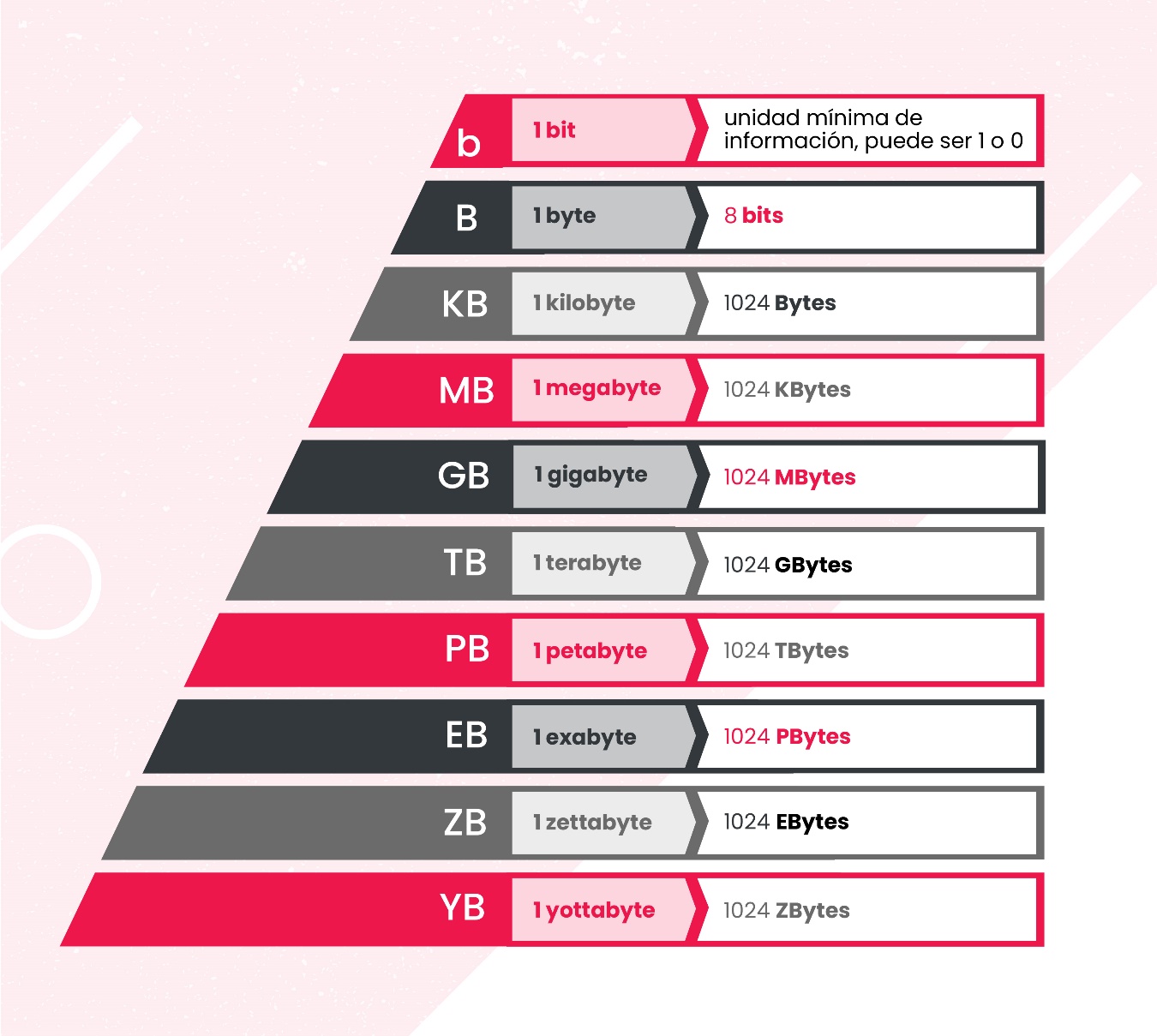
|  |
| --- |
| **Dual channel** |
| Permite el acceso simultáneo a dos módulos de memoria. Para ello, todos los módulos de memoria deben tener la misma capacidad, velocidad, frecuencia, latencia y fabricante. |

**La memoria principal posee las siguientes características:** Puede ser volátil. Su capacidad es limitada. Su velocidad es mayor que la memoria secundaria. Su costo es más elevado que la memoria secundaria.

**Para acceder simultáneamente a dos módulos de memoria RAM, estos deben ser:** De la misma capacidad, velocidad, frecuencia, latencia y fabricante.

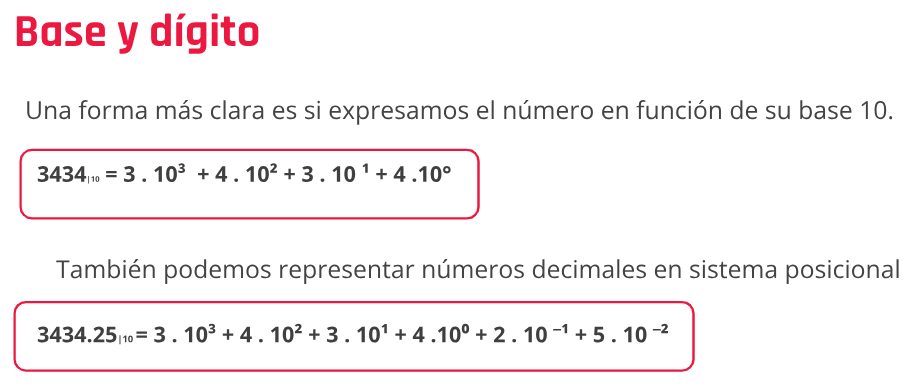
**La velocidad total de una memoria RAM se obtiene al sumar la velocidad de cada módulo. Verdadero**

**UNIDADES DE MEDIDA**



El sistema de numeración es un conjunto de símbolos y reglas de generación que permiten construir todos los números válidos del sistema.

