REPORTES ARQUITECTURAS BÁSICAS

Bryan Rodríguez Reyes

CLIENTE SERVIDOR: Es una arquitectura de red en la que las funciones se dividen entre los proveedores de servicios y los solicitantes de servicios. Los servidores proporcionan recursos o servicios, como bases de datos, archivos o aplicaciones.

Características:

1.- Roles diferenciados: clientes, servidores.

2.- comunicación en red: interacción, protocolos.

3.- Solicitudes y servicios.

4.- Centralización: gestión de recursos.

5.- Escalabilidad: expansión, manejo de carga.

Ventajas:

1.- Seguridad: En una arquitectura cliente-servidor la centralización de recursos mejora la seguridad de maneras como: la autentificación centralizada, el cifrado de datos y lo más importante el control de acceso.

2.- Mantenimiento simplificado: actualizaciones del lado del servidor, reducción de la complejidad del cliente para así evitar los posibles errores que puedan suceder.

3.-Escalabilidad: equipo de carga, asignación de recursos, se pueden agregar varios servidores para manejar el aumento del tráfico lo que garantiza que el sistema pueda escalar para satisfacer las crecientes demandas

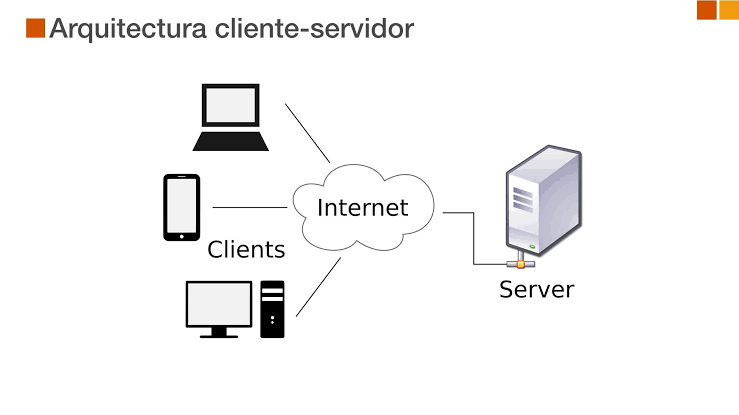
Desventajas:

1.- Dependencia del servidor: Si el servidor tiene un solo punto de procesamiento o una parte específica que es mucho más lenta en comparación con el resto del sistema, puede causar retrasos y disminuir la eficiencia general. Por ejemplo, si la CPU del servidor está constantemente al 100% de uso.

2.- Cuello de botella: ocurre cuando un componente del sistema (como el servidor, la base de datos o la red) limita el rendimiento general del sistema.

3.- Costos: Se refiere a los gastos asociados con la operación, mantenimiento y actualización del servidor, esto no es muy difícil de explicar ya que abarca cualquier gasto que se llegue a utilizar ya sea para comprar más equipo, para mejorar los dispositivos o alguna razón parecida.

4.- Reducción de rendimiento: Esto puede ocurrir debido a varios factores, como una alta carga de trabajo, limitaciones en la capacidad del hardware, problemas de red o software lento o que no cumple con el estándar. Cuando un servidor enfrenta una reducción en el rendimiento, puede experimentar tiempos de respuesta más largos, disminución en la capacidad para manejar múltiples solicitudes simultáneamente y en general una experiencia de usuario degradada.

Diagrama:

Three-tier: La arquitectura de tres niveles es una estructura de diseño de software que separa una aplicación en tres capas distintas: la capa de presentación, la capa de lógica de negocio y la capa de datos. Esta separación permite una mejor organización, escalabilidad y mantenimiento de las aplicaciones.

Características:

Capa de Presentación capa 1:

1.- Función: Interactúa con el usuario final, mostrando datos y recibiendo entrada, algunas tecnologías comunes o que aceptan la capa serían: HTML, CSS, JavaScript.

2.- Capa de Lógica de Negocio capa 2:   
Función: Procesa la lógica del negocio y las reglas de la aplicación, manipulando datos y ejecutando operaciones basadas en las solicitudes de la capa de presentación, algunas tecnologías comunes serían, Lenguajes de programación (Java, C#, Python, PHP).

3.- Capa de Datos capa 3:

Función: Almacena, recupera y gestiona los datos, interactuando con bases de datos y sistemas de almacenamiento, algunas tecnologías comunes son: sistemas de gestión de bases de datos como MySQL, sql.

