



ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



Día, Fecha:	Miércoles, 19 de febrero
Hora de inicio:	4:30 – 6:10 pm

Introducción a la Programación y Computación 1 Sección G

Max Rodrigo Durán Canteo



Clase 5

MVC y Manejo de Hilos



Agenda

- MVC
- Aplicaciones
- Flujo de Trabajo (MVC)
- Ventajas del MVC
- Aplicaciones
- Concurrencia
- Paralelismo
- Sistemas Operativos y Procesadores
- Procesos
- Hilos



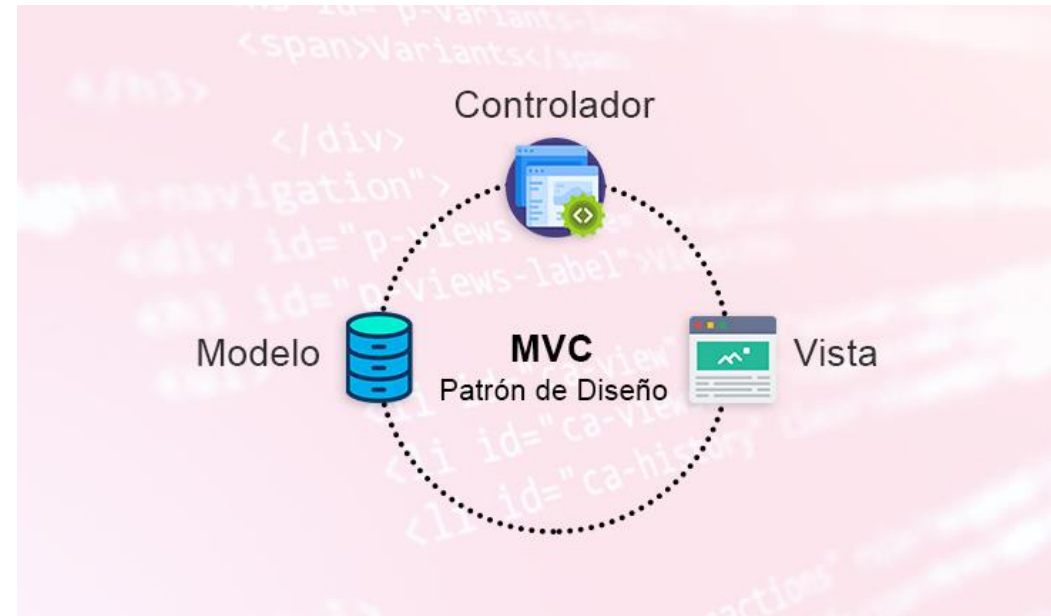
MVC

Modelo Vista Controlador

Definición, ¿Qué es?

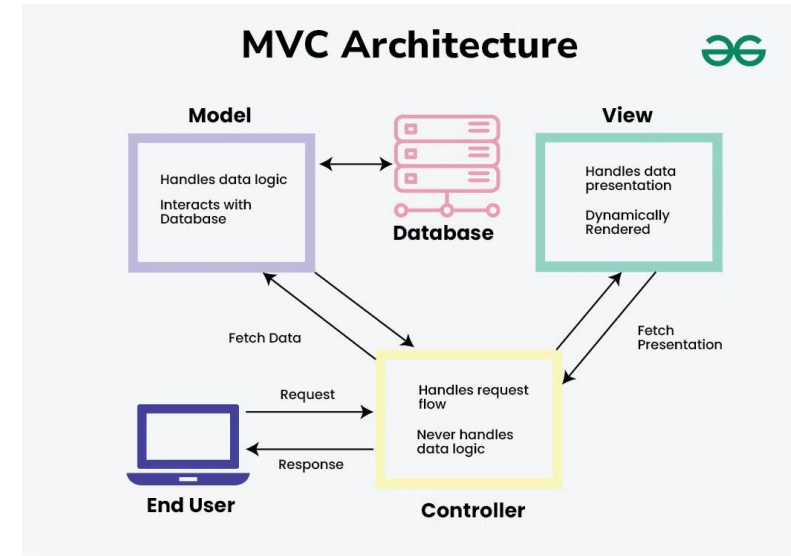
MVC

El patrón MVC (Model-View-Controller) es un patrón de diseño arquitectónico que organiza una aplicación en tres componentes principales: Modelo (Model), Vista (View) y Controlador (Controller). Su objetivo es separar las responsabilidades para facilitar el mantenimiento, la escalabilidad y la reutilización del código.



Componentes

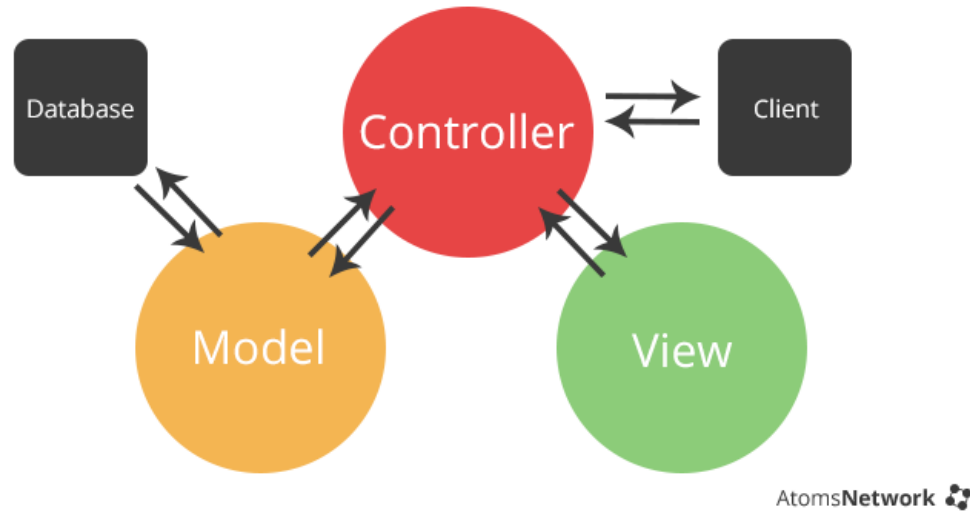
- **Modelo:** Representa los datos y la lógica de negocio de la aplicación. Maneja las reglas de negocio, la lógica de la aplicación y la gestión de los datos persistentes (por ejemplo, bases de datos).
- **Vista:** Es responsable de la presentación de los datos al usuario. Es lo que el usuario ve e interactúa (la interfaz de usuario). La vista obtiene datos del modelo y los muestra de una forma que sea comprensible para el usuario.
- **Controlador:** Actúa como intermediario entre la vista y el modelo. Recibe las entradas del usuario desde la vista, las procesa y ejecuta las acciones necesarias, como interactuar con el modelo o actualizar la vista.



Como Aplicarlo

El enfoque MVC permite manejar de manera eficiente cómo los datos se muestran al usuario y cómo las interacciones del usuario afectan la aplicación.





Modelo

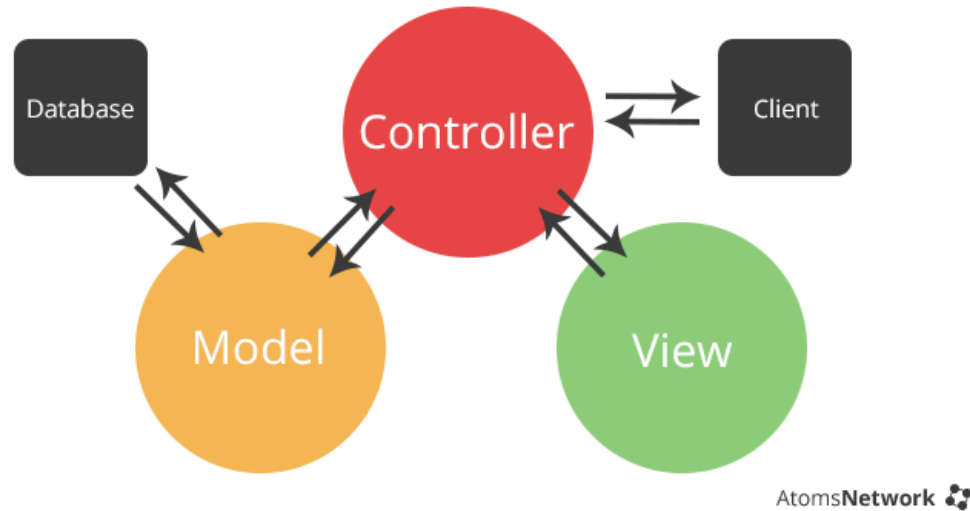
- Es la capa lógica y de datos de la aplicación.
- Representa la estructura de los datos (tablas en bases de datos, estructuras de objetos, etc.).
- Gestiona la lógica de negocio y la interacción con la base de datos.
- Responde a solicitudes de datos realizadas por el controlador.
- Notifica a la vista cuando los datos cambian.



Para un sistema de usuarios, el modelo puede incluir la definición de la entidad "Usuario" y métodos como crearUsuario, editarUsuario, o obtenerUsuarioPorID.

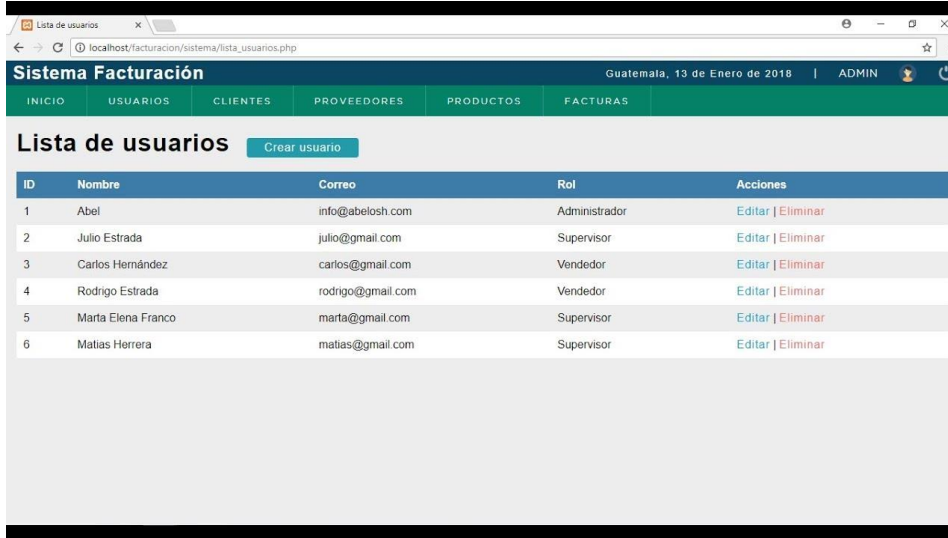
Ejemplo

Modelo



Vista

- Es la interfaz de usuario.
- Presenta los datos al usuario final (generalmente en forma visual).
- Depende del modelo, pero no lo modifica directamente.
- Se actualiza automáticamente cuando el modelo cambia, en implementaciones más avanzadas.



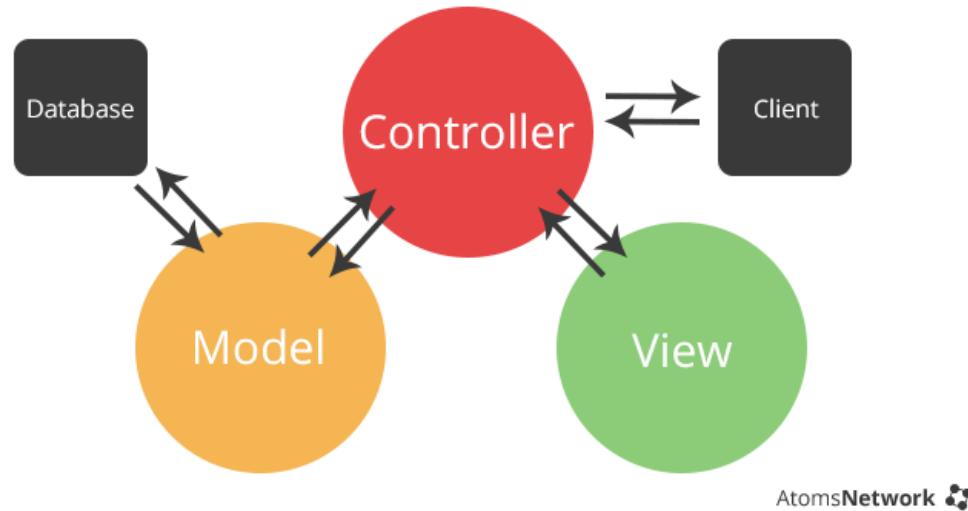
The screenshot shows a web browser window with the address bar displaying 'localhost/facturacion/sistema/lista_usuarios.php'. The page title is 'Lista de usuarios'. Below the title is a navigation bar with tabs: INICIO, USUARIOS, CLIENTES, PROVEEDORES, PRODUCTOS, and FACTURAS. The 'USUARIOS' tab is active. To the right of the tabs, it shows 'Guatemala, 13 de Enero de 2018' and 'ADMIN'. Below the navigation bar, there is a section titled 'Lista de usuarios' with a 'Crear usuario' button. Below this is a table with 5 columns: ID, Nombre, Correo, Rol, and Acciones. The table contains 6 rows of user data. Each row has 'Editar' and 'Eliminar' links in the 'Acciones' column.