Base: sistema de numeración

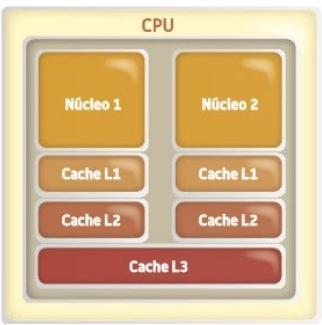


**MEMORIA PRINCIPAL:**

**REGISTROS:** Es una memoria de muy alta velocidad, se utiliza en los procesadores para acceder a info importante de manera rápida.

1. PC: program counter
2. IR: instructions register
3. MAR: memory address register
4. MDR: memory data register
5. Accumulator

**Caché:** la diferencia entre memoria caché L1, L2 y L3, L4, obedece a un orden de jerarquía establecido por cercanía al procesador, velocidad y capacidad.



**¿Qué es la memoria principal?** Es la memoria de la computadora donde se almacenan temporalmente tanto los datos como los programas que la CPU está procesando, o va a procesar, en un determinado momento. Es la memoria conocida como rom la cual utiliza el procesador para cargar su configuración inicial al arrancar.

**La memoria RAM...** Significa memoria de acceso aleatorio. Permite acceder a cualquier byte de memoria sin acceder a los bytes precedentes.

**La memoria caché es..** Un tipo de memoria que guarda datos para que las solicitudes futuras de esos datos se puedan atender con mayor rapidez.

**La memoria caché del procesador contiene los niveles L1, L2, L3 y L4.** L1 es el nivel con menor capacidad de almacenamiento. Dependiendo del procesador puede haber o no, un nivel L4.

**Cuando el CPU necesita un dato, ¿cómo lo busca?** Lo busca en su memoria caché y si no lo encuentra, se lo pide a la memoria RAM.

**QUIZ**

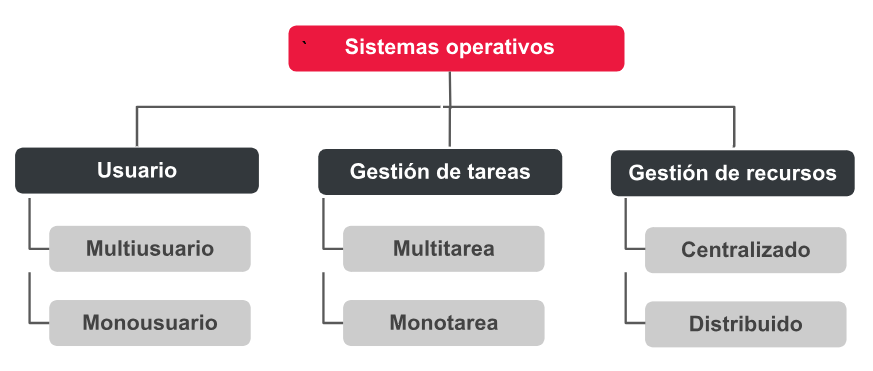
* **Los discos sólidos funcionan con magnetismo.** Falso
* **¿Qué es un disco M2?** Una variante del disco solido.
* **Los Blu-ray solo poseen una capa de escritura.** Falso.
* **¿Dónde se produce el cuello de botella en los discos SSD?** En el bus sata.
* **¿De qué material están compuestas las cintas magnéticas?** Óxido de hierro

**SISTEMAS OPERATIVOS**

Es el soporte lógico que controla el funcionamiento del equipo físico. Conjunto de programas y funciones que ocultan los detalles del hardware ofreciendo al usuario una vía sencilla y flexible de acceso al mismo.

Recursos administrados por el sistema operativo:

* Gestionar la memoria de acceso aleatorio y ejecutar las aplicaciones designando los recursos necesarios.
* Administrar la CPU
* Direccionar entradas y salidas
* Administrar la info
* Dirigir autorizaciones de uso
* Administrar los archivos

Los sistemas operativos **monotarea** se caracterizan por poder realizar solo una tarea a la vez sin que se pueda interrumpir. Estos son los sistemas operativos más primitivos, como DOS. Por ejemplo, si queremos imprimir algún archivo en estos sistemas operativos, no vamos a poder realizar ninguna otra tarea hasta que la computadora imprima y pueda recibir otra instrucción.

Los sistemas operativos **multitarea** nos permiten realizar varias tareas al mismo tiempo, estos son muchos más comunes hoy en día.

Los sistemas operativos **monousuarios**, como su nombre lo indica, soportan solo a un usuario a la vez, no importa cuántos procesadores tenga la computadora o cuántas tareas realice el usuario, solo podrá dar servicio a uno. Ejemplos podrían ser las primeras versiones de Windows para computadoras domésticas, como Windows 1.0.

Los sistemas operativos **multiusuario** pueden dar servicio a varios usuarios al mismo tiempo, ya sea por medio de terminales conectadas a la computadora o por sesiones remotas en una red de comunicaciones. Ejemplos de estos sistemas operativos pueden ser Unix, Linux o Solaris.

La estructura **monolítica** está constituida por un solo programa compuesto de una serie de rutinas entrelazadas entre sí, de tal forma que pueden comunicarse entre ellas. Estos sistemas operativos suelen estar hechos a medida, por lo que son muy rápidos, pero no tienen flexibilidad para soportar diferentes tipos de aplicaciones.

**Jerárquica:** A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software del sistema operativo, donde una parte del sistema contenía subpartes y estaba organizado en forma de niveles. A este sistema operativo se le conoce como estructura jerárquica, ya que se subdivide en capas o anillos, perfectamente definidas y con una clara interfaz con respecto al resto de los recursos.

Los sistemas operativos tipo **máquina virtual** separan dos conceptos que suelen estar unidos en el resto de sistemas: la multiprogramación y la máquina extendida. El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes

El sistema operativo **cliente-servidor** sirve para toda clase de aplicaciones, por lo tanto, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales. La idea es mantener la visión que tiene un usuario de una computadora personal, pero la red le permite compartir el espacio del disco o la impresora con el fin de economizar recursos.