Ventajas:

1. Escalabilidad: Permite escalar cada capa de forma independiente. Por ejemplo, se pueden añadir más servidores para la capa de datos si es necesario.
2. ﻿﻿﻿Reusabilidad: Las capas de lógica de negocio y datos pueden ser reutilizadas en diferentes aplicaciones, esto nos ahorra mucho tiempo y bastantes preocupaciones.
3. ﻿﻿﻿Facilidad de Mantenimiento: Los cambios en una capa tienen un impacto limitado en otras capas, simplificando las actualizaciones y correcciones, esto es de gran ayuda ya que el mantenimiento resulta muy fácil.
4. Seguridad Mejorada: Se pueden implementar controles de acceso y seguridad para cada capa, proporcionando una mejor protección y disminuyendo los riesgos.

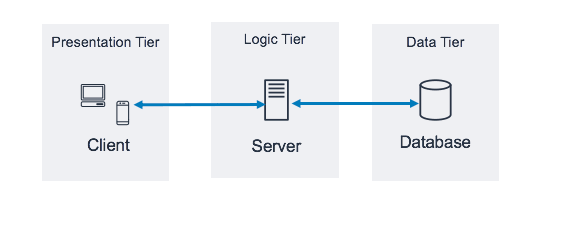
Desventajas:

1. Complejidad: La separación en tres capas puede introducir una complejidad adicional en el diseño y la gestión de la aplicación.
2. ﻿﻿﻿Rendimiento: La comunicación entre capas puede introducir sobrecarga y latencia, lo que puede afectar el rendimiento, especialmente en aplicaciones con alta demanda.
3. ﻿﻿﻿Desarrollo y Pruebas: Requiere una mayor coordinación entre los equipos que desarrollan y prueban cada capa, lo que puede complicar el proceso.
4. ﻿﻿﻿Costos: La implementación de una arquitectura de tres niveles puede requerir una inversión inicial más alta en términos de infraestructura y desarrollo.

Resumen:

En resumen, la arquitectura de tres niveles ofrece una estructura organizada y escalable para aplicaciones complejas, aunque puede implicar desafíos en términos de complejidad y rendimiento.

Diagrama:

referencia:

[docs.aws.amazon.com](https://docs.aws.amazon.com/es_es/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html" \t "_self)

N.tier: es un modelo de diseño de software en el que la aplicación se divide en varias capas o niveles distintos, cada uno con una función específica. Generalmente, estas capas incluyen:

1.-Capa de presentación: Interfaz de usuario.

﻿﻿﻿2.- Capa de lógica de negocio: Procesos y reglas del negocio.

﻿﻿﻿3.- Capa de acceso a datos: Interacción con la base de datos.

﻿﻿﻿4.- Capa de almacenamiento: Bases de datos y almacenamiento de datos.

Características:

1.- Separación de responsabilidades: Cada capa se encarga de una función específica, lo que mejora la . organización del código

* ﻿﻿2.- Modularidad: Las capas son modulares y pueden ser desarrolladas, probadas y mantenidas de forma independiente.
* ﻿﻿3.- Escalabilidad: Facilita la expansión de la aplicación mediante la adición de nuevas capas o la mejora de las existentes.
* ﻿﻿4.- interoperabilidad: Permite que diferentes componentes interactúen a través de interfaces bien definidas.

Ventajas:

1.- Mantenimiento mejorado: Cambios en una capa no afectan directamente a otras capas.

2.- Reusabilidad: Las capas pueden ser reutilizadas en diferentes aplicaciones o contextos.

3.- Escalabilidad: Facilita la escalabilidad horizontal y vertical de la aplicación.

﻿﻿. 4.- Desarrollo independiente: Los equipos pueden trabajar en diferentes capas de manera simultánea, mejorando la eficiencia del desarrollo.

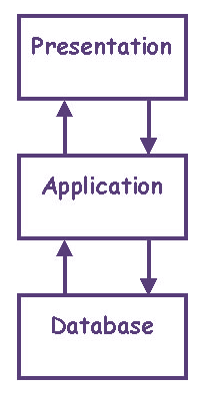
Desventajas:

* Complejidad: La estructura en capas puede introducir una complejidad adicional en la arquitectura.
* ﻿﻿Rendimiento: Las múltiples capas pueden agregar sobrecarga en el rendimiento debido a la necesidad de comunicación entre ellas.
* ﻿﻿Sobreingeniería: Puede ser excesiva para aplicaciones simples, llevando a una complejidad innecesaria.

Resumen:

Este enfoque es común en aplicaciones empresariales y sistemas de gran escala donde la gestión y la separación de responsabilidades son cruciales

Diagrama:



Peer-to-peer:

es un modelo de red en el que los nodos (o pares) interactúan directamente entre sí, sin necesidad de un servidor centralizado.