ID	Nombre	Correo	Rol	Acciones
1	Abel	info@abelosh.com	Administrador	Editar Eliminar
2	Julio Estrada	julio@gmail.com	Supervisor	Editar Eliminar
3	Carlos Hernández	carlos@gmail.com	Vendedor	Editar Eliminar
4	Rodrigo Estrada	rodrigo@gmail.com	Vendedor	Editar Eliminar
5	Marta Elena Franco	marta@gmail.com	Supervisor	Editar Eliminar
6	Matias Herrera	matias@gmail.com	Supervisor	Editar Eliminar

Un formulario de registro o una lista de usuarios renderizada en HTML.

Ejemplo

Vista



Controlador

- Actúa como un intermediario entre el modelo y la vista.
- Recibe las entradas del usuario (a través de la vista) y las procesa.
- Llama a los métodos del modelo para realizar operaciones o actualizar los datos.
- Actualiza la vista en función de los resultados.
- Son las reglas para ejecución del programa.

Rellena los datos del formulario

Nombre	<input type="text" value="xve"/>
Apellidos	<input type="text" value="xve"/>
Correo	<input type="text" value="miMail@miDominio.com"/>

Confirma el envío del formulario

Estas seguro de enviar los valores introducidos en el formulario?

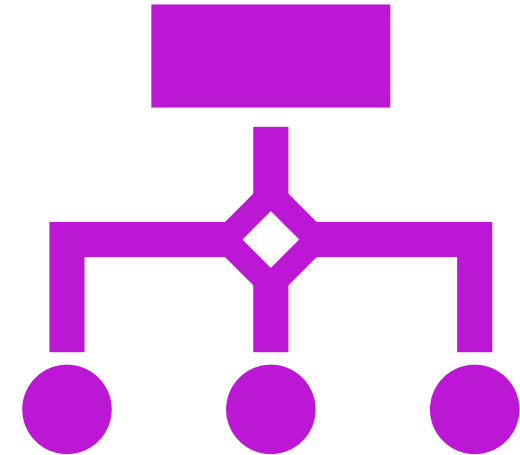
Si un usuario presiona el botón "Guardar", el controlador valida los datos, llama al método guardarUsuario del modelo, y decide si mostrar un mensaje de éxito o error.

Ejemplo

Controlador

Flujo de Trabajo

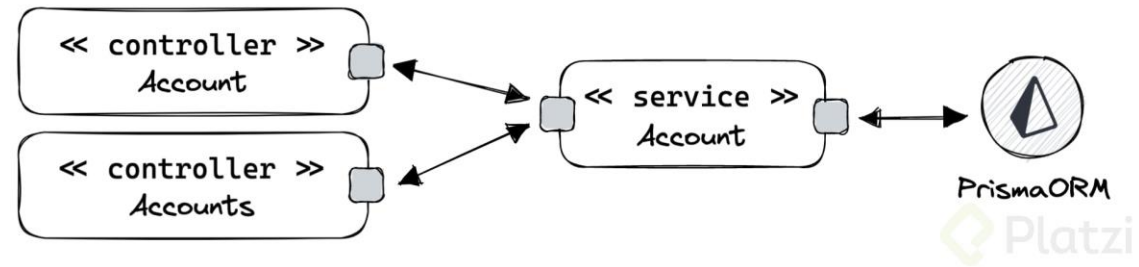
- El usuario interactúa con la vista (por ejemplo, llena un formulario o hace clic en un botón).
- La vista envía la solicitud al controlador.
- El controlador procesa la solicitud, valida los datos, y realiza las operaciones necesarias en el modelo.
- El modelo actualiza los datos y notifica al controlador si hubo algún cambio.
- El controlador envía las actualizaciones a la vista, que presenta la información actualizada al usuario.



Ventajas (MVC)



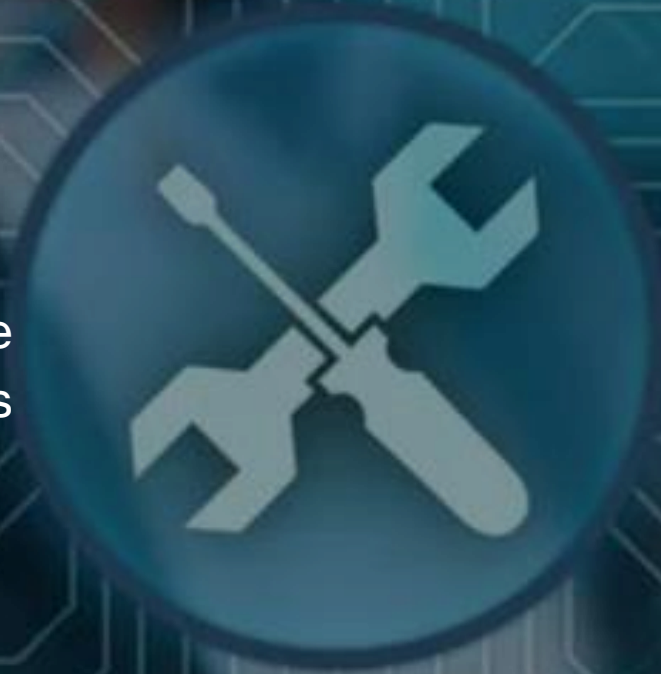
Separación de responsabilidades



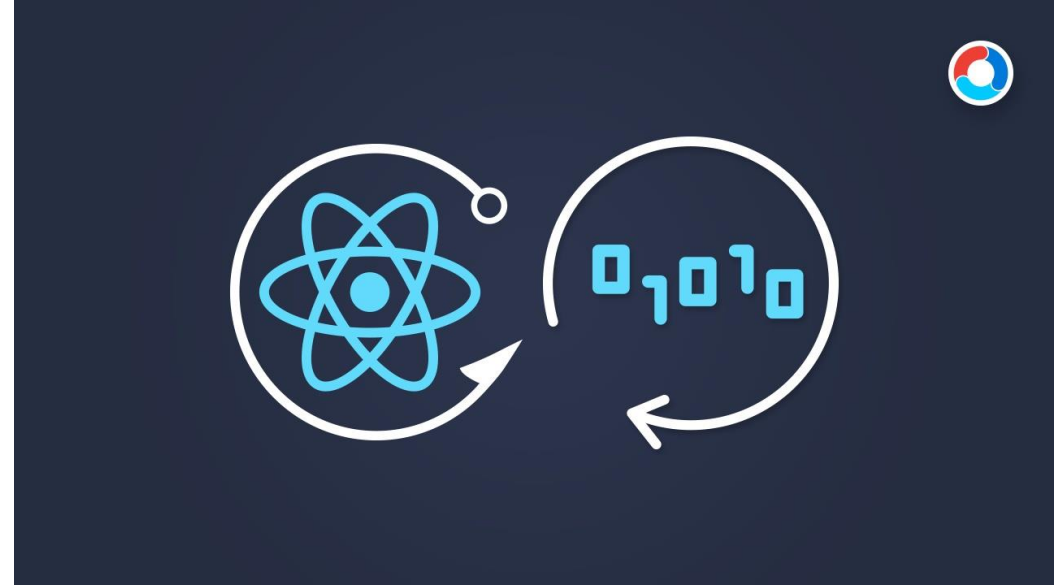
- El modelo gestiona los datos.
- La vista gestiona la presentación.
- El controlador gestiona la lógica de aplicación.

Mantenibilidad

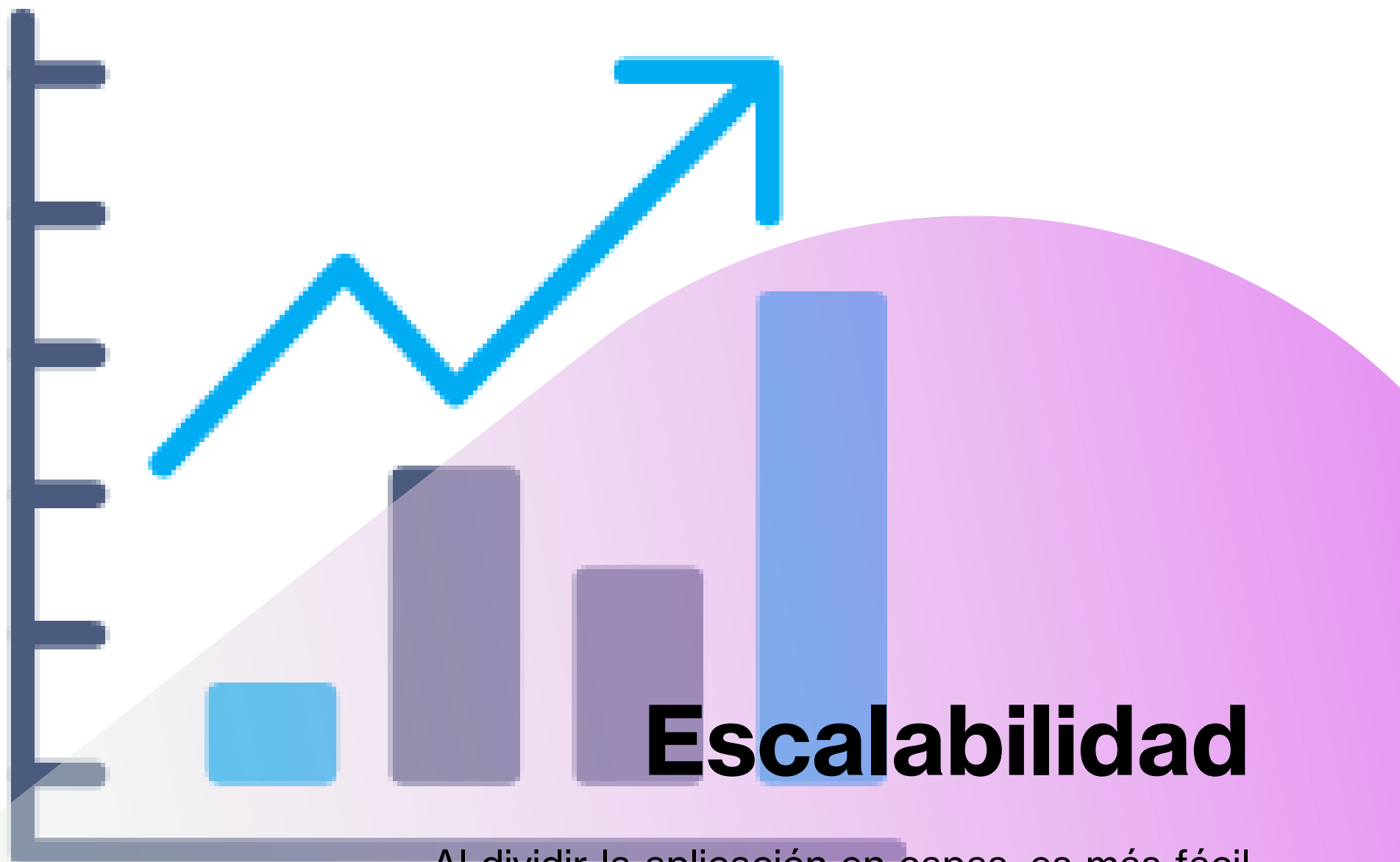
El código está mejor organizado, lo que facilita corregir errores o implementar nuevas funcionalidades.



Reutilización del código



- Las vistas pueden reutilizar componentes.
- Los modelos pueden reutilizarse en diferentes controladores o vistas.

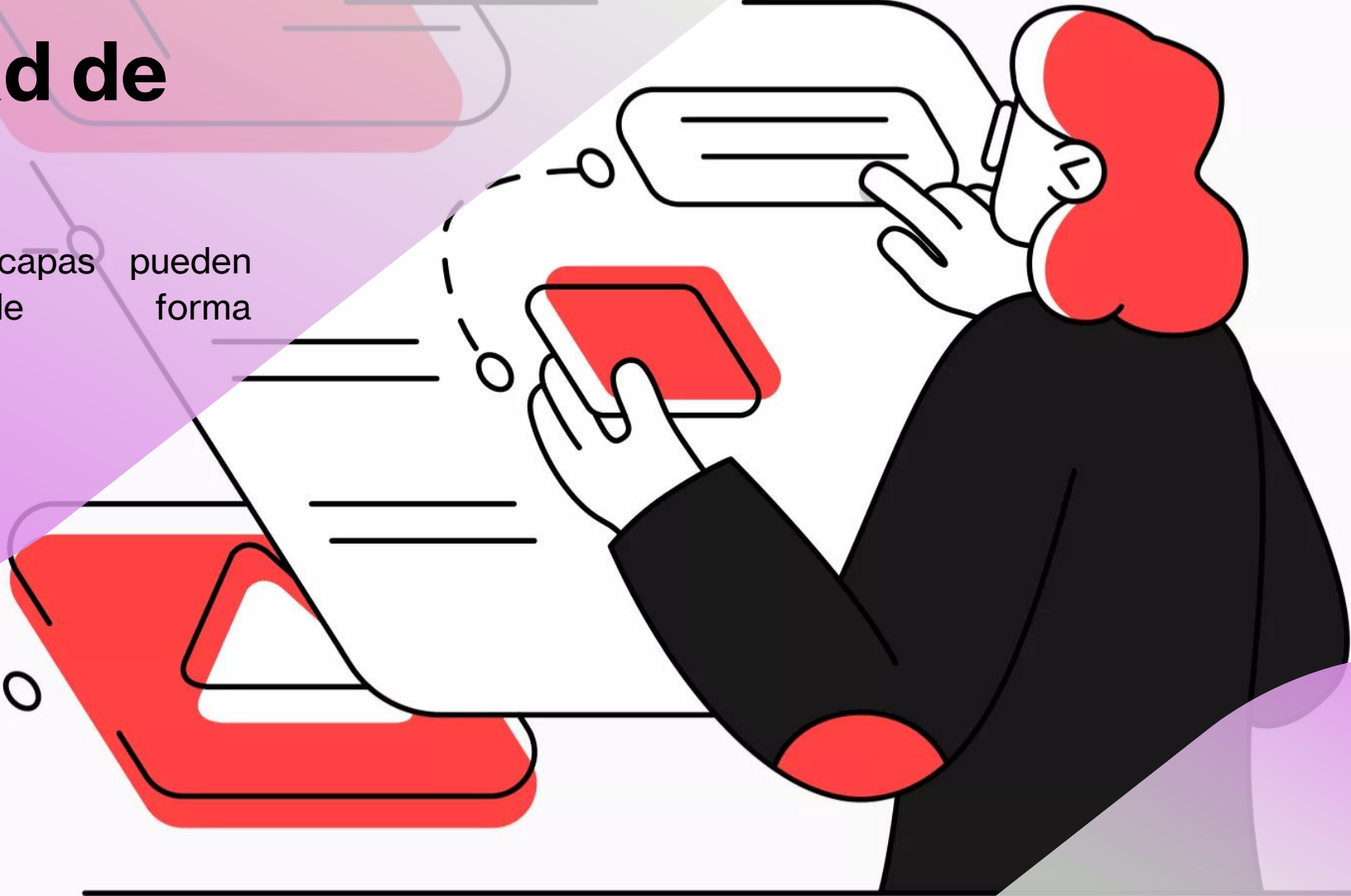


Escalabilidad

Al dividir la aplicación en capas, es más fácil añadir nuevas características o ampliarla.

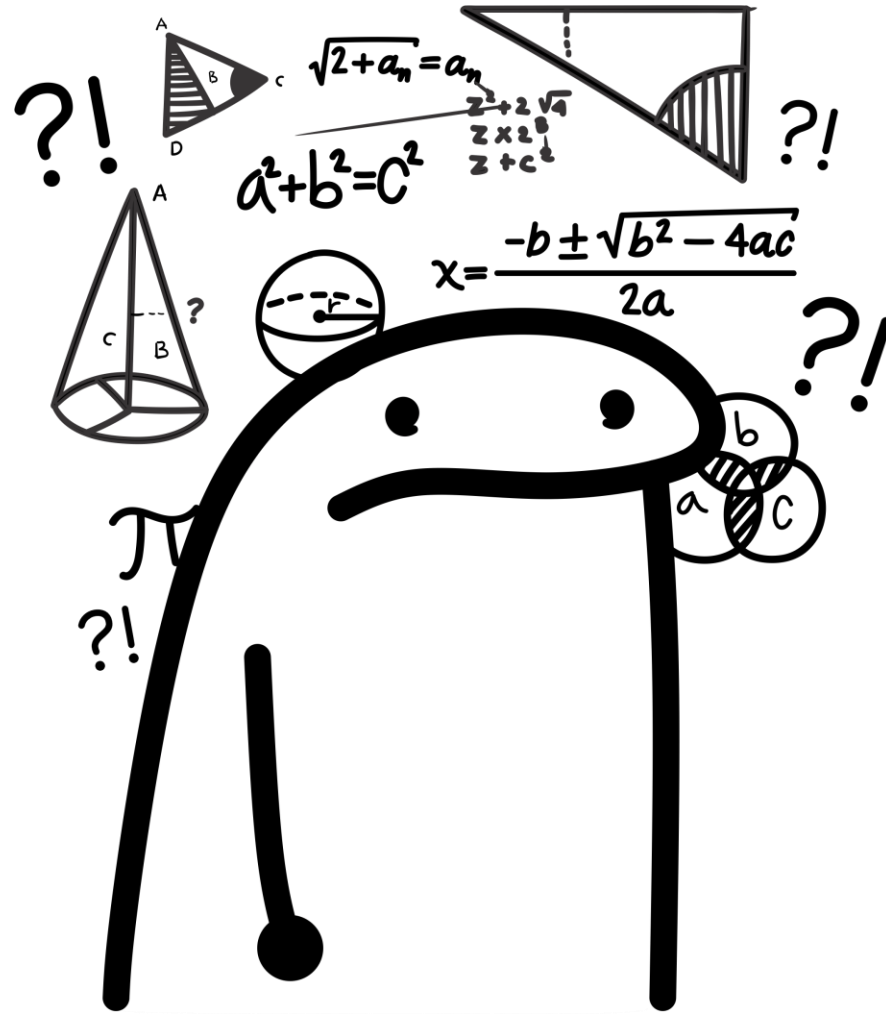
Facilidad de prueba

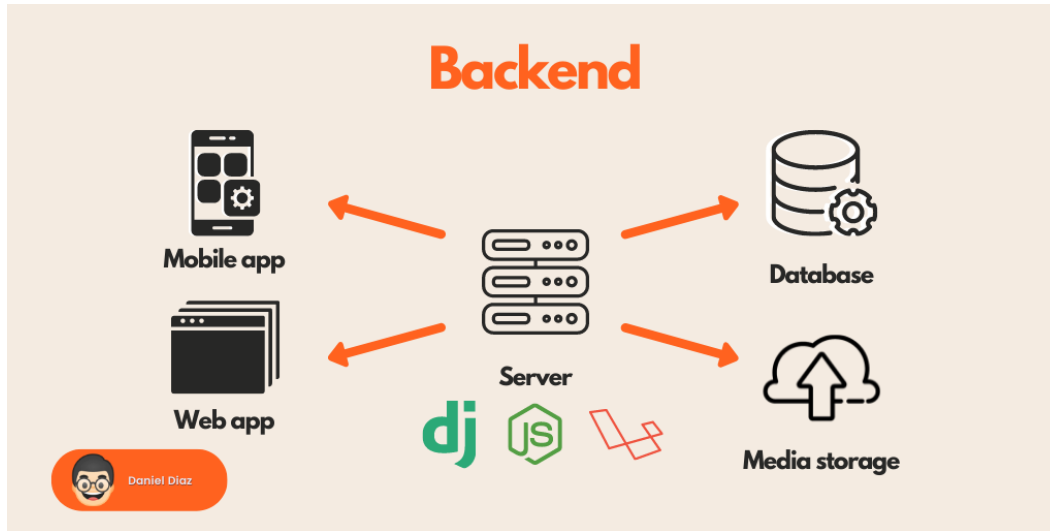
Las diferentes capas pueden probarse de forma independiente.



¿Dónde se aplica el patrón MVC?

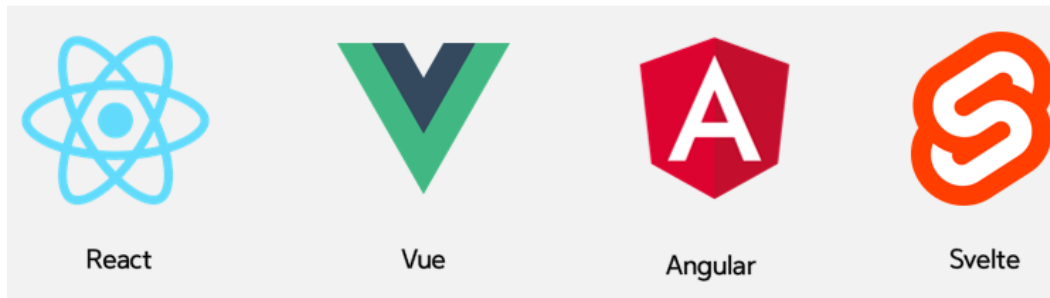
El patrón MVC se usa ampliamente en el desarrollo de aplicaciones web, móviles y de escritorio.





- Laravel (PHP)
- Django (Python)
- Ruby on Rails (Ruby)
- ASP.NET (.NET)
- Spring Boot (Java)

Backend



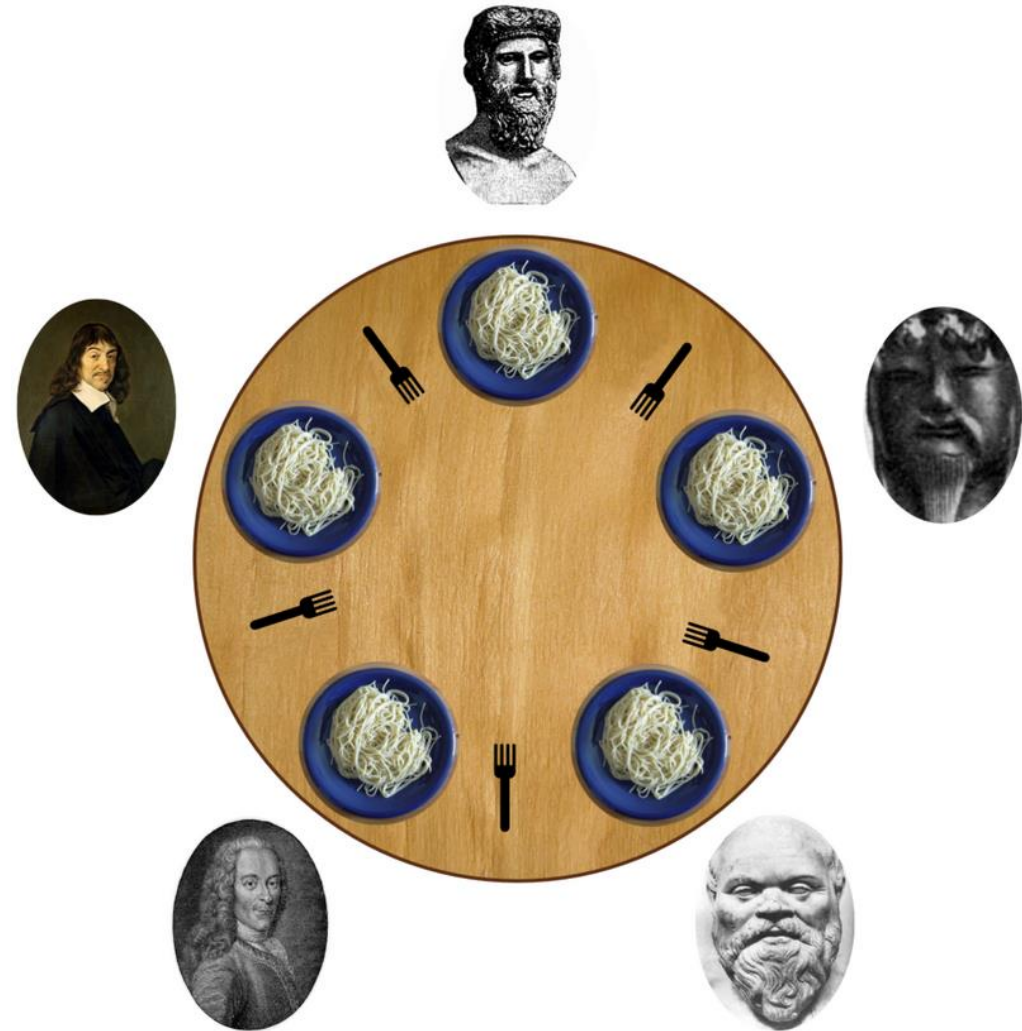
Frontend

Aunque el frontend no implementaba MVC de manera directa, hay frameworks con conceptos inspirados en el mismo.

- Angular
- Vue
- Svelte

Concurrencia

Se refiere a la habilidad de distintas partes de un programa, algoritmo, o problema de ser ejecutado en desorden o en orden parcial, sin afectar el resultado final. Los cálculos (operaciones) pueden ser ejecutados en múltiples procesadores, o ejecutados en procesadores separados físicamente o virtualmente en distintos hilos de ejecución.



Fundamentos de Sistemas Operativos

INTERBLOQUEO (5/5)



ITESO
Universidad Jesuita
de Guadalupe

Video

[https://www.youtube.com/
watch?v=yFoq8mTL9RE](https://www.youtube.com/watch?v=yFoq8mTL9RE)

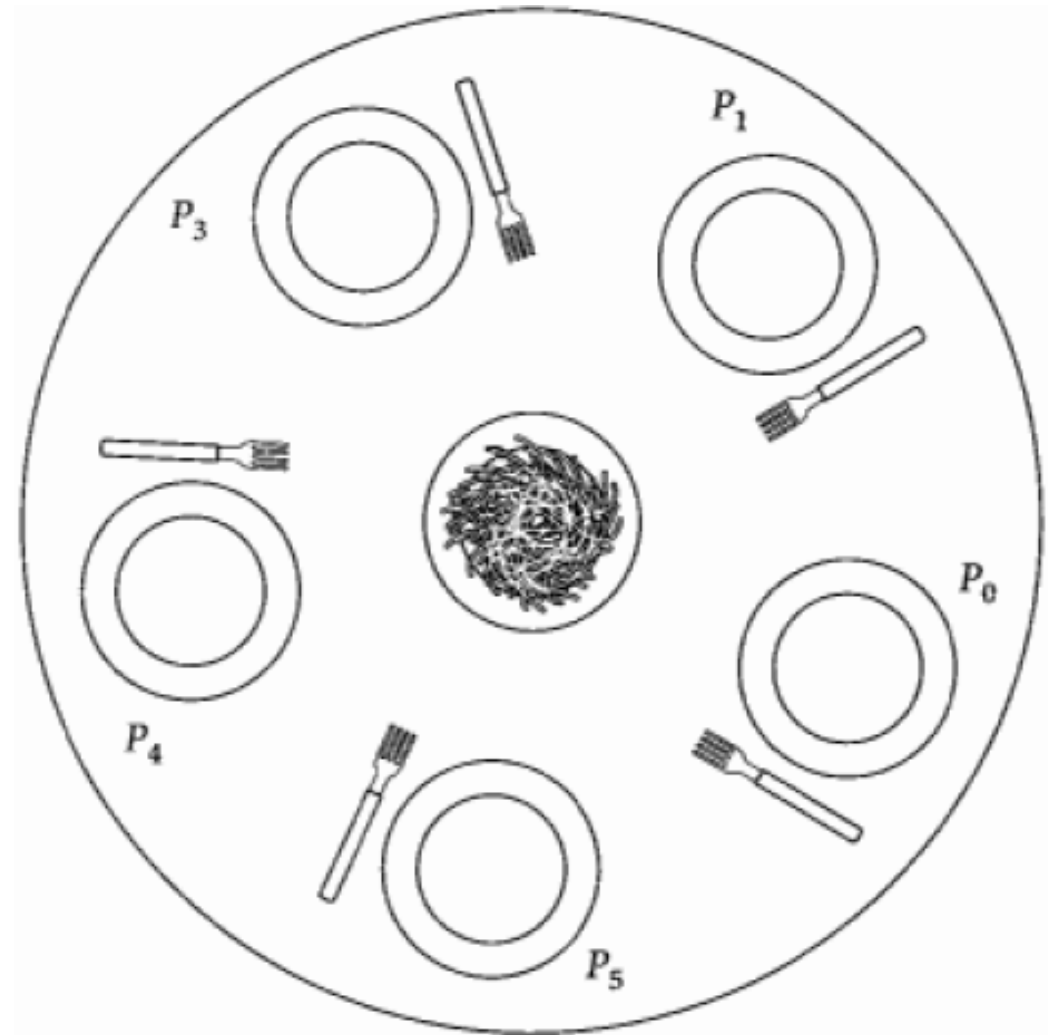
Para entenderlo mejor

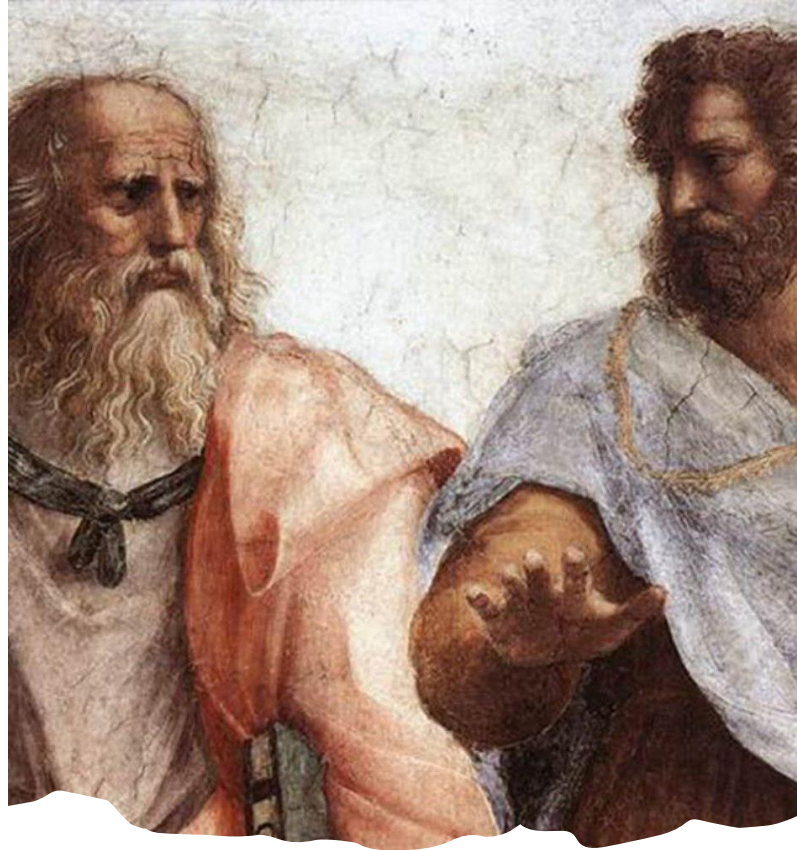
Imagina una cena entre
5 filósofos



Empieza el problema

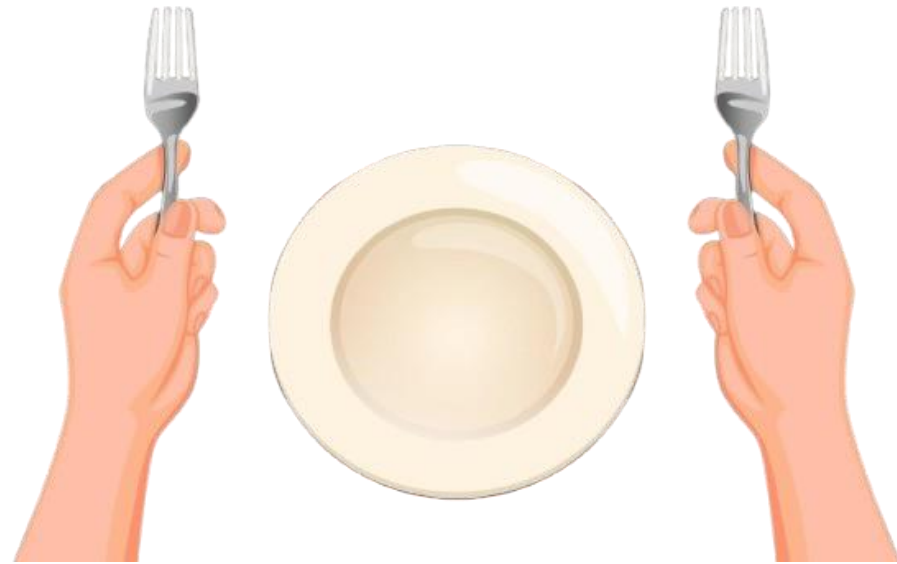
Los filósofos están sentados en una mesa redonda y cada uno tiene un plato de espaguetis frente a sí y entre cada plato hay un tenedor.





Un filósofo puede estar pensando o comiendo

Si un filósofo va a comer, primero intenta agarrar el tenedor a su izquierda y luego el tenedor a su derecha.



Al terminar de comer, el filósofo vuelve a pensar, y el ciclo se repite.





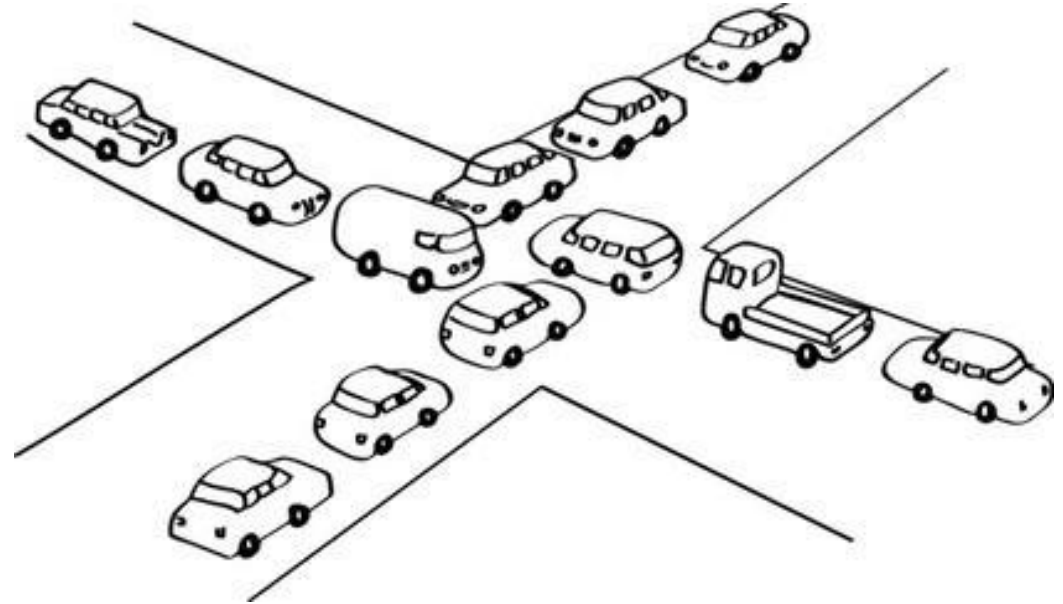
Aquí reside el problema

El problema es que, entonces los filósofos no podrían comer al mismo tiempo, pues están constantemente compitiendo por tenedores.

La concurrencia es...

La capacidad de ejecutar varios procesos de manera simultánea, de modo que estos puedan compartir recursos.

Así, la cena de los filósofos es un problema de concurrencia, es decir, un deadlock.

