UNIVERSIDAD LAICA ELOY ALFARO DE MANABÍ EXT. EL CARMEN

TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN

MATERIA:
sistemas distribuidos

PROFESOR:
ing. Cesar Sinchiguano

8VO SEMESTRE

TEMA: Modelos de arquitectura

CHICA GOMEZ RUTH ESTEFANIA

INTRODUCCION

LAS ESTRUCTURAS QUE DESCRIBEN CÓMO SE ORGANIZAN Y COORDINAN LAS INTERACCIONES ENTRE COMPONENTES DISTRIBUIDOS SE CONOCEN COMO MODELOS DE ARQUITECTURA DE SISTEMAS DISTRIBUIDOS. ESTOS MODELOS SON ESENCIALES PORQUE DETERMINAN LA SEGURIDAD, LA CAPACIDAD DE RESPUESTA, LA TOLERANCIA A FALLOS Y LA EFICIENCIA DE LOS SISTEMAS DISTRIBUIDOS.

Modelo Cliente - Servidor

- Este modelo se caracteriza por la comunicación entre un cliente que solicita servicios y un servidor que los proporciona. El servidor atiende las peticiones de los clientes de forma independiente.

Modelo Cliente - Servidor ventajas y desventajas

- Escalabilidad
- Facilidad de mantenimiento
- Seguridad
- Fiabilidad y robustez
- Los clientes pueden acceder a los servicios del servidor desde ubicaciones remotas utilizando internet o una red interna

- Dependencia del servidor y complejidad en la gestión de usuarios.
- Mayor riesgo de ataques
- El tráfico de red generado por múltiples clientes que solicitan datos al servidor puede sobrecargar la red

Modelo de Servidor Proxy

Un servidor proxy actúa como intermediario entre un cliente y un servidor, gestionando las conexiones y optimizando el acceso a los recursos.

Seguridad	Rendimiento	Privacidad
Un proxy puede bloquear el acceso a sitios web sospechosos o filtrar contenido inapropiado.	Al almacenar en caché el contenido, reduce el tiempo de carga y mejora la velocidad de acceso.	Puede enmascarar la dirección IP del cliente, protegiendo su identidad.

Ventajas y desventajas del servidor Proxy

- Privacidad y anonimato
- Control de acceso
- Mejora de la seguridad
- Aceleración del rendimiento
- Optimización del ancho de banda

- Velocidad limitada
- Compatibilidad limitada
- Posibles vulnerabilidades de seguridad
- Falta de encriptación
- Riesgo de privacidad

Peer-to-Peer

EL MODELO PEER-TO-PEER PERMITE QUE CADA NODO EN LA RED ACTÚE TANTO COMO CLIENTE COMO SERVIDOR. ESTO PROMUEVE LA DESCENTRALIZACIÓN Y LA ESCABILIDAD, FACILITANDO LA COMPARTICIÓN DE RECURSOS Y LA COLABORACIÓN ENTRE USUARIOS.

Ejemplos Comunes de Redes P2P:

BitTorrent: Usado para compartir archivos grandes de manera distribuida.

Blockchain: Las criptomonedas como Bitcoin operan en redes P2P, donde los nodos participan en la verificación de transacciones.

Skype (en sus inicios): Utilizaba un modelo P2P para la transmisión de voz y video entre usuarios.

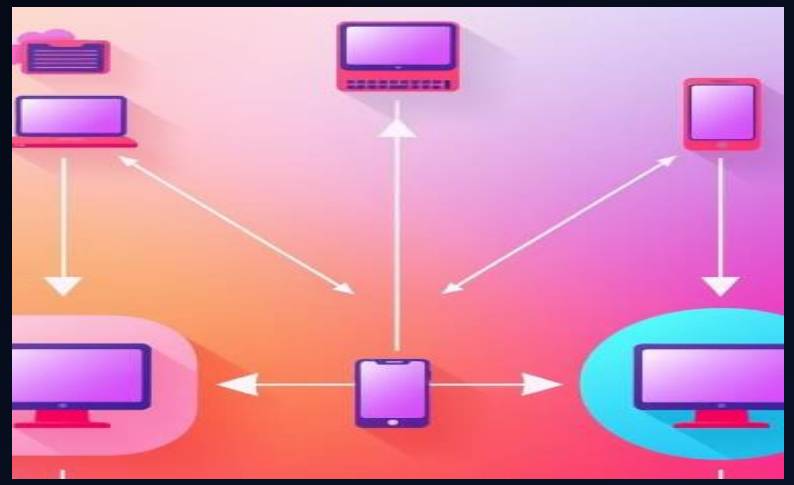
Características Peer-to-Peer

DESCARGAS

- Los usuarios pueden compartir archivos directamente entre sí, sin necesidad de un servidor central.