Características:

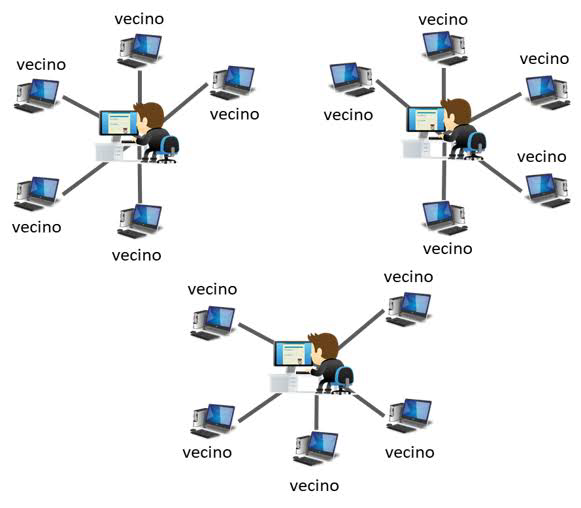
1. Descentralización: No existe un servidor central; todos los nodos tienen roles equivalentes.
2. ﻿﻿﻿Autonomía: Los nodos pueden operar de manera independiente y comunicar sus propios recursos.
3. ﻿﻿﻿Escalabilidad: La red puede crecer de manera eficiente, ya que cada nuevo nodo contribuye con recursos adicionales.
4. Redundancia: Los datos y servicios se distribuyen entre varios nodos, reduciendo el riesgo de fallos únicos.

Ventajas:

1. ﻿﻿﻿Escalabilidad: A medida que se añaden más nodos, la capacidad de la red y la disponibilidad de recursos aumentan.
2. ﻿﻿﻿Resiliencia: La red es más resistente a fallos, ya que la pérdida de un nodo no afecta gravemente a la red en su conjunto.
3. ﻿﻿﻿Costo-efectividad: No requiere servidores centralizados costosos; cada nodo contribuye a la infraestructura.

Desventajas:

1. Seguridad: La falta de un control
2. centralizado puede hacer que sea más difícil implementar y mantener medidas de seguridad efectivas.
3. ﻿﻿﻿Consistencia: La sincronización y la coherencia de los datos pueden ser un desafío, especialmente en redes grandes.
4. ﻿﻿﻿Gestión: La administración y el mantenimiento de la red pueden ser más complejos debido a la naturaleza descentralizada.
5. ﻿﻿﻿Rendimiento: En redes grandes, el rendimiento puede verse afectado por la sobrecarga de la comunicación entre nodos.

Diagrama

Loose coupling / tight coupling

* Loose Coupling (Acoplamiento Débil): Es una práctica de diseño en la que los componentes de un sistema tienen dependencias mínimas entre sí.
* Tight Coupling (Acoplamiento Fuerte): Es una práctica en la que los componentes están estrechamente conectados y dependen mucho unos de otros.

Características:

Loose Coupling:

Independencia: Los componentes pueden funcionar y evolucionar de manera independiente.

* ﻿﻿Interacción a través de Interfaces: Los componentes interactúan mediante

interfaces definidas, lo que minimiza el conocimiento mutuo.

* ﻿﻿Flexibilidad: Facilita la integración de nuevos componentes o la modificación de los existentes sin afectar al sistema  
  en general.
* Tight Coupling:

Dependencia Mutua: Los componentes están altamente interdependientes y  
conocen los detalles internos de los otros.

Ventajas:

Loose Coupling

Flexibilidad y Mantenibilidad: Los cambios en un componente tienen menos probabilidad de afectar a otros, lo que facilita la evolución del sistema.

* ﻿﻿Escalabilidad: Los componentes pueden ser reemplazados o

actualizados sin necesidad de modificar el sistema en su totalidad.

* ﻿﻿Reusabilidad: Los componentes desacoplados pueden ser reutilizados en diferentes contextos.
* Desventajas:
* ﻿﻿Sobrecarga de Interfaz: Puede haber una mayor complejidad en la definición y gestión de las interfaces.
* ﻿﻿Rendimiento: La interacción a través de interfaces puede introducir una ligera sobrecarga en términos de rendimiento.

Tight Coupling:

﻿﻿Ventajas:

* ﻿﻿Simplicidad en la Implementación:  
  Puede ser más sencillo de implementar inicialmente debido a la interacción directa y menos abstracción.
* ﻿﻿Rendimiento: La interacción directa puede ser más eficiente en términos de rendimiento, ya que no hay una capa de abstracción adicional.

Desventajas:

* ﻿﻿Rigidez: Cambios en un componente pueden requerir cambios en otros, lo que puede complicar la evolución del sistema.
* ﻿﻿Menor Reusabilidad: Los componentes altamente acoplados tienden a ser menos reutilizables en  
  diferentes contextos.
* ﻿﻿Mantenibilidad: Puede ser más difícil de mantener debido a las fuertes dependencias entre componentes.

Diagrama:

