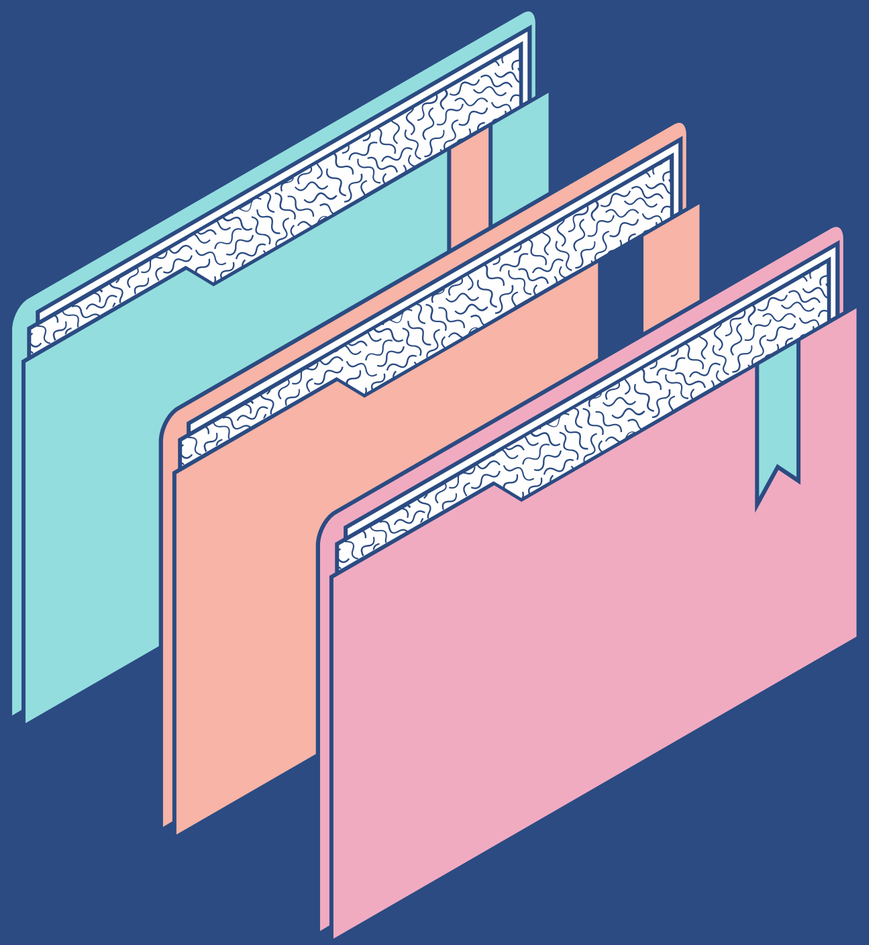
EVAN ALBERTO AGUILAR GARCIA

modelo cliente-servidor

Una mirada a la importancia de la tecnología tanto en el cliente como el servidor

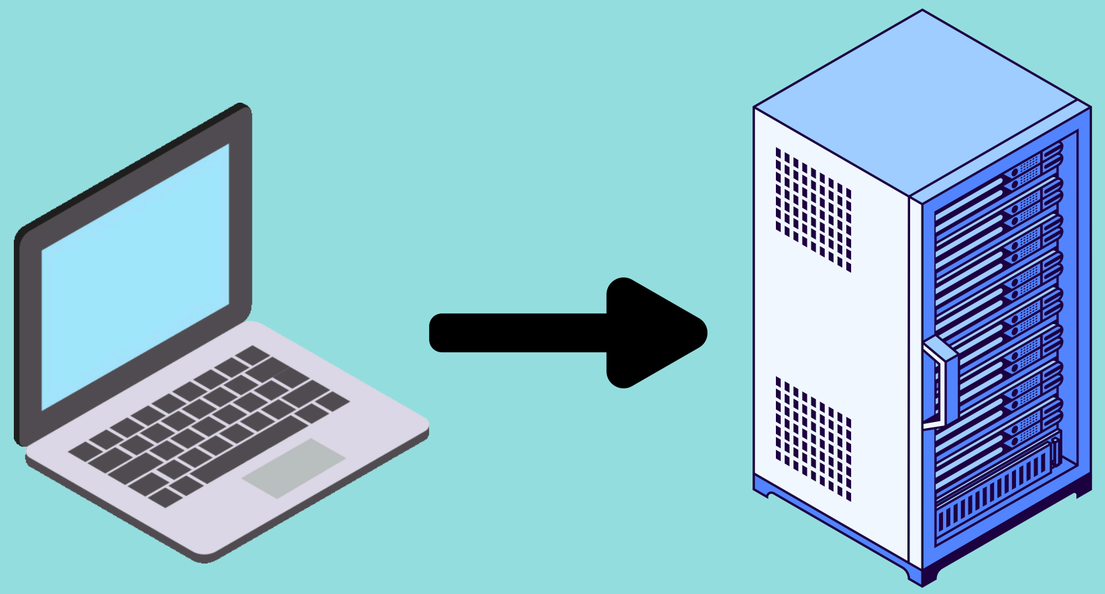
Índice



TEMAS CLAVE QUE SE HABLARAN
EN ESTA PRESENTACION

Introducción

El modelo cliente-servidor, un paradigma fundamental en informática, establece la estructura esencial para la interacción entre programas en una red. En este enfoque, los programas se dividen en dos roles principales: los clientes, que solicitan servicios o recursos, y los servidores, que proveen dichos servicios o recursos en respuesta a las solicitudes de los clientes. Esta división facilita la distribución eficiente de tareas y la gestión de recursos en entornos de red, permitiendo una comunicación fluida y organizada entre los componentes de un sistema informático



1. Importancia del modelo cliente-servido

- Este modelo resulta crucial por diversas razones:

- **Escalabilidad**: Permite distribuir la carga de trabajo entre múltiples servidores, posibilitando el crecimiento de la red sin afectar el rendimiento.

- **Flexibilidad**: Facilita la incorporación de nuevos servicios y la adaptación a cambios en las necesidades de los usuarios.

- **Modularidad**: Promueve la organización del código en módulos independientes, simplificando el desarrollo y mantenimiento del software.

- **Acceso remoto**: Brinda la posibilidad de acceder a recursos y servicios desde cualquier lugar con conexión a la red.

- **Compartir recursos**: Permite compartir recursos de manera eficiente entre múltiples usuarios

2. Componentes

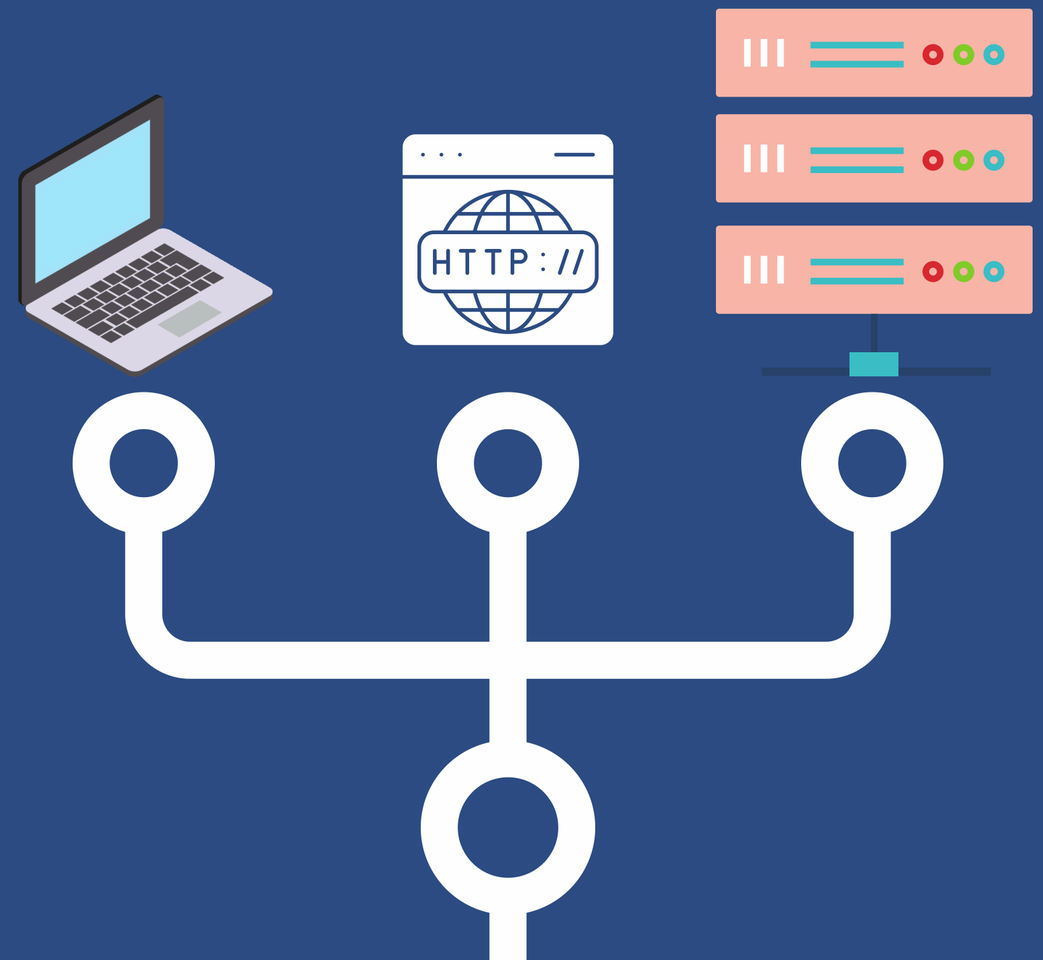
- Los componentes básicos del modelo cliente-servidor son:

- **Clientes**: Programas que inician solicitudes a los servidores para obtener recursos o servicios.

- **Servidores**: Programas que reciben las solicitudes de los clientes, procesan la información y envían las respuestas correspondientes.

- **Red**: La infraestructura de comunicación que conecta a los clientes con los servidores.

- **Protocolos**: Reglas que definen la forma en que los clientes y servidores se comunican entre sí.

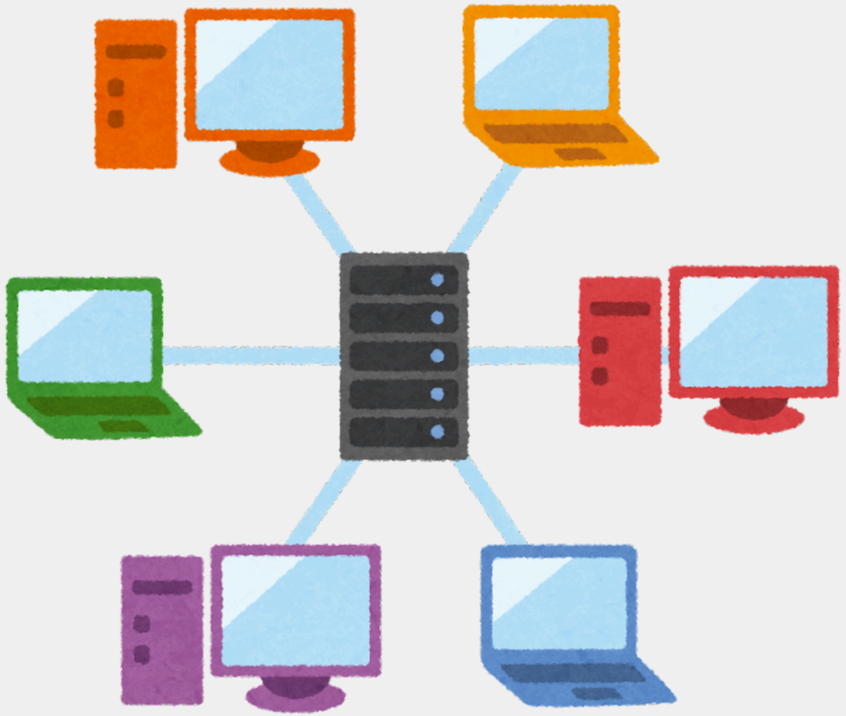


3. Diferencia entre cliente y servidor

- La principal diferencia radica en su función:

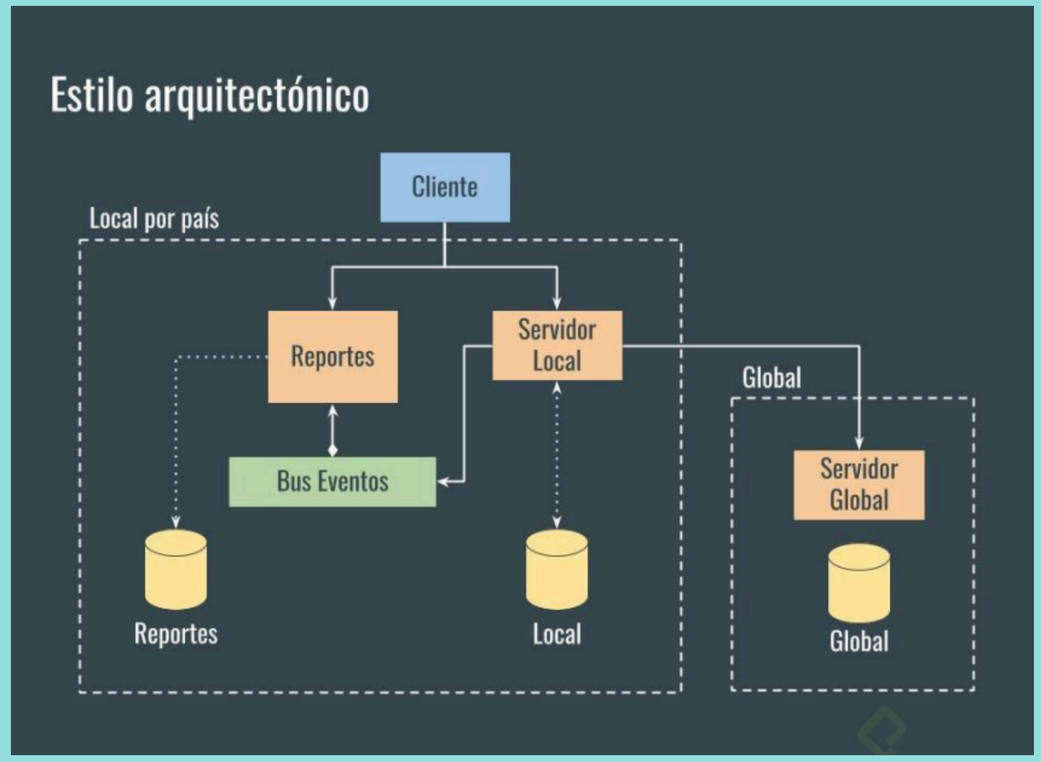
- **Clientes: Inician la interacción, solicitan recursos o servicios y reciben las respuestas.**

- **Servidores: Responden a las solicitudes de los clientes, procesan la información y envían las respuestas**



4. Tipos de arquitecturas cliente-servidor

Existen diversas arquitecturas cliente-servidor, cada una con sus características y aplicaciones. En este caso, solo investigaremos las arquitecturas cliente servidor solicitadas en la asignación: **Arquitectura en dos capas**, **Arquitectura en tres capas** y **arquitectura en N capas.**



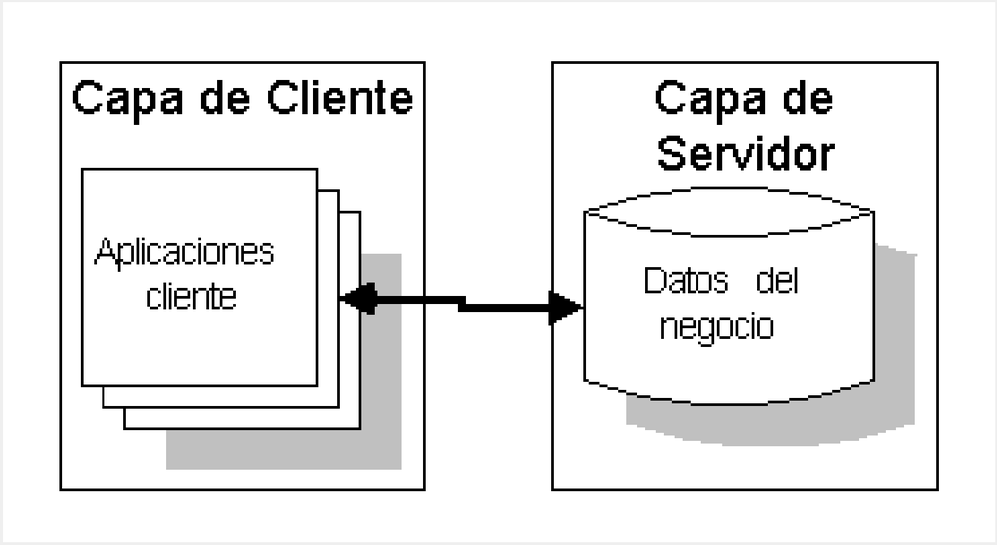
4.1 Arquitectura de dos capas

La arquitectura de dos capas, en su esencia, es la más simple y directa de las arquitecturas cliente-servidor. Su belleza radica en su minimalismo, compuesto únicamente por dos componentes:

- **Clientes**: Los iniciadores de la interacción, solicitando recursos o servicios al servidor.
- **Servidor**: El proveedor de recursos, procesando las solicitudes y enviando las respuestas correspondientes.

Esta simplicidad se traduce en varias ventajas:

- **Implementación y Comprensión Sencillas:** Su estructura básica la hace fácil de poner en marcha y entender, incluso para aquellos con menos experiencia.
- **Eficiencia Optimizada**: Al minimizar los intermediarios, la latencia se reduce, mejorando la velocidad de respuesta.
- **Bajo Costo de Implementación**: Requiere menos recursos de hardware y software en comparación con arquitecturas más complejas.



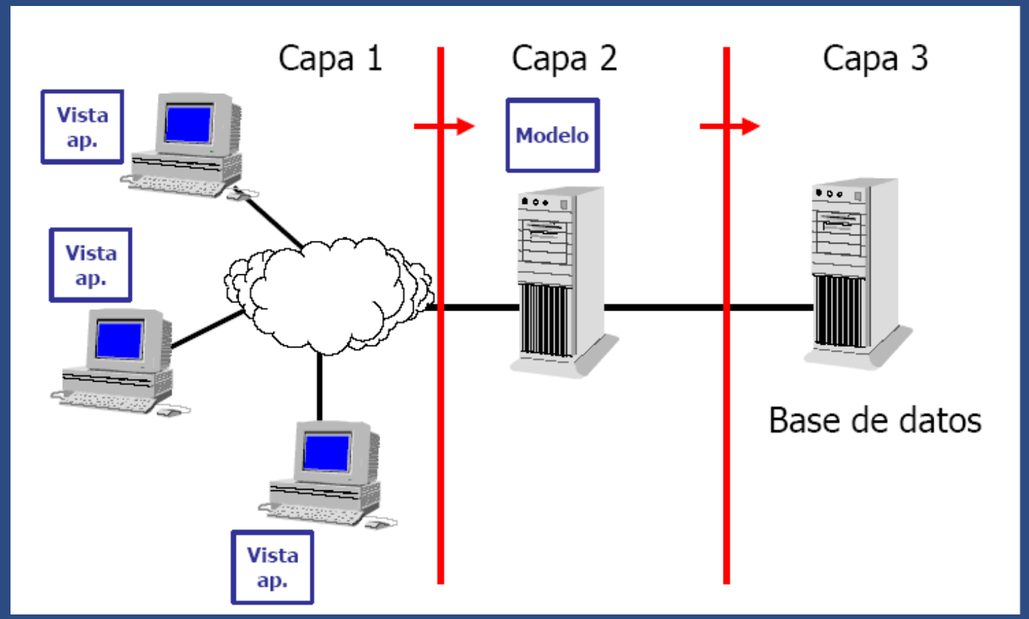
4.2 Arquitectura en tres capas

La arquitectura de tres capas introduce un refinamiento esencial: la capa de aplicación. Esta capa intermedia se interpone entre los clientes y el servidor, asumiendo responsabilidades específicas:

- **Procesamiento de la Lógica de Negocio**: Encapsula la lógica central de la aplicación, aislando al servidor de la complejidad.
- **Comunicación con Bases de Datos**: Gestiona la interacción con las bases de datos, optimizando el acceso a la información.
- **Validación de Datos**: Implementa reglas para garantizar la integridad y seguridad de los datos.

Esta arquitectura aporta beneficios considerables:

- **Seguridad Mejorada**: La capa de aplicación protege la lógica de negocio del servidor, dificultando ataques no autorizados.
- **Escalabilidad Optimizada**: Se pueden agregar más servidores de aplicación para distribuir la carga de trabajo y mejorar el rendimiento.
- **Reutilización del Código**: La lógica de negocio encapsulada facilita su reutilización en otras aplicaciones.
- **Flexibilidad Aumentada:** Permite cambios independientes en cada capa sin afectar a las demás.



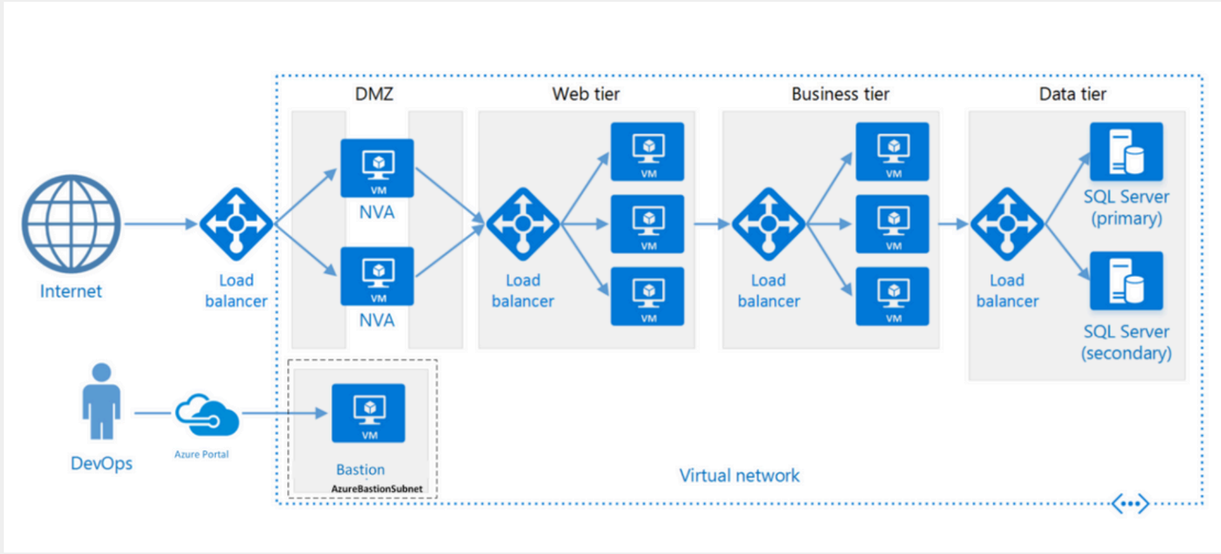
4.3 Arquitectura en N capaz

La arquitectura N capas lleva el concepto de modularidad al extremo, dividiendo la aplicación en múltiples capas especializadas. Cada capa se encarga de una función específica, como la presentación, la lógica de negocio o el acceso a datos.

Esta arquitectura ofrece una flexibilidad sin precedentes:

- **Modularización Extrema**: Permite dividir la funcionalidad en capas específicas, facilitando su desarrollo y mantenimiento.

- **Reutilización Óptima del Código**: Las capas se pueden reutilizar fácilmente en diferentes partes.



5.0, 5.1 Ventajas.

1. **Escalabilidad**:
   - Ejemplo: Un sitio web de comercio electrónico con una base de usuarios en constante crecimiento puede agregar fácilmente más servidores para manejar el aumento de tráfico sin afectar el rendimiento general.
   - Beneficio: Esta capacidad de escalar permite que las aplicaciones cliente-servidor se adapten a un número cada vez mayor de usuarios y a demandas de procesamiento más complejas.
2. **Flexibilidad**:
   - Ejemplo: Una aplicación empresarial puede modificar su lógica de negocio en la capa de aplicación sin necesidad de realizar cambios en la interfaz de usuario o en la capa de acceso a datos.
   - Beneficio: Esta flexibilidad permite que las aplicaciones se adapten a cambios en los requisitos del negocio o en las tecnologías subyacentes sin necesidad de reescribir todo el código.
3. **Modularidad**:
   - Ejemplo: Un equipo de desarrolladores puede trabajar en diferentes capas de una aplicación cliente-servidor de forma simultánea, sin interferir entre sí.
   - Beneficio: La modularidad facilita el desarrollo, mantenimiento y actualización de aplicaciones complejas, ya que cada capa puede ser tratada como un módulo independiente.
4. **Acceso Remoto**:
   - Ejemplo: Los empleados pueden acceder a los datos y recursos de la empresa desde cualquier lugar con conexión a internet, utilizando dispositivos móviles o computadoras portátiles.
   - Beneficio: El acceso remoto mejora la productividad y la colaboración, permitiendo que los empleados trabajen desde cualquier lugar y encualquier momento.
5. **Uso Compartido de Recursos**:
   - Ejemplo: Múltiples usuarios pueden acceder y utilizar los mismos recursos, como bases de datos, impresoras o software, de manera eficiente.
   - Beneficio: El uso compartido de recursos optimiza la utilización de los recursos informáticos y reduce costos.

5.2 Desventajas

1. **Dependencia del Servidor**:
   - Ejemplo: Si el servidor falla, toda la aplicación se vuelve inoperable, lo que puede tener graves consecuencias para las empresas que dependen de ella.
   - Desventaja: La dependencia del servidor central introduce un único punto de fallo, lo que hace que la aplicación sea vulnerable a interrupciones.
2. **Seguridad**:
   - Ejemplo: Un servidor que almacena datos confidenciales puede ser un objetivo atractivo para ataques cibernéticos, lo que pone en riesgo la información de la empresa y de los usuarios.
   - Desventaja: La seguridad del servidor y de la red es crucial para proteger los datos y recursos, pero puede ser un desafío complejo y cos toso de abordar. •
3. **Complejidad**:
   - Ejemplo: Las arquitecturas cliente-servidor complejas, como la de tres capas o N capas, pueden ser difíciles de implementar, administrar y mantener, especialmente para equipos pequeños o con poca experiencia.
   - Desventaja: La complejidad puede generar problemas de rendimiento, errores y dificultades para realizar cambios o actualizaciones en la aplicación.
4. **Costos**:
   - Ejemplo: La implementación y el mantenimiento de una infraestructura cliente-servidor, incluyendo hardware, software, licencias y personal calificado, pueden implicar costos significativos.
   - Desventaja: Los costos iniciales y continuos pueden ser un obstáculo para pequeñas empresas o proyectos con presupuestos limitados.

6. Ejemplos de modelo cliente servidor

El modelo cliente-servidor se utiliza en una amplia gama de aplicaciones, incluyendo:

- **Correo electrónico**: Los clientes de correo electrónico se conectan a servidores de correo para enviar y recibir mensajes.
- **Páginas web:** Los navegadores web actúan como clientes que solicitan páginas web a servidores web.
- **Aplicaciones web:** Las aplicaciones web se ejecutan en servidores y son accedidas por clientes a través de navegadores web.
- **Servicios de almacenamiento en la nube**: Los clientes de almacenamiento en la nube sincronizan archivos con servidores remotos.
- **Mensajería instantánea**: Los clientes de mensajería instantánea se conectan a servidores para comunicarse con otros usuarios

7. Recursos para aprender:

Sitios web:

- Wikipedia: https://es.wikipedia.org/wiki/Cliente-servidor
- MDN Web Docs: https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview

Vídeos:

- YouTube - FaztTech: https://www.youtube.com/watch?v=49zdlyLSwhQ
- YouTube - EDX: https://www.youtube.com/watch?v=49zdlyLSwhQ

Libros:

- "Redes de Computadoras" de Andrew S. Tanenbaum: https://www.amazon.com.mx/Redes-Computadoras-Tanenbaum/dp/6073208170
- "Arquitectura de Software Cliente-Servidor" de Robert Stevens: https://www.amazon.com.mx/Building-Microservices-Designing-Fine-GrainedSystems/dp/1491950358

Cursos en línea:

- Coursera - "Arquitectura de Sistemas Distribuidos": https://www.coursera.org/courses?query=distributed%20systems
- edX - "Introducción a las Redes Informáticas": https://www.edx.org/es/learn/computer-networking/universidad-del-rosario-redes-de-computadores