**¿Un sistema operativo es el soporte físico que controla el funcionamiento del equipo lógico? Falso**

**Los recursos fundamentales que administra el sistema operativo son:** El procesador. La memoria. La información.

**El sistema operativo permite que varios usuarios ejecuten simultáneamente sus programas es:** Multiusuario.

**¿Cuál de la siguiente lista es un sistema operativo para servidores?**

* Red Hat., Microsoft Windows Server., Debian

**Arquitectura cliente-servidor**

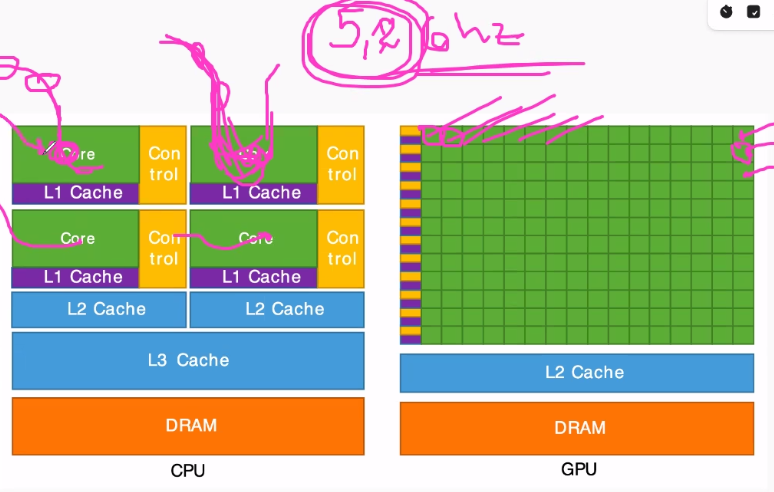
La arquitectura cliente-servidor persigue el objetivo de procesar la información de un modo distribuido. De esta forma, pueden estar dispersos en distintos lugares y acceder a recursos compartidos.

Además de la transparencia y la independencia del hardware y del software, una implementación cliente-servidor debe tener las siguientes características:

* Utilizar *protocolos asimétricos*, donde el servidor se limita a escuchar en espera de que un cliente inicie una solicitud.
* El acceso es transparente, multiplataforma y multiarquitectura.
* Se facilitará la *escalabilidad*, de manera que sea fácil añadir nuevos clientes a la infraestructura —escalabilidad horizontal— o aumentar la potencia del servidor o servidores, aumentando su número o su capacidad de cálculo —escalabilidad vertical—

**Cliente:** De manera genérica, nos referimos a una computadora, normalmente con prestaciones ajustadas, sin embargo, en entornos cliente-servidor, se utiliza el término front end, ya que es un proceso que solicita los servicios del servidor a través de una petición del usuario.

Un proceso cliente se encarga de interactuar con el usuario, por lo que estará construido con alguna herramienta que permita implementar interfaces gráficas (GUI).



* **Sistemas operativos Open source (a voluntad del usuario):** Ubuntu y Red hat
* **Propietary software:** no permite modificaciones, ej: Windows

**Servidor:** Genéricamente un servidor es un ordenador, pero con prestaciones elevadas. Sin embargo, desde este enfoque, un servidor es un proceso que ofrece recursos y servicios a los clientes que lo solicitan (back end).

Según el tipo de servidor implantado, tendremos un tipo de arquitectura cliente-servidor diferente. A su vez, debido a que los programas y datos se encuentran centralizados, se facilita la integridad y el mantenimiento.

**Middleware (procesa el pedido para que sea comprendido por el servidor):** Es la parte del software del sistema que se encarga del transporte de los mensajes entre el cliente y el servidor y facilita la interconexión de sistemas heterogéneos sin utilizar tecnologías propietarias. Por lo cual, se ejecuta en ambos lados de la estructura.

El middleware permite *independizar* a los clientes y a los servidores. Además, ofrece más control sobre el negocio, debido a que permite obtener información desde diferentes orígenes —uniendo tecnologías y arquitecturas distintas— y ofrecer de manera conjunta. Otra característica es que los sistemas están débilmente acoplados ya que interactúan mediante el envío de mensajes.

**PASOS o flujo de comunicación con el servidor:**

1. Inicio del servidor
2. Cliente realiza una petición
3. El servidor recibe la solicitud, hace las verificaciones necesarias y si todo es correcto la procesa
4. El servidor responde con el resultado solicitado
5. El middleware realiza las comprobaciones en caso de ser necesarias, si es el resultado esperado se muestra al usuario final o cliente.

**Tipos de Cliente:**

* Navegador web
* Navegador móvil
* Aplicación de escritorio o mobile

**Datacenter:** almacenan los servidores de forma segura, protegidos de problemas energéticos o climáticos.

**SISTEMAS OPERATIVOS DE RED MÁS UTILIZADOS:**

* Windows
* Linux

**Características fundamentales de un sistema operativo:**

* **Soporte de red:** Es indispensable que tengan un soporte completo para poder brindar conectividad.
* **Amplia compatibilidad con el hardware:**   
  Un punto fundamental para el aprovechamiento pleno de las características del servidor es que el S.O. sea capaz de exprimir al máximo las características técnicas del hardware en donde se ejecuta, es por ello que se debe priorizar el uso de S.O. actualizados y con un soporte importante de controladores. Por ejemplo, que nuestros controladores permitan acceder a características avanzadas de gestión de discos rígidos, con el propósito de realizar arreglos redundantes para tener mayor velocidad y tolerancia a fallos.
* **Seguridad:** Teniendo en cuenta el rol que cumple el servidor, es de vital importancia que el S.O. instalado sea seguro; eso implica no solo que este actualizado con todos los parches/actualizaciones, sino que además debe tener aplicadas políticas estrictas de acceso para prevenir accesos no autorizados o ataques.  Adicionalmente esa seguridad debe reforzarse con la instalación de Firewalls (por software o hardware) y antivirus. En este ítem también debemos incluir el respaldo de la información, ya sea por medio de herramientas que el propio S.O. nos ofrezca o instalando software externo, con el propósito de tener la menor pérdida de datos posible en caso de fallos fatales.
* **Tolerancia a fallos:** Debemos priorizar S.O. que en su arquitectura permita una tolerancia a fallos, ya sea mediante la generación de granja de servidores, que interconectados, operen como una gran unidad de proceso, dando la posibilidad que ante la caída de uno de los integrantes de la granja, otro puede tomar su rol y responsabilidad.
* **Manejo diferente del hardware:** Debido al diferente propósito, los S.O. de estaciones de trabajo no pueden aprovechar todo el hardware disponible, como, por ejemplo, el manejo de memoria RAM —teniendo el caso de Windows 10 64-bit que puede manejar 6TB de RAM, mientras que Windows Server 2019 alcanza los 24TB—.
* **Características soportadas:** Hay funcionalidades que nativamente un S.O. de estación de trabajo no es capaz de brindar, ya que en su versión de kernel están limitadas o deshabilitadas —casos tales como la virtualización en algunas versiones de Windows 10—.
* **Soporte:** Algo muy importante a tener en cuenta es, cuando nuestro negocio o aplicación depende de un S.O., es el soporte por parte del fabricante/desarrollador. En el caso de los S.O. de estación de trabajo, el soporte/cobertura que tenemos es para un uso específico, si sobre esa base quisiéramos desplegar una arquitectura, por ejemplo, de servidor web, si bien es probable que nos funcione, vamos a carecer de soporte técnico ya que el fabricante nos indicará que para ese propósito esta la version “Server” del producto.