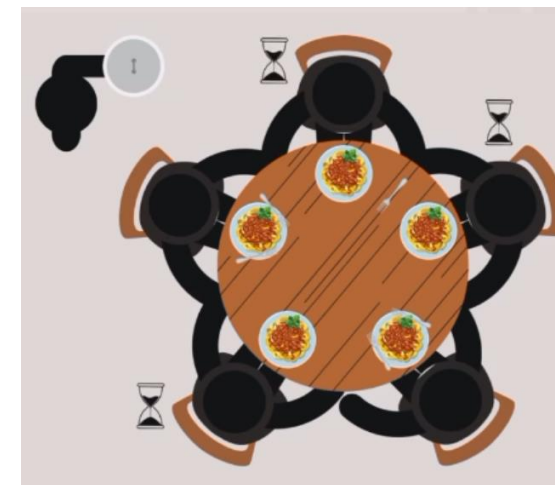
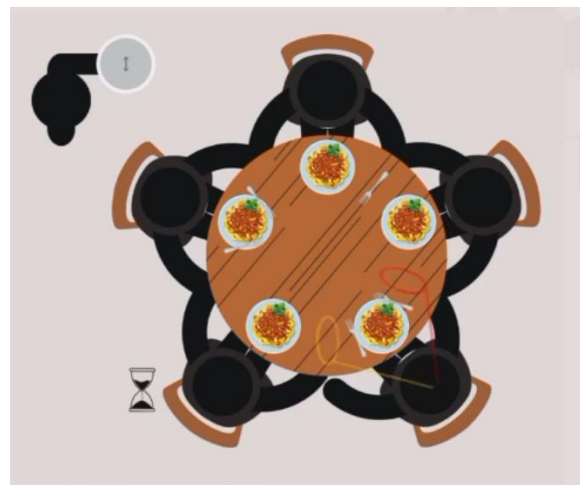
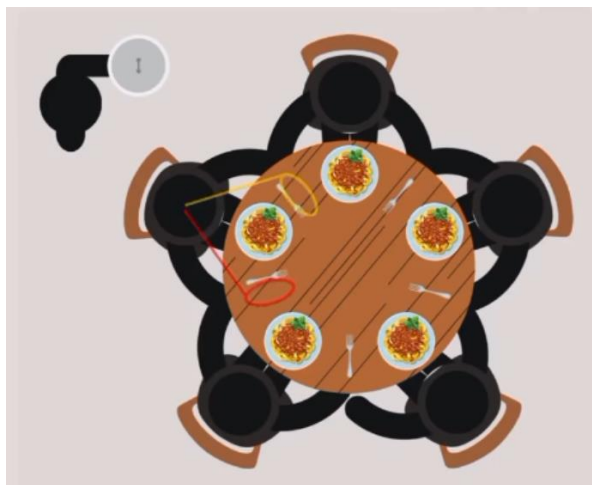


Una posible solución

Una de las soluciones más aceptadas es la de Edsger Dijkstra, quien propuso que solo $N-1$ filósofos puedan sentarse en la mesa, donde N es el número de tenedores.

De este modo, quedará un tenedor disponible y al menos uno de los filósofos podrá comer.





Que exista un “camarero”, y que los filósofos tengan que pedirle permiso para comer.



El camarero solo les permitirá hacerlo si hay tenedores disponibles, y si no, podrá su solicitud en cola.

Ejemplo gracias a...

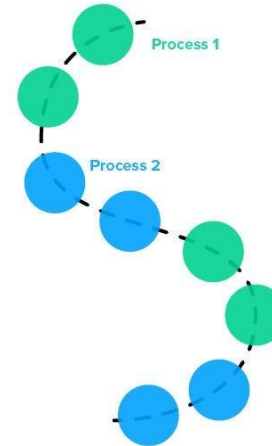
https://www.instagram.com/p/C9_D-00gxfG/?img_index=1



Paralelismo

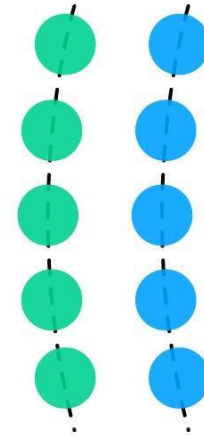
Es una forma de computación en la cual varios cálculos pueden realizarse simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo.

Concurrency



vs

Parallelism



Tarea 1

Tarea 2

Tarea 3

Secuencial

Hilo 1



Concurrente
1 Procesador

Hilo 1



Hilo 2



Hilo 3



Concurrente
3 o más procesadores

Hilo 1



Hilo 2



Hilo 3



Concurrencia Vs Paralelismo

A black and white photograph of a pair of glasses, with the frame and temples visible. The background is a blurred image of a piece of paper covered in handwritten mathematical equations and formulas. The text "Una forma para entenderlo de mejor forma" is overlaid on the left side of the image.

**Una forma para
entenderlo de mejor
forma**



Se te ha retado a que debes comer un pastel enorme y cantar una canción completa



Tarea 1

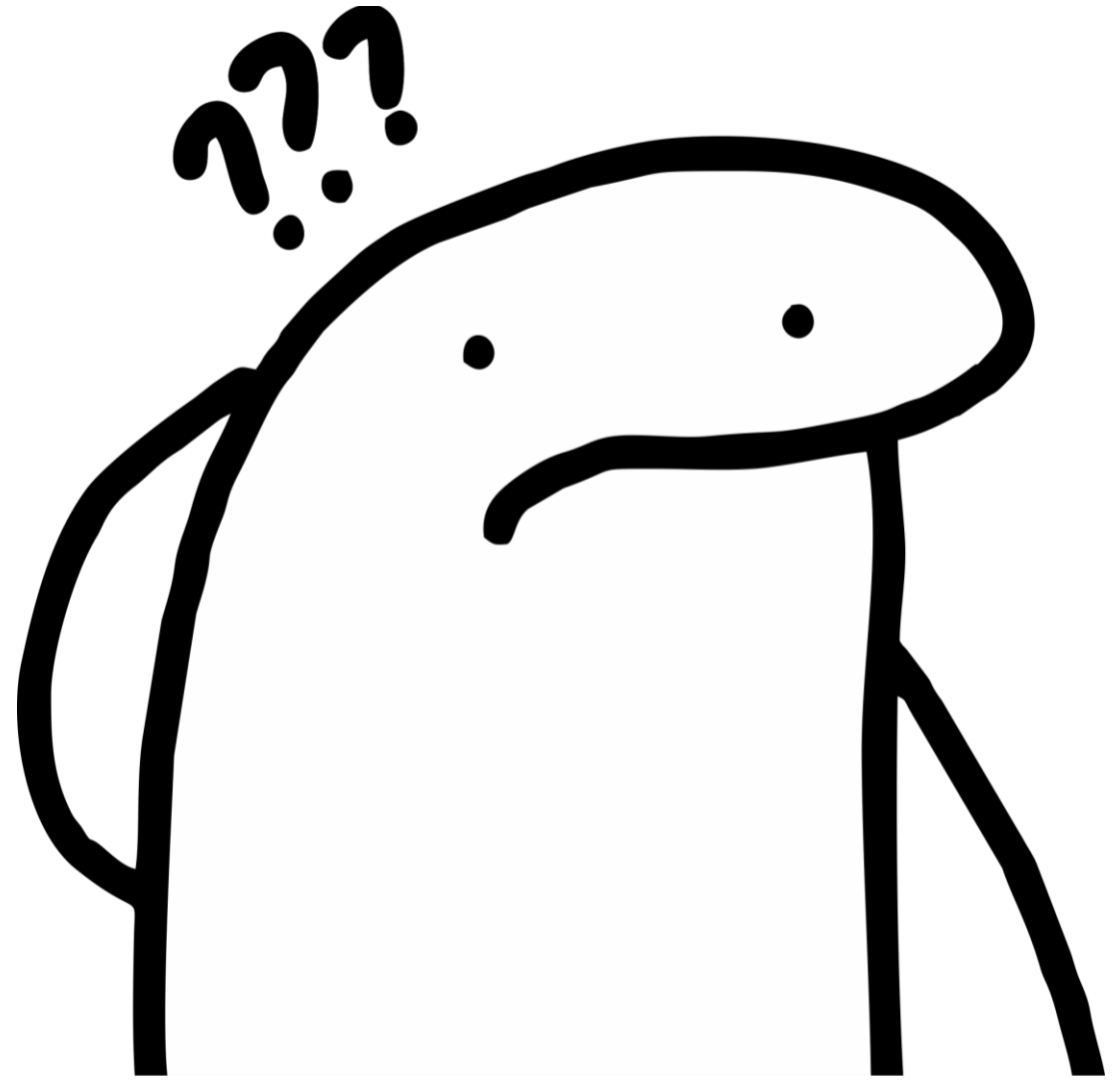
Comer Pastel



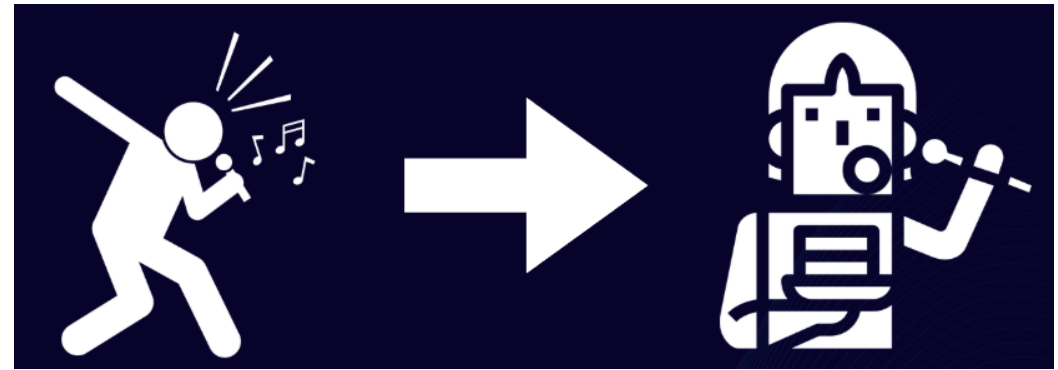
Tarea 2

Cantar

Pero no te han dicho que debas
hacerlo al mismo tiempo.

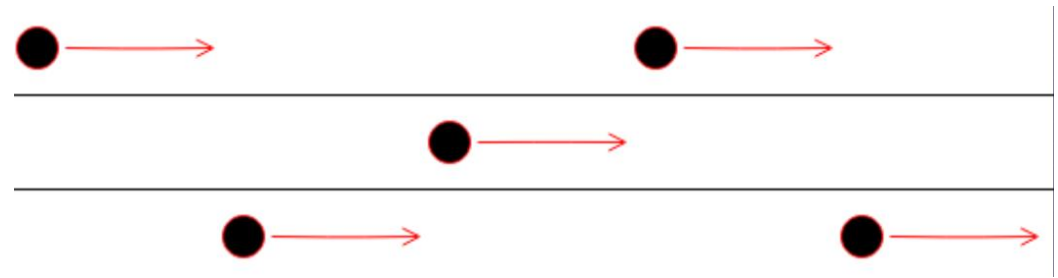


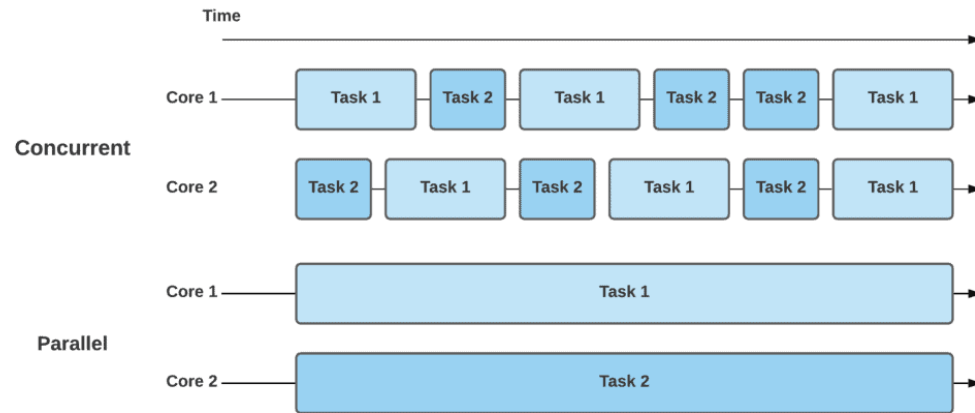
Es decir, que perfectamente podrías cantar la canción y luego comerte el pastel o viceversa.



Concurrencia

La concurrencia se refiere a la capacidad de un sistema para manejar múltiples tareas al mismo tiempo de manera intercalada. Aunque las tareas no necesariamente se ejecutan simultáneamente, el sistema las organiza de tal forma que avancen de manera aparente simultánea. La concurrencia es útil para gestionar múltiples operaciones en un mismo núcleo de procesamiento.



