

Modelo cliente servidor

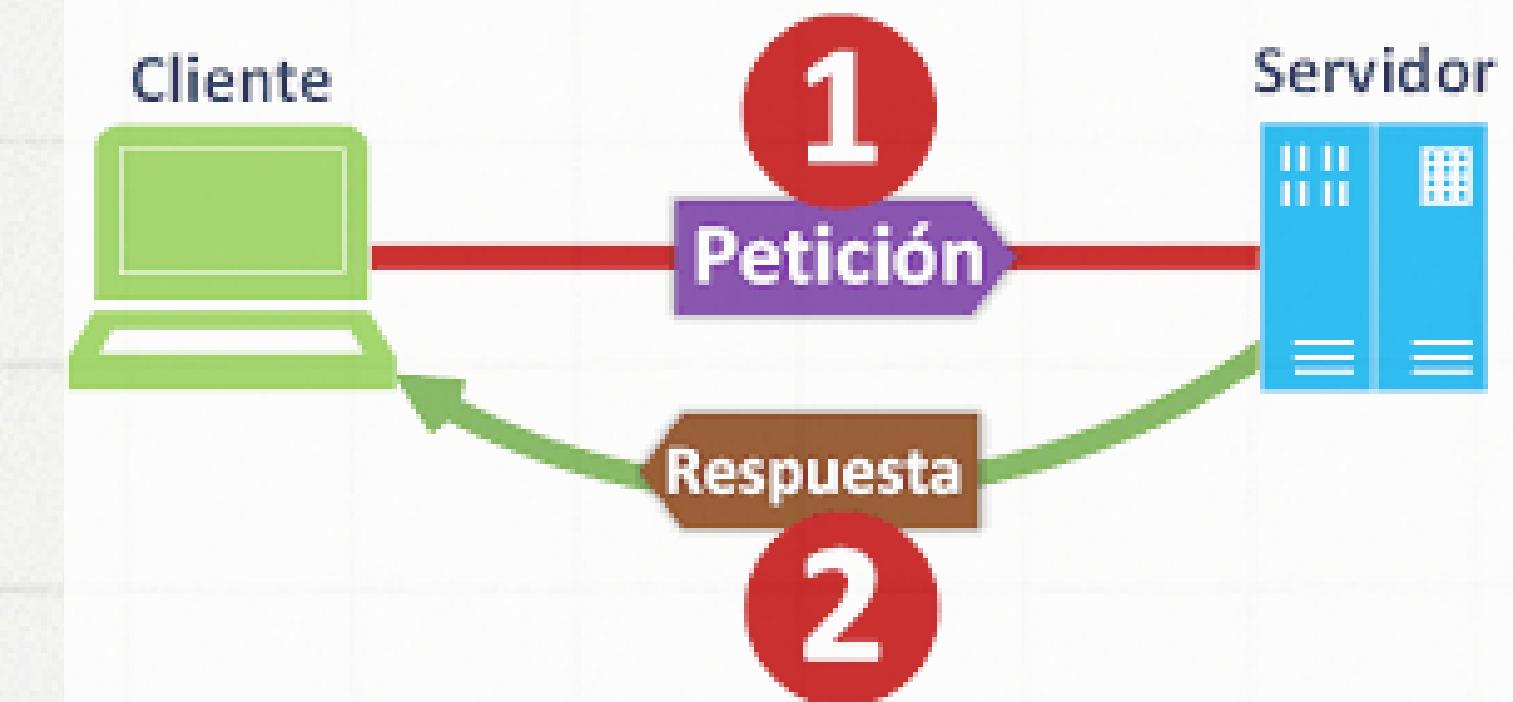
Proyecto “Explorando el Universo”

Raquel Rehbein Brange

¿Qué es el modelo cliente servidor?

La arquitectura cliente-servidor permite la comunicación bidireccional entre 2 aplicaciones distintas, en resumen decimos que: el cliente envía una petición (request) al servidor y este envía una respuesta (response) al cliente.

Actualmente es uno de los estilos arquitectónicos más conocidos y es utilizado en aplicaciones web, aplicaciones de correo electrónico, redes sociales, aplicaciones de streaming, entre otros.