**% de utilización:**

* **75% Windows server: 2003-2019**

**WINDOWS**

VENTAJAS:

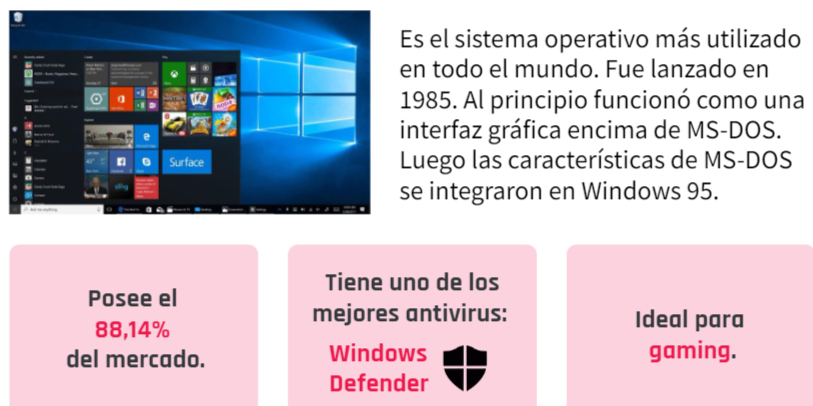
* **1. Integración:**casi todas las aplicaciones del mercado son compatibles con Windows.
* **2. Soporte:** existe mucho soporte, tanto oficial como en diferentes foros y webs.
* **3. Facilidad de uso:**es muy fácil de usar, sumado a eso, la mayoría de nosotros creció utilizando alguna versión de Windows.

DESVENTAJAS:

**1. Seguridad:** al ser el S.O. más utilizado, es el que más malware atrae.

**2. Costo:**es un sistema operativo costoso.

**3. Tamaño:** algunas versiones de Windows ocupan mucho espacio de disco.



**MAC**

* **25% unix: GNU Linux, FreeBSD, macOsServer**

VENTAJAS:

**1. Estabilidad:**debido a la minuciosa optimización entre su hardware y software el S.O. casi no presenta fallos.

**2. Seguridad:**este S.O. es reconocido por su seguridad y menos vulnerabilidad ante los malware.

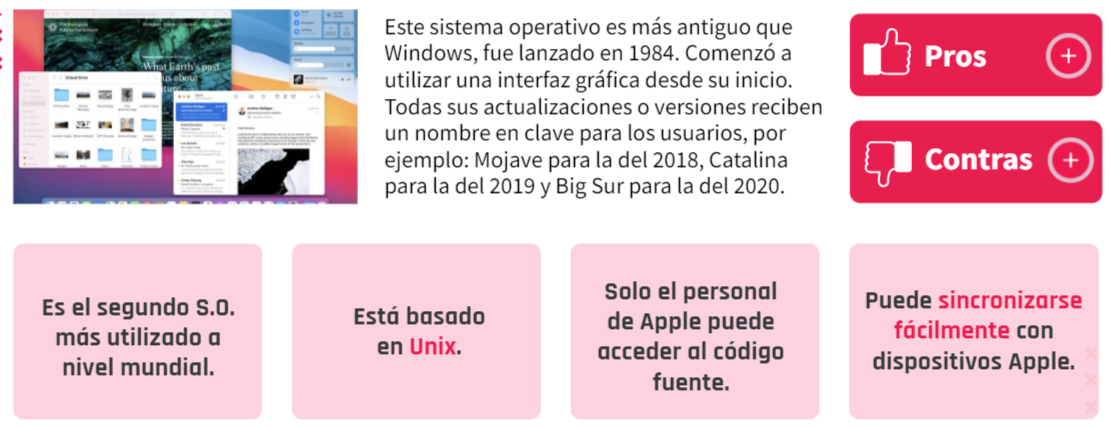
**3. Diseño:**tiene un diseño muy estético y fácil de utilizar.

DESVENTAJAS:

**1. Costo:**comprar una MAC suele ser muy costosa.

**2. Compatibilidad:**a pesar que fue aumentando con el tiempo, todavía existen muchas aplicaciones no compatibles con Mac OS.

**3. No apto para Gamers:** en estos sistemas no corren la mayoría de los últimos juegos.



**LINUX**

VENTAJAS:

**1. Costo:**Cualquiera puede usarlo, modificarlo y redistribuirlo sin costo.

**2. Seguridad:**Los sistemas operativos basados en Linux poseen pocos virus o estos son inexistentes.

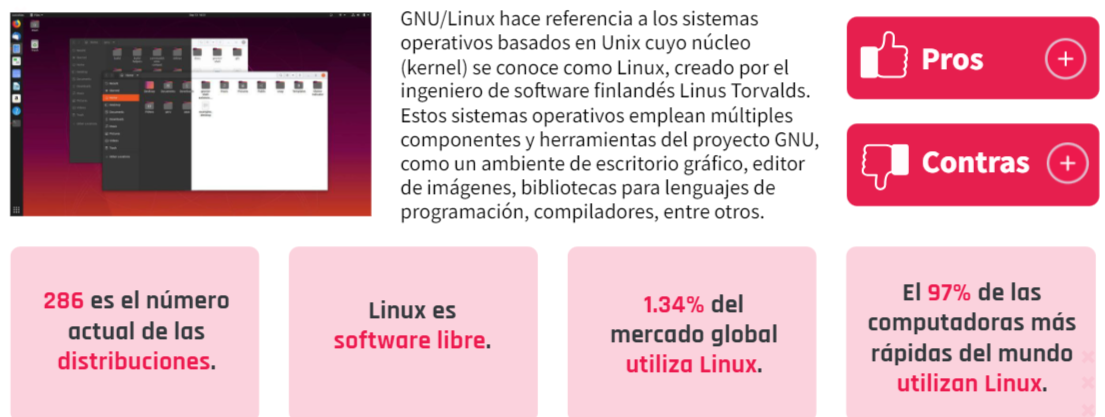
**3. Estabilidad:**Es muy estable, tanto que es muy usado en servidores.

DESVENTAJAS:

**1. Complejidad:** a la mayoría se le hace difícil entenderlo y manejarlo, incluso puede volverse problemático instalar un simple programa.

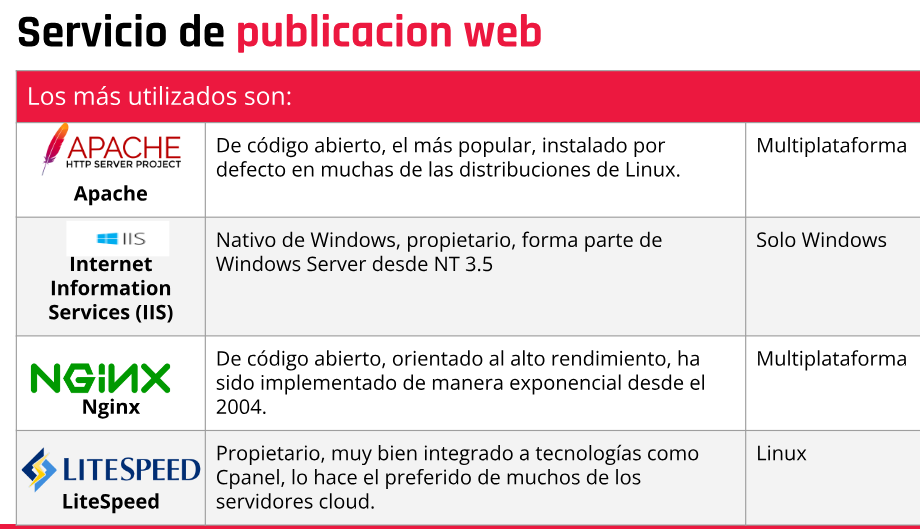
**2. Compatibilidad:**muchos de los programas más utilizados en el mercado no funcionan o necesitan parches y conversiones para funcionar.

**3. Portabilidad:** la portabilidad de las distribuciones de Linux no es prioritaria para muchos desarrolladores de software y de hardware.



Servicios más utilizados de los S.O.:

* Publicación web: despachar el contenido de un website al usuario
* Base de datos
* Correo electrónico
* Archivos
* Red
* Dominio



BASES DE DATOS:

RDBMS: relational database management systems, es un tipo de software de servidor que permite la organización de la información mediante el uso de tablas, índices y registros.

No-SQL: bases de datos no relacionales.

tareas: selección, actualización y borrado de datos, dependiendo de los privilegios de usuario.

* MySQL: Utilizadas en desarrollo web y de aplicaciones que utilizan lenguajes como PHP, Ruby o Python.
* PostgreSQL: open source y orientado a objetos, ofrece gran estabilidad, robustez y velocidad.
* Microsoft SQL server: ofrece compatibilidad nativa con ASP/ASP.NET así como con toda la suite de desarrollo de app de S.O. Windows.
* Mongo DB: es software libre, no guarda datos en tablas sino en estructuras BSON dinámicas , acceso rápido y fácil.

Correos: es una de las apps mas populares en usar el protocolo TCP/IP, que permite en cuestión de seg comunicarnos.

* MTA (mail transport agent): transferir emails de un host a otro.
* Qmail
* Exim
* Postfix
* Moicrosot Exchange server
* Courier
* Cyrus
* MDA (mail delivery agent): recibe correo de un MTA y lo lleva al inbox de la casilla de correo que previamente se comunicó con el servidor POP o IMAP.
* Dovecot
* Procmail
* Maildrop

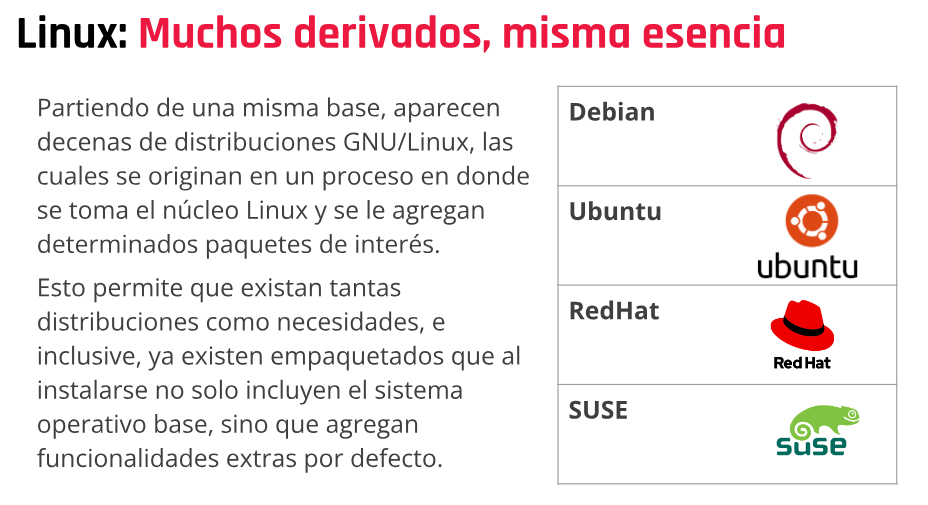
Servicio de archivos:

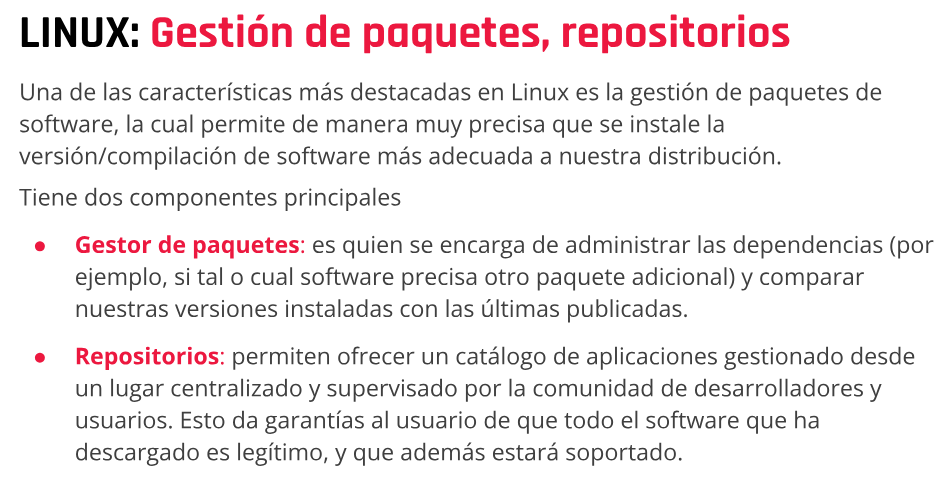
* CIFS/Samba (Linux)
* NTFS share (nativo Microsoft)
* NFS (desarrollado por sun y originario de los sistemas UNIX).

**Servicio de red:**

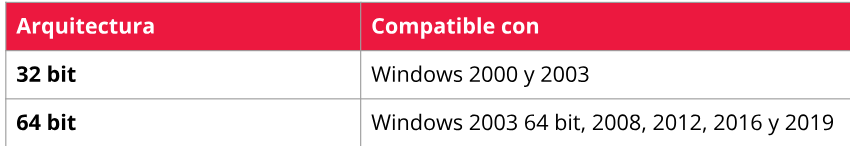
Tomando como base sistemas Linux se implementan una serie de paquetes-softwares que dan como resultado routers, firewalls o proxies.

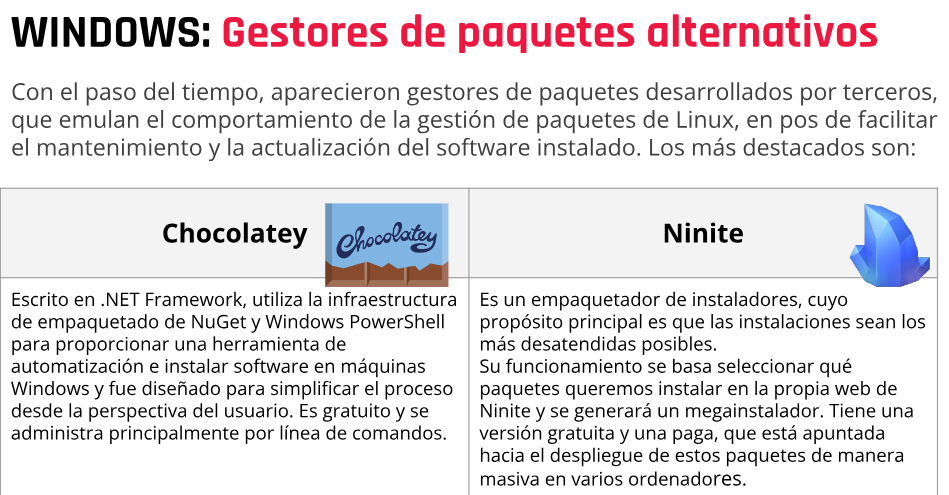
* PFSense: stack basado en BSD, sirve de router, VPN, firewall
* OPNSense: derivado del rpimero y de monowall, routers corporativos
* DD-WRT: basado en Linux, routers de hogar





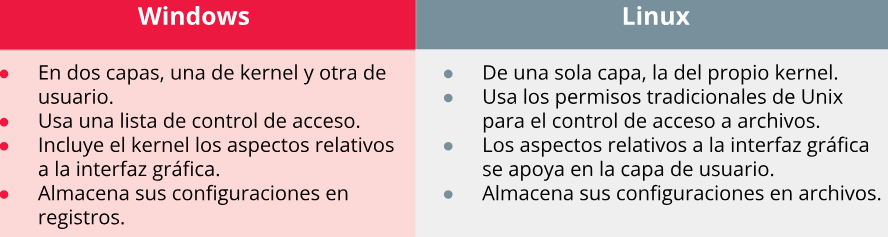




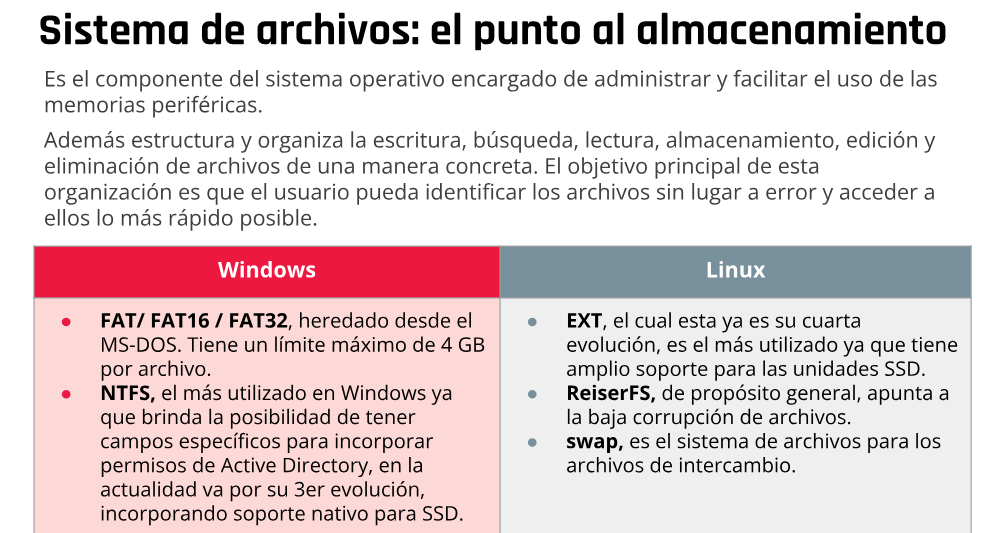


Diferencias técnicas Windows y Linux:

* Kernel: es el software que constituye parte fundamental del O.S., se ejecuta en modo privilegiado, le da acceso seguro a los distintos programas hacia el hardware de la computadora.