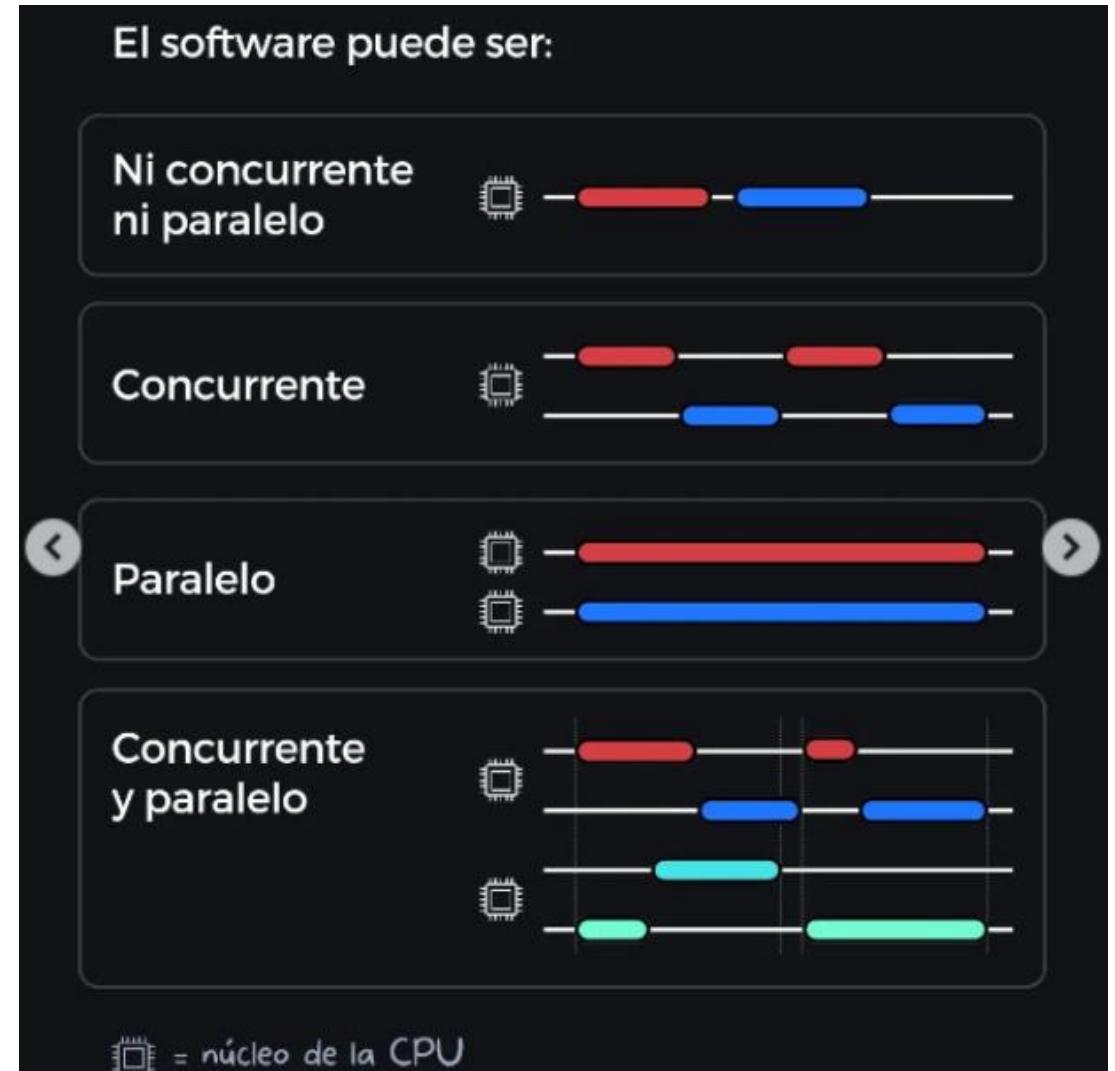


El paralelismo implica la ejecución simultánea de múltiples tareas, utilizando varios núcleos o procesadores. En este caso, las tareas realmente se ejecutan al mismo tiempo en hardware diferente. El paralelismo es ideal para acelerar el procesamiento de grandes volúmenes de datos.

Paralelismo

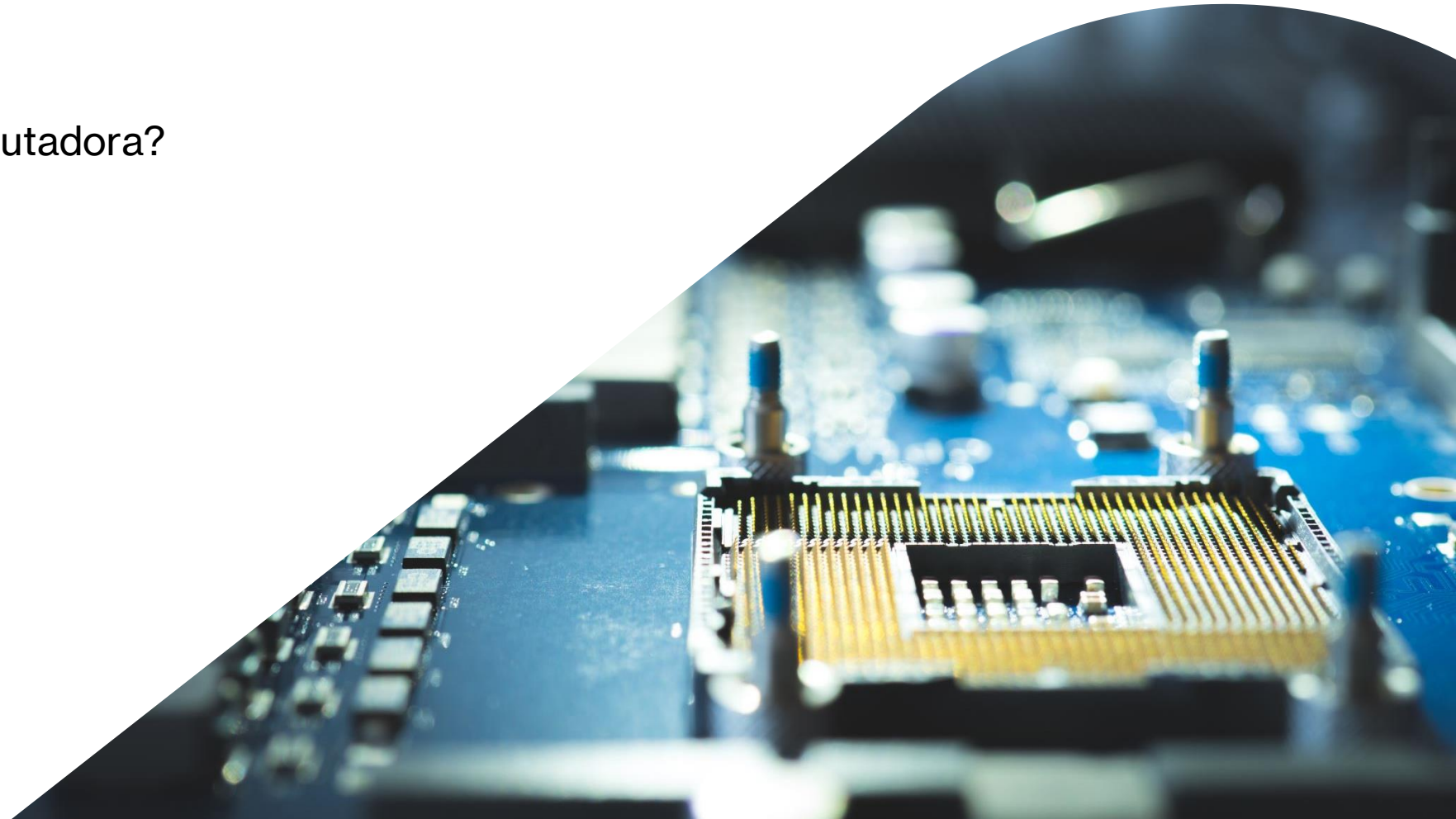
Ejemplo

https://www.instagram.com/p/C-6JbX3Ojfm/?img_index=1



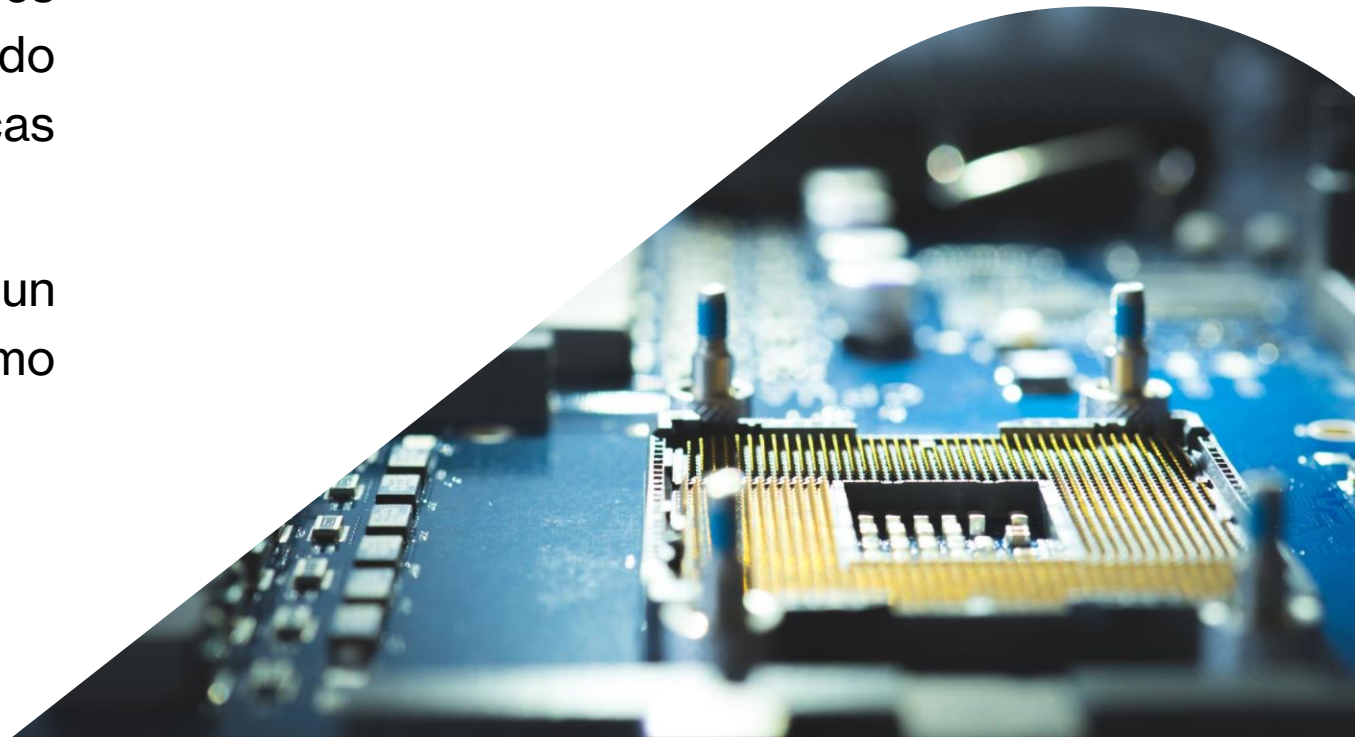
Sistemas Operativos y Procesadores

¿Cómo funciona la computadora?



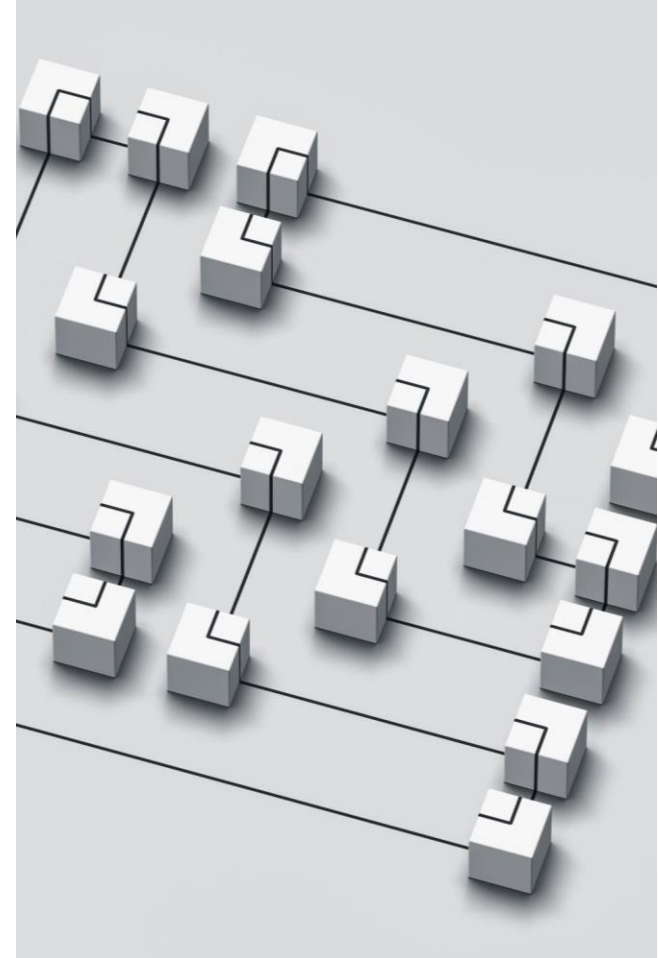
Procesador

- Es el hardware especializado de la computadora que interpreta instrucciones de un programa informático, utilizando únicamente operaciones básicas aritméticas, lógicas y de entrada o salida.
- Una computadora puede tener más de un procesador. Esto se conoce como multiprocesamiento.