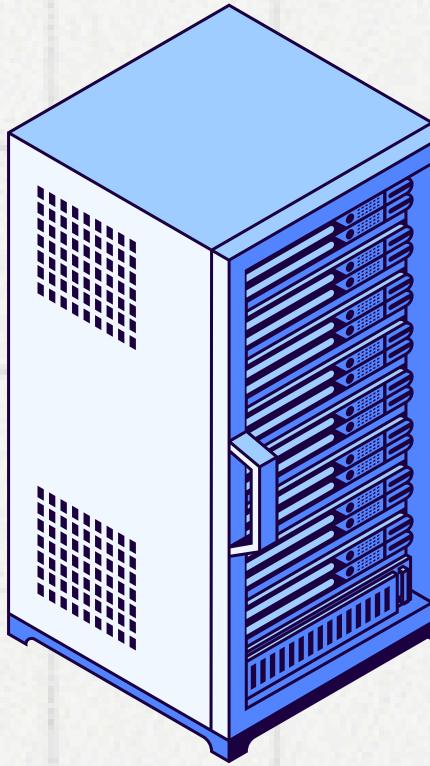


Componentes clave



Cliente(s)

- Dispositivo que realiza peticiones de recursos a un servidor. Estos pueden ser texto, contenido html, videos, imágenes, etc.
- Puede ser un ordenador o una aplicación.



Servidor

- Computadora o conjunto de computadoras que están a la espera de peticiones o solicitudes de recursos.
- Generalmente cuentan con mucha potencia de hardware ya que suelen comunicarse con varios clientes simultáneamente.



Web(Red)

- Conjunto de clientes, servidores y bases de datos unidos de una forma física o no física.
- Es donde Ocurre la comunicación

¿Cómo funciona?

Para que ocurra la comunicación entre cliente y servidor se necesita respetar una serie de reglas mas conocidas como protocolo.

El protocolo mas usado en esta arquitectura es http (Hypertext Transfer Protocol).

Comunicación paso a paso:

Paso 1:

- El cliente abre la comunicación haciendo una solicitud.

Paso 2 :

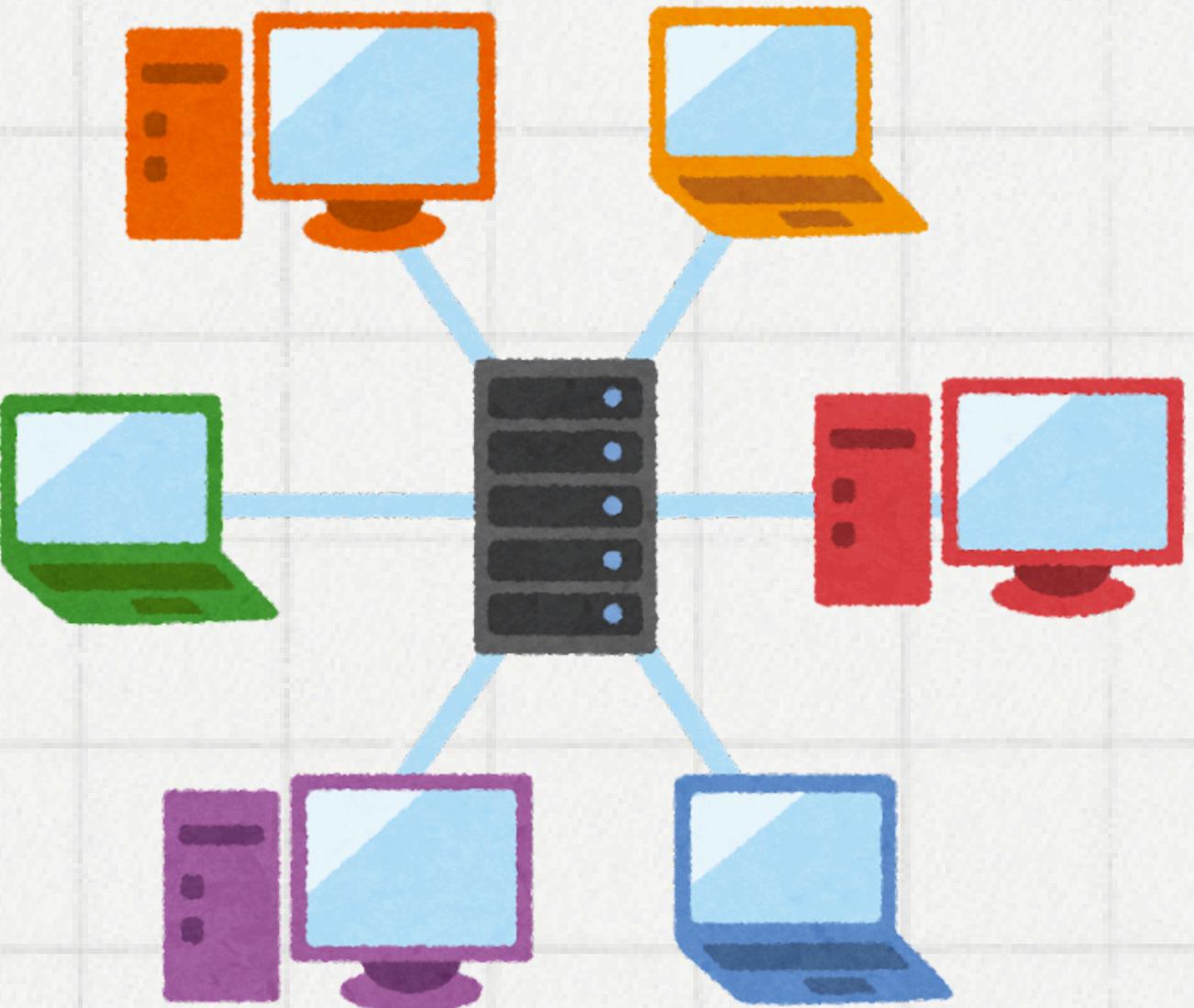
- El servidor recibe la solicitud del cliente y la interpreta para poder elegir la acción a realizar.