* Sistemas de archivos: encargado de administrar y facilitar el uso de las memorias periféricas, estructura y organiza la escrituta, búsqueda, lectura, almacenamiento, edición y eliminación de archivos
* Niveles de ejecución





**POWERSHELL**

Vimos que la GUI es la manera más extendida de administrar servidores Windows. Hablamos de sus ventajas en lo que a curva de aprendizaje se refiere y sus pocos cambios con el paso del tiempo, lo que hace que una vez familiarizados con estas consolas podemos dominar en varios aspectos cualquiera de las versiones de Windows Server.

Las consolas de administración, al ser componentes específicos a determinadas funciones —administración de dispositivos, discos físicos, usuarios de dominio, etc.— en determinados escenarios, tales como la automatización de procesos, pueden no ser suficientes y es allí cuando nos surge la necesidad de administrar nuestro Windows con comandos, tal como lo haríamos en Linux.

Windows siempre tuvo una interfaz de línea de comandos, pero limitada en funciones y apoyada en un lenguaje casi obsoleto como Batch.

**EL LENGUAJE**

El lenguaje de la consola incluye declaración de variables, variables especiales predefinidas, operadores matemáticos, incluyendo igualdades y desigualdades numéricas, manejo de vectores, comparación entre estos, operadores de asignación, vectores asociativos —hashtables—, valores booleanos, ciclos y ruptura de los mismos, operadores de expansión para simplificación de ejecuciones complejas —creación de vectores por medio de llamados a procedimientos, creación dinámica de vectores, etc.—; comentarios, operadores de comparación binaria, caracteres de escape, orden de ejecución, ciclos del tipo "foreach", creación de procedimientos y funciones, creación de filtros, estructuras condicionales complejas —if/then/else/elseif/switch—, operador de invocación dinámica del contenido de variables.

**LOS COMANDOS**

Los comandos de PowerShell llamados cmdlets —por command-applets—. Están estructurados de la siguiente manera: un verbo y un nombre separados por un guión (-): verbo-nombre. Vamos a ver un ejemplo, para ello abrimos nuestra consola de

Powershell y ejecutamos:

Get-Command

El verbo —evidentemente en inglés— describe la acción a realizar sobre el nombre. En el anterior ejemplo, recuperamos —Get— los comandos —Command—. En este caso, el comando nos devolverá una lista de los comandos disponibles de Powershell. Con PowerShell encontramos numerosos verbos genéricos tales como Get, Set, Add, Remove, entre otros, que se combinan con diferentes nombres como Path, Variable, Item, Object, Computer, etcétera. Los nombres que constituyen los comandos están siempre en singular y esto es válido también para los parámetros. Por lo tanto, es posible, mezclando verbos y nombres, acordarse fácilmente de un buen número de comandos. Hay que tener en cuenta que los comandos, así como sus parámetros asociados, pueden escribirse indistintamente en mayúsculas o en minúsculas. El analizador sintáctico PowerShell no es case sensitive.

Bash

Ya vimos que para poder interactuar con los S.O. teníamos a la interfaz gráfica (GUI) o la interfaz de línea de comandos (CLI), y supimos que la primera es más utilizada en Windows y la segunda, en Linux.

La CLI de Linux, además de ser muy potente, tiene de manera nativa la interpretación de lenguaje llamado BASH, o Bash Shell, el cual tiene como propósito la creación de scripts que facilitan la administración del sistema.

Las funciones Bash pueden:

* Eliminar tareas repetitivas.
* Ahorrar tiempo.
* Proporcionar una secuencia de actividades bien estructurada, modular y formateada.
* Con scripts, podemos proporcionar valores dinámicos a comandos usando argumentos de línea de comando.
* Puede simplificar comandos complejos en una sola unidad en ejecución.
* Una vez creada, se puede ejecutar cualquier cantidad de veces por cualquier persona. Construye una vez y ejecuta muchas veces.
* Los flujos lógicos se pueden construir utilizando funciones Bash.
* Las funciones Bash se pueden ejecutar al inicio del servidor o agregando un cron job programado.
* Los comandos pueden ser depurados.
* Tener comandos de shell interactivos.

Un ejemplo sencillo de una función Bash sería

#!/bin/bash

testfunction

testfunction(){

echo "My first function"

}

El cual como resultado de ejecución devolverá en pantalla

My first function

Las funciones bash pueden aceptar cualquier número de parámetros. El siguiente ejemplo acepta dos parámetros:

#!/bin/bash

testfunction(){

echo $1

echo $2

}

Los scripts bash soportan:

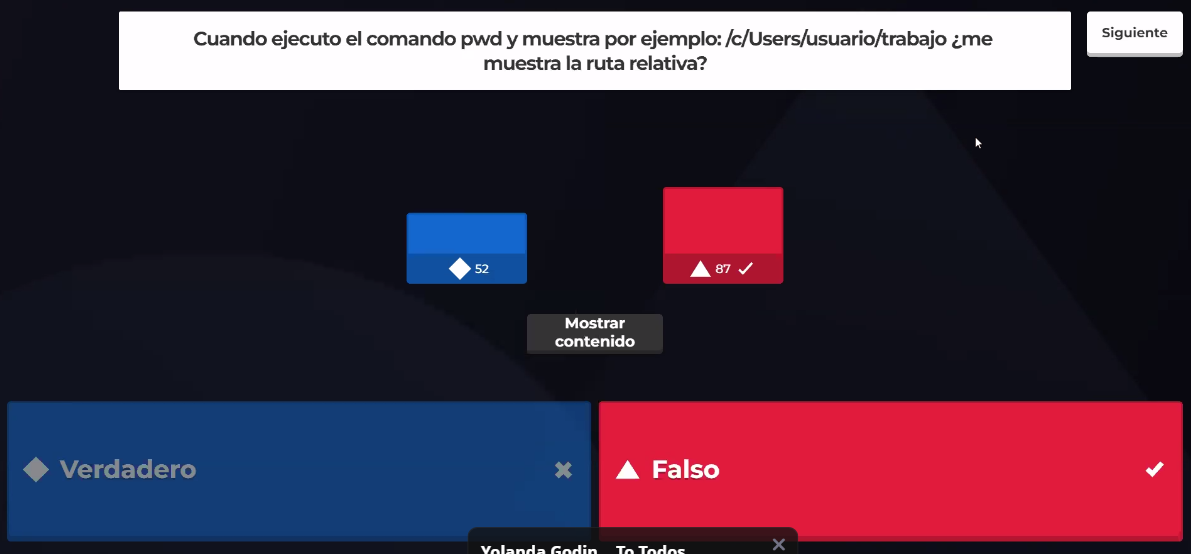
* Bucle *while*
* Bucle *for*
* Declaración *if*
* Elemento lógico *and*
* Elemento lógico *or*
* Declaración *Else If*
* Declaración *case*

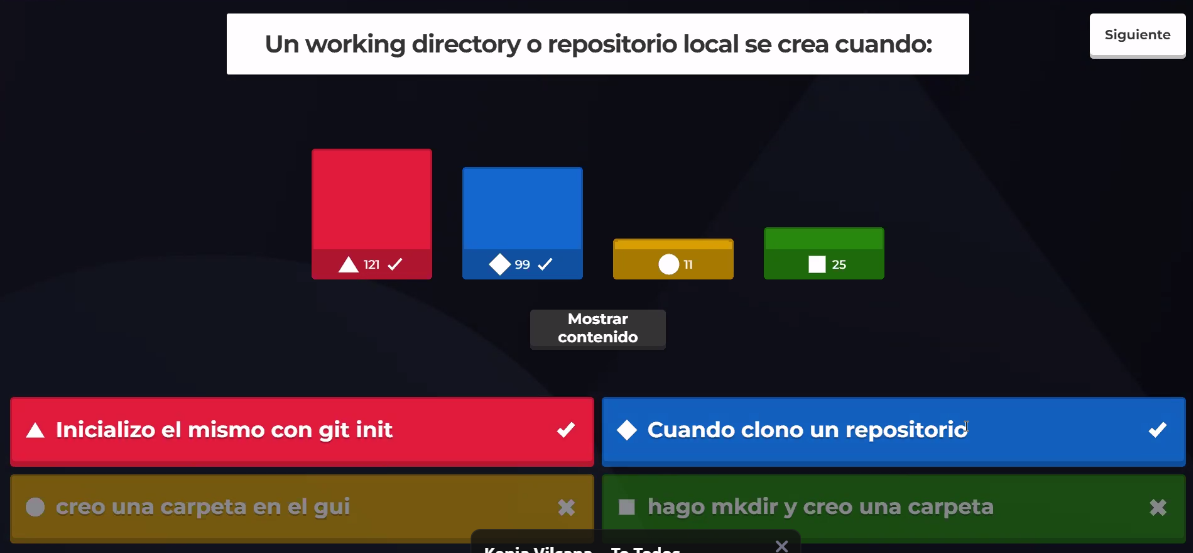














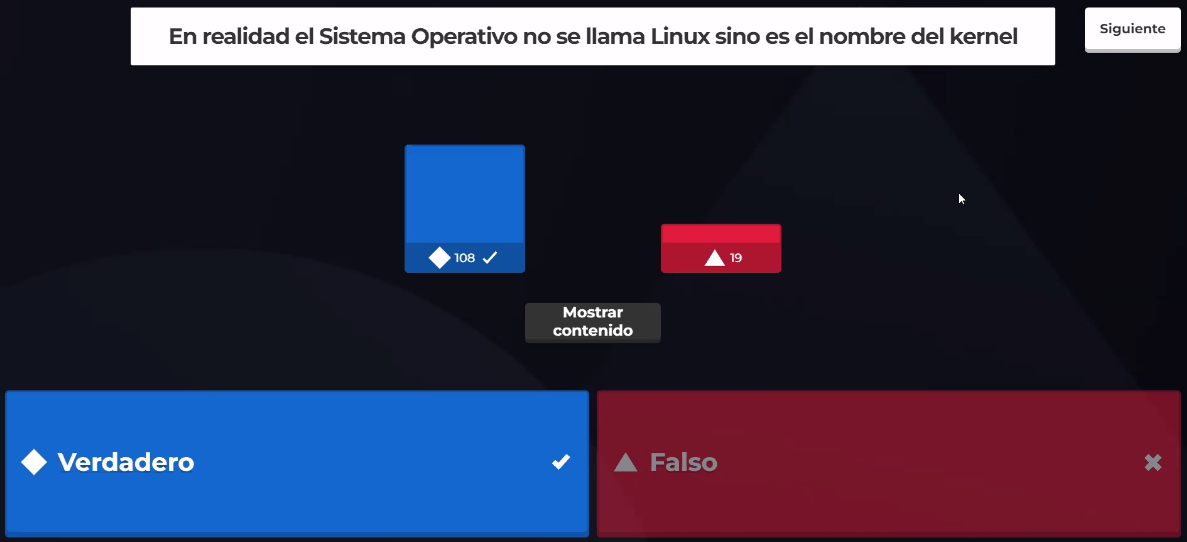
* **Git Branch -M master: para cambiar de main a master**



**El repositorio remoto se llama origin.**









**La memoria caché está dentro del procesador.**





**History: historial de comandos en la terminal**