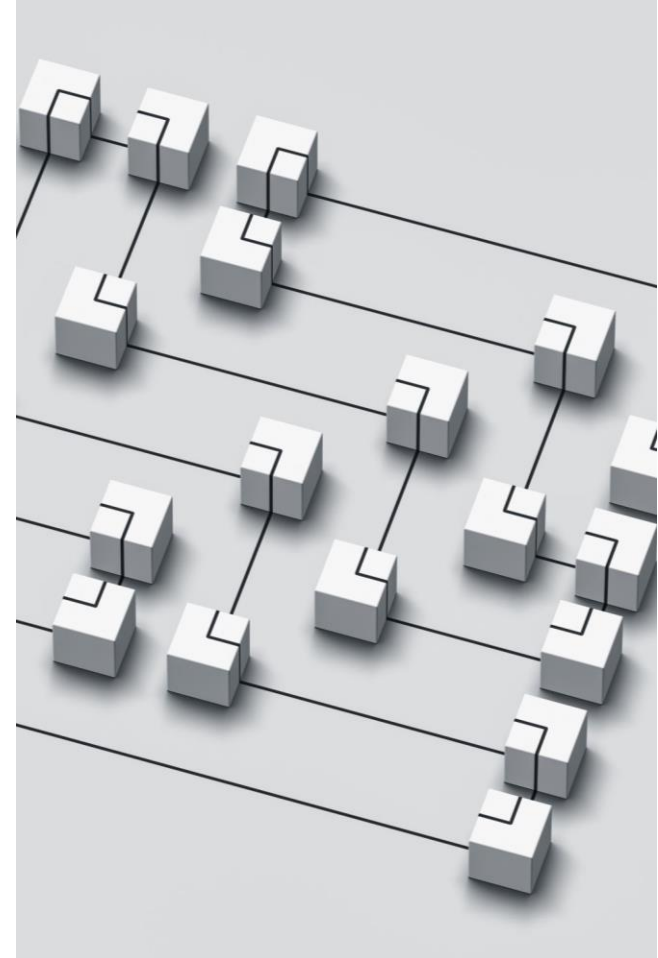
Multiprocesamiento

La mayoría de los CPU, y de hecho, la mayoría de los dispositivos de lógica secuencial, son de naturaleza síncrona. Están diseñados y operan en función de una señal de sincronización. Esta señal, conocida como señal de reloj, usualmente toma la forma de una onda cuadrada periódica. Calculando el tiempo máximo en que las señales eléctricas pueden moverse en las varias bifurcaciones de los muchos circuitos de un CPU, los diseñadores pueden seleccionar un período apropiado para la señal del reloj.



Multiprocesamiento

Cuando los procesadores y su interconexión son implementados en un único chip de silicio, la tecnología se conoce como un procesador multinúcleo. El "performance" o la velocidad de un procesador depende de, entre muchos otros factores, la velocidad del reloj (generalmente dada en múltiplos de hertz) y las instrucciones por ciclo de reloj (IPC), que juntos son los factores para las instrucciones por segundo (IPS) que el CPU puede rendir.

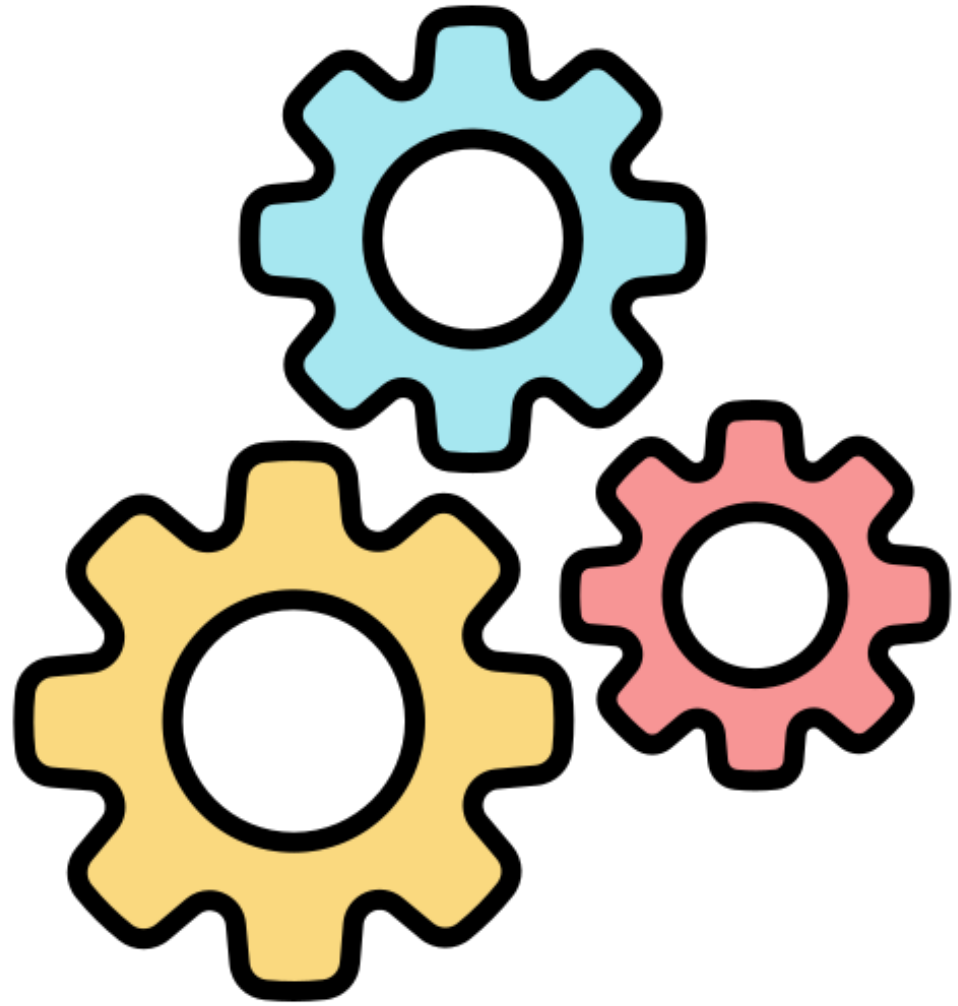


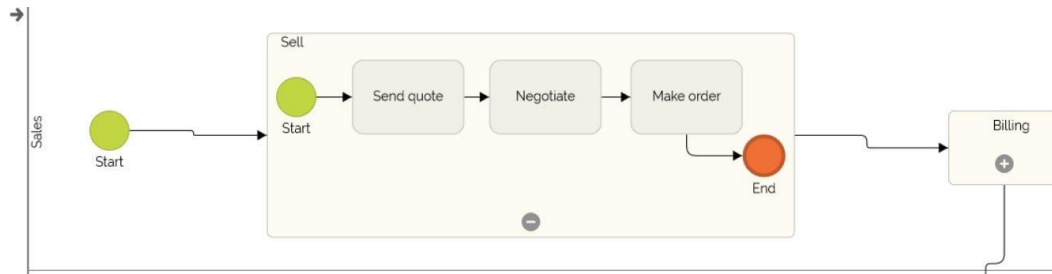
Sistema Operativo

Un sistema operativo es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes. En ciertos textos, el sistema operativo es llamado indistintamente como núcleo o kernel, pero debe tenerse en cuenta que la diferencia entre kernel y sistema operativo solo es aplicable si el núcleo es monolítico, lo cual fue muy común entre los primeros sistemas. En caso contrario, es incorrecto llamar al sistema operativo núcleo.

Proceso

Es una instancia en ejecución de un programa de computadora. Puede incluir el código ejecutable, datos, variables de entorno, asignaciones de memoria, recursos del sistema operativo y otros atributos que describen el estado de ejecución del programa.





Sub-Proceso

Es una secuencia de instrucciones que puede ser ejecutada por un procesador de manera independiente de otros subprocesos. Los subprocesos comparten recursos con otros subprocesos del mismo proceso, lo que permite la ejecución de múltiples tareas de manera concurrente.

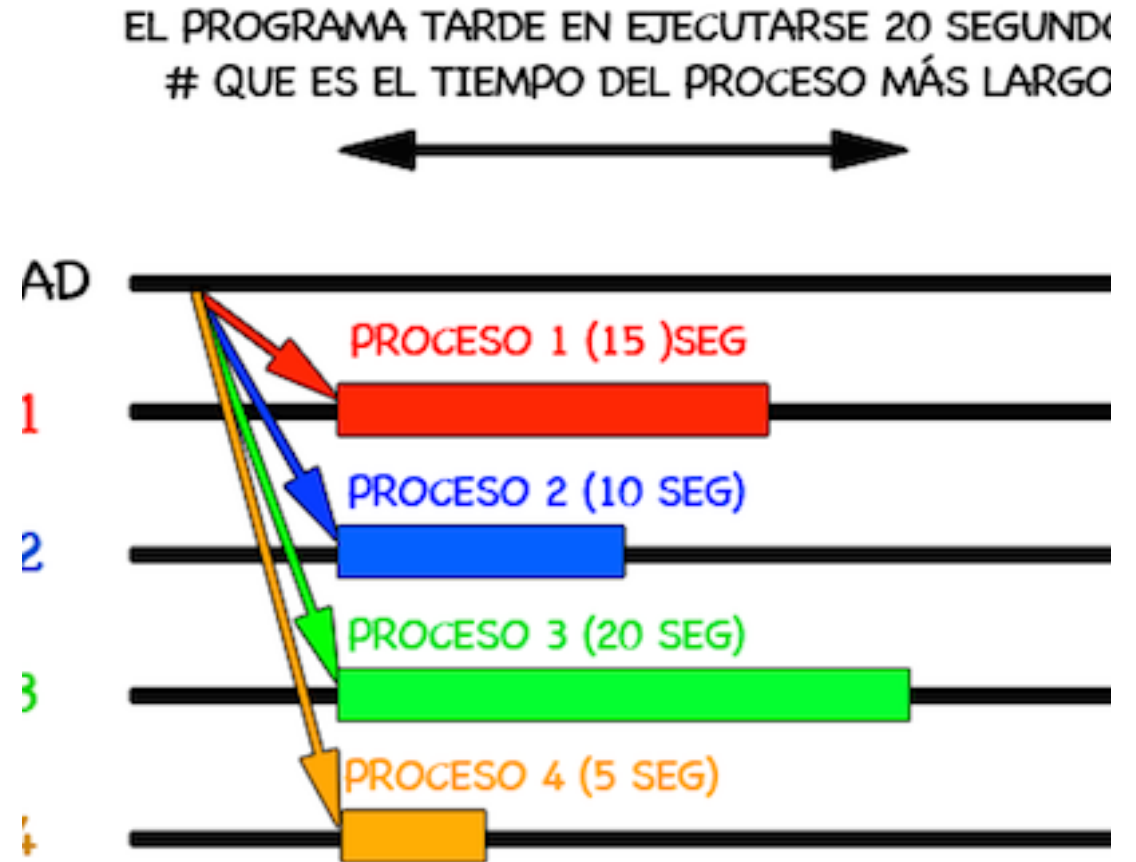
Hilos

¿Qué son los hilos? Estados de hilos



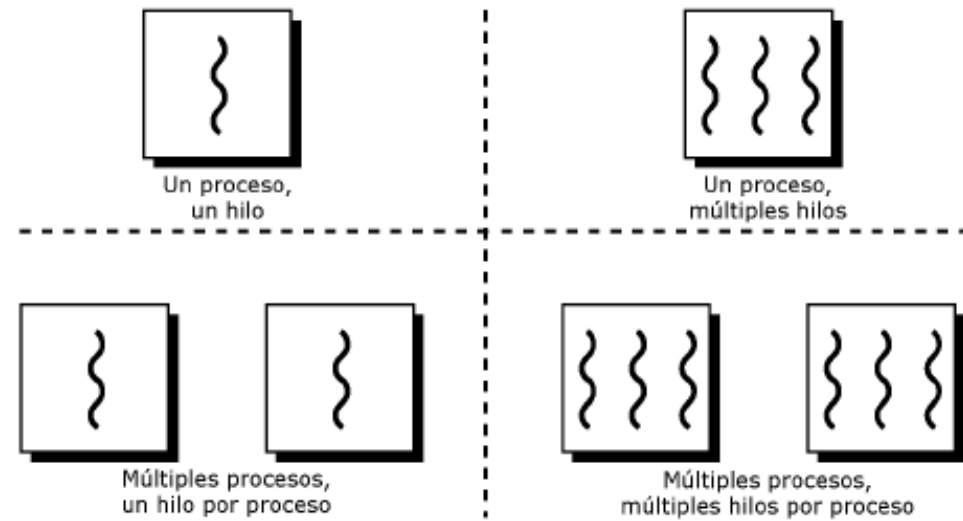
Hilo

En sistemas operativos, un hilo (del inglés thread), proceso ligero o subproceso, es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo. La destrucción de los hilos antiguos por los nuevos es una característica que no permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, la situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.