Paso 3:

- Después de revisar la solicitud, el servidor prepara una respuesta adecuada, con los datos o informacion requerida.
- El servidor envía la respuesta al cliente a través de la red.

Paso 4:

- La información es recibida por el cliente.

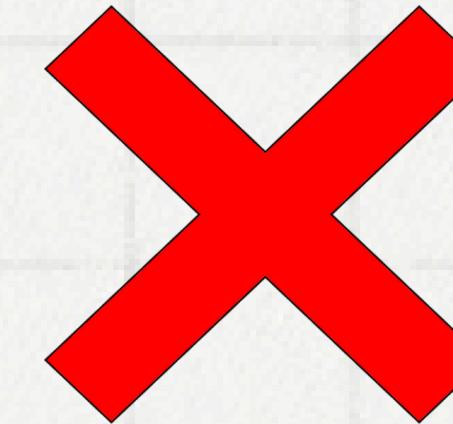


Ventajas



- **Centralización:** El servidor actúa como la única fuente de verdad, lo que garantiza que los clientes siempre accedan a la información actualizada y se evite la dispersión de datos desactualizados.
- **Seguridad:** El servidor suele estar protegido por firewalls o subredes que dificultan el acceso directo de atacantes a la base de datos o recursos, asegurando que cualquier acceso deba pasar primero por el servidor.
- **Facilidad de Instalación (Cliente):** El cliente generalmente es una aplicación sencilla con pocas o ninguna dependencia, lo que facilita su instalación y despliegue en distintos entornos.
- **Separación de Responsabilidades:** Esta arquitectura permite implementar la lógica de negocio de manera independiente del cliente, facilitando el desarrollo, mantenimiento y escalabilidad del sistema.
- **Portabilidad:** Al tener aplicaciones separadas, es posible desarrollar cada componente para que funcione en distintas plataformas. Por ejemplo, el servidor puede estar optimizado para Linux, mientras que el cliente puede ser multiplataforma, compatible con diferentes sistemas operativos.

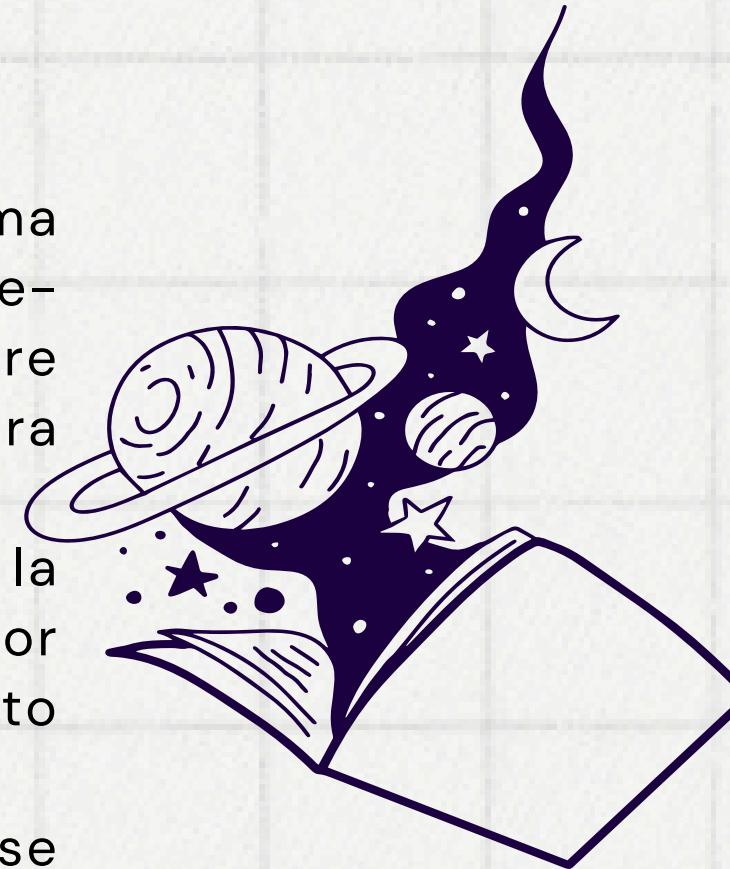
Desventajas



- **Actualizaciones (clientes):** Una de las complicaciones es gestionar las actualizaciones en los clientes, pues puede haber muchos terminales con el cliente instalado y tenemos que asegurar que todas sean actualizadas cuando salga una nueva versión.
- **Concurrencia:** Una cantidad no esperada de usuarios concurrentes puede ser un problema para el servidor, quien tendrá que atender todas las peticiones de forma simultánea, aunque se puede mitigar con una estrategia de escalamiento, siempre será un problema que tendremos que tener presente.
- **Todo o nada:** Si el servidor se cae, todos los clientes quedarán totalmente inoperables.
- **Protocolos de bajo nivel:** Los protocolos más utilizados para establecer comunicación entre el cliente y el servidor suelen ser de bajo nivel, como Sockets, HTTP, RPC, etc. Lo que puede implicar un reto para los desarrolladores.
- **Depuración:** Es difícil analizar un error, pues los clientes están distribuidos en diferentes máquinas, incluso, equipos a los cuales no tenemos acceso, lo que hace complicado recopilar la traza del error.

Por qué Elegí Este Patrón para Mi Proyecto?

1. Relevancia para la Plataforma Educativa: La arquitectura cliente-servidor permite la interacción entre estudiantes y profesores de manera eficiente y centralizada.
2. Facilidad de Uso: La centralización de la lógica de negocio en el servidor simplifica el desarrollo y mantenimiento de la aplicación.
3. Escalabilidad: Esta arquitectura se adapta bien a la necesidad de escalar la aplicación conforme aumenten los usuarios..



Ejemplo de uso

Descripción de Cómo se Aplica en "Explorando el Universo":

1. Clientes: Navegadores web o aplicaciones móviles que permiten a los estudiantes y profesores interactuar con la plataforma.
2. Servidor: Un servidor web que gestiona las solicitudes de los clientes, realiza lógica de negocio, y se comunica con la base de datos para procesar la información.

Conclusiones

La arquitectura cliente-servidor es un modelo flexible y fundamental para el desarrollo de aplicaciones, permitiendo una clara separación de responsabilidades entre el cliente y el servidor. Esta estructura facilita el mantenimiento y mejora la gestión de grandes volúmenes de usuarios y datos de manera centralizada. Dado que estamos desarrollando una plataforma de educación astronómica, esta arquitectura es particularmente adecuada, ya que garantiza eficiencia y centralización en la interacción de los usuarios. Sin embargo, su eficacia depende de la calidad de la red y la capacidad del servidor para manejar múltiples solicitudes concurrentes. En conjunto, la arquitectura cliente-servidor sigue siendo una elección confiable para aplicaciones que requieren alta disponibilidad, seguridad y escalabilidad."

**Muchas
gracias!**