Hilo

- Un hilo es simplemente una tarea que puede ser ejecutada al mismo tiempo que otra tarea.
- Algunos lenguajes de programación tienen características de diseño expresamente creadas para permitir a los programadores lidiar con hilos de ejecución (como Java o Delphi). Otros (la mayoría) desconocen la existencia de hilos de ejecución y estos deben ser creados mediante llamadas de biblioteca especiales que dependen del sistema operativo en el que estos lenguajes están siendo utilizados (como es el caso del C y del C++).



Estados de Hilos



Creación: Cuando se crea un proceso se crea un hilo para ese proceso.



Bloqueo: Cuando un hilo necesita esperar por un suceso, se bloquea (salvando sus registros de usuario, contador de programa y punteros de pila).



Desbloqueo: Cuando el suceso por el que el hilo se bloqueó se produce, el mismo pasa a la final de los Listos.



Terminación: Cuando un hilo finaliza se liberan tanto su contexto como sus columnas.

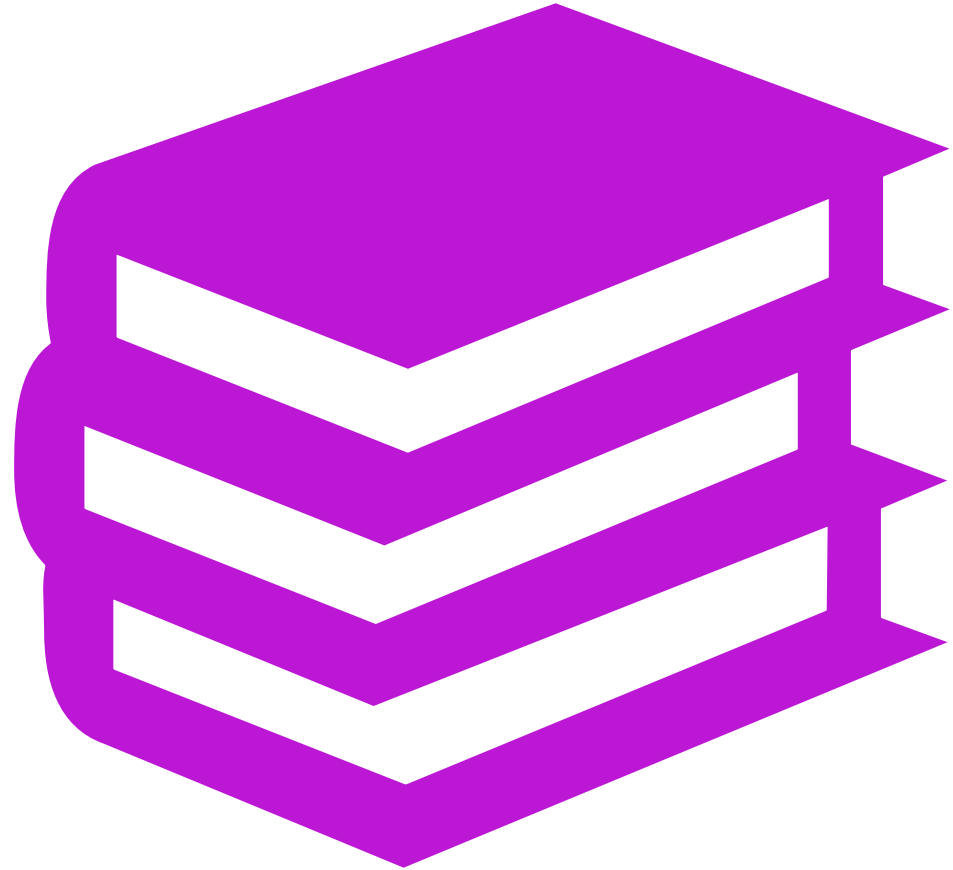
Creación de hilos

¿Cómo usar hilos en JAVA?



Hilos en JAVA

Una de las características de Java es que es multi-hilo. Esto significa que provee librerías e interfaces nativas para implementar el threading en sus aplicaciones.





Opción 1: Clase Thread

Thread es una clase en Java que implementa la interfaz “Runnable”. Esta clase permite ejecutar como un hilo, tanto un proceso definido en su método “run” cuando se hereda, y al llamar a su método “start”.

Pasos para la creación de un hilo con Thread

Crear una clase

Heredar la clase “Thread”.

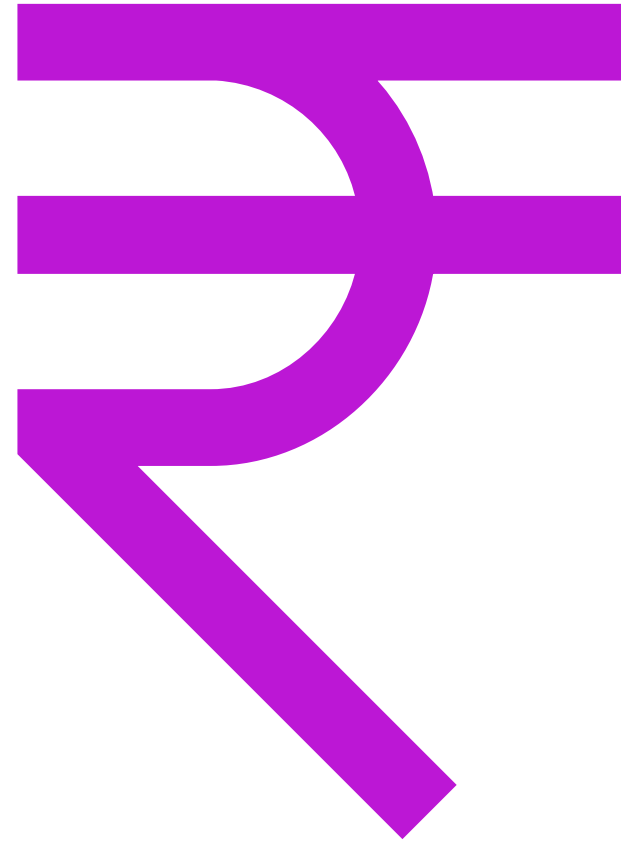
Sobreescribir el método “run”

Instanciar la clase

Llamar al método “start”

Opción 2: Interfaz Runnable

Esta interfaz contiene el método “run”, el cual es ejecutado como hilo dentro por el método “start” de un objeto de la clase “Thread”.





Pasos para la creación de un hilo con Runnable

Crear una clase

Implementar la interfaz “Runnable”

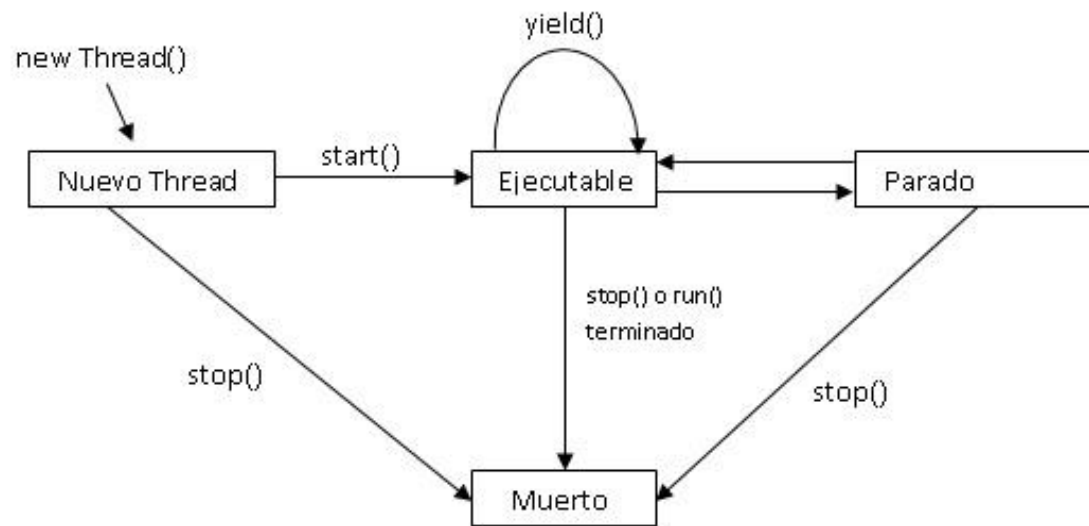
Sobreescribir el método “run”

Instanciar la clase

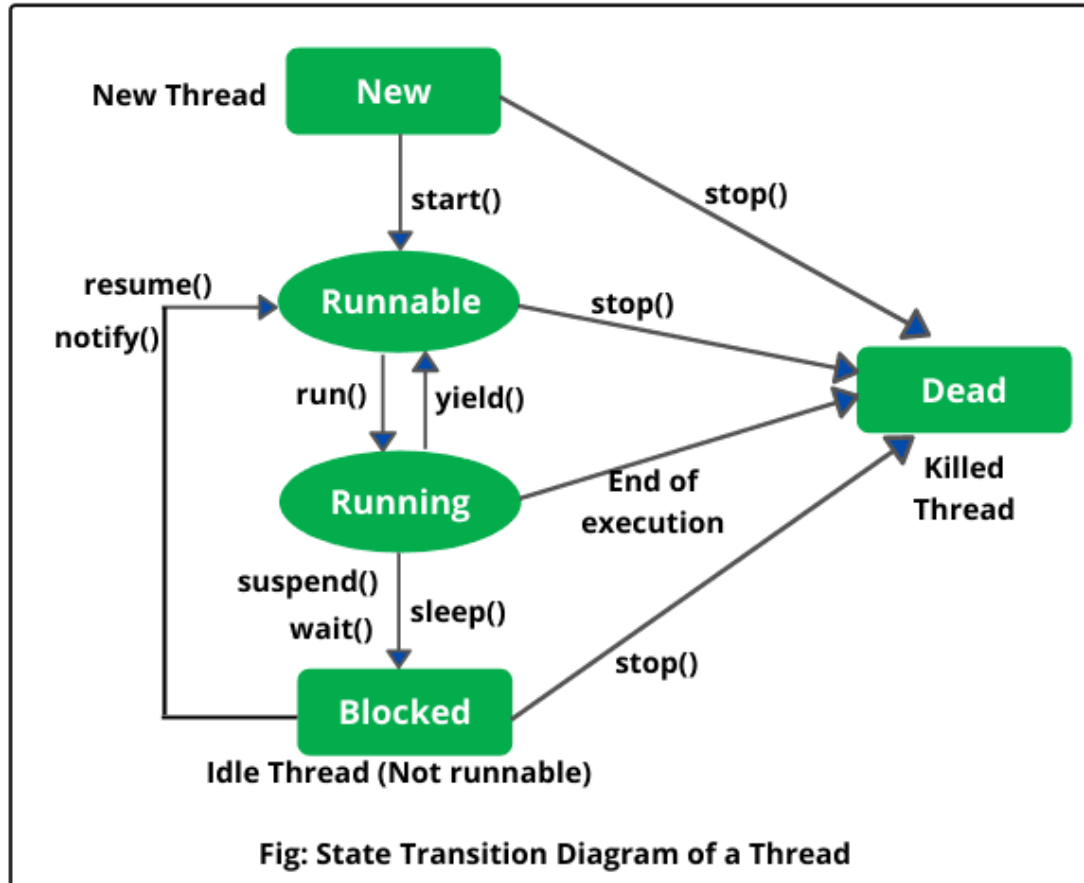
Crear un nuevo objeto “Thread”, y pasar como parámetro al constructor, la instancia de la clase que implementa “Runnable”

Llamar al método “start” de la instancia de “Thread”

Ciclo de Vida



- Siempre se inicializa el hilo.
- El hilo inicia cuando se le indica.
- El hilo se le considera muerto cuando terminó su ejecución o la detuvimos manualmente.
- El estado de parado, son pausas consecutivas en el hilo.



Métodos Comunes

- **start():** Indica iniciar una tarea. Se invoca con el método `run()`.
- **run():** Cuerpo de una tarea o hilo de ejecución. Es un llamado por el método `start()`.
- **sleep(long):** Provoca que la tarea en curso entre en pausa durante un número de milisegundos indicado por "long".
- **stop():** Detiene la ejecución de una tarea y la destruye.
- **suspend():** Detiene la ejecución sin destruir la tarea del sistema ni el estado de la tarea.
- **resume():** Continúa la ejecución desde donde se había suspendido.

Ejemplos



Dudas o Comentarios