COLABORACIÓN

- Permite a los usuarios colaborar en proyectos y compartir información de forma eficiente.



Ventajas y desventajas Peer-to-Peer

- Las ventajas del modelo Peer-to-Peer incluyen la reducción de costos, la resiliencia ante fallos y la distribución de carga. Este modelo es ideal para aplicaciones que requieren un alto grado de interacción y colaboración entre usuarios.
- En una red Peer-to-Peer, los nodos se comunican directamente, lo que puede aumentar el riesgo de ataques como la distribución de malware o la suplantación de identidad. La falta de un control centralizado puede dificultar la implementación de medidas de seguridad efectivas.

GRID

Una arquitectura grid conecta recursos informáticos distribuidos, permitiendo compartir procesamiento, almacenamiento y otros recursos de forma transparente.

Ventajas y desventajas del modelo GRID

- Gran capacidad de procesamiento
- Uso de recursos
- Escalabilidad
- Reducción de costos

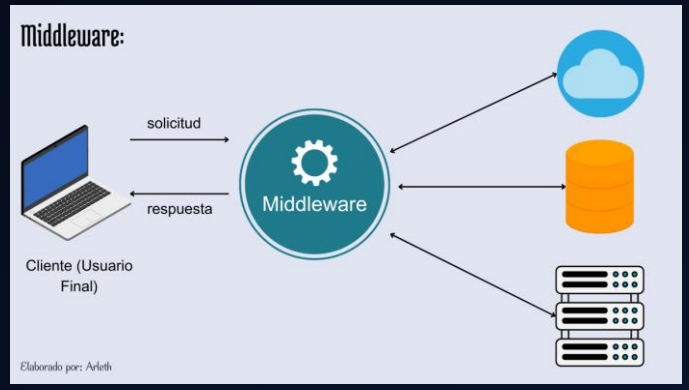
- Problemas de compatibilidad y heterogeneidad
- Dependencia de la conectividad
- Complejidad en la implementación de y gestión

MIDDLEWARE

Software que actúa como puente entre diferentes aplicaciones, sistemas operativos y plataformas, facilitando la interoperabilidad y la comunicación.

Ventajas y desventajas del modelo Middleware

- El middleware oculta la complejidad de las operaciones subyacentes en un sistema distribuido, proporcionando una interfaz simple para que las aplicaciones interactúen.
- Los desarrolladores pueden tener dificultades para identificar el origen de un problema, ya que puede estar relacionado con la comunicación en lugar de la lógica de la aplicación.



Arquitectura de Microservicios en Tres Capas

Capa de Presentación (Frontend):

- Esta es la capa más cercana al usuario y se encarga de la **interfaz de usuario** (UI). En una arquitectura de microservicios, esta capa puede ser una aplicación web, móvil, o cualquier otra interfaz gráfica.

Ejemplo: Una aplicación web que presenta un catálogo de productos o una app móvil para realizar pedidos.

Capa de Lógica de Negocio (Backend o Middleware):

- Esta capa maneja la lógica de negocio del sistema. En una arquitectura de **microservicios**, los servicios están desacoplados, es decir, cada servicio maneja una parte específica de la funcionalidad del sistema.

Ejemplo: Microservicios responsables de gestionar usuarios, productos, pagos o inventarios.

Arquitectura de Microservicios en Tres Capas

Capa de Datos (Data Layer):

- Esta es la capa que se encarga de la persistencia y gestión de los datos. En los sistemas distribuidos basados en microservicios, la arquitectura de bases de datos puede ser variada. Cada microservicio puede tener su propia base de datos, lo que sigue el principio de "Base de Datos por Microservicio".

Ejemplo: Bases de datos de usuarios, sistemas de inventario o almacenamiento de registros de pedidos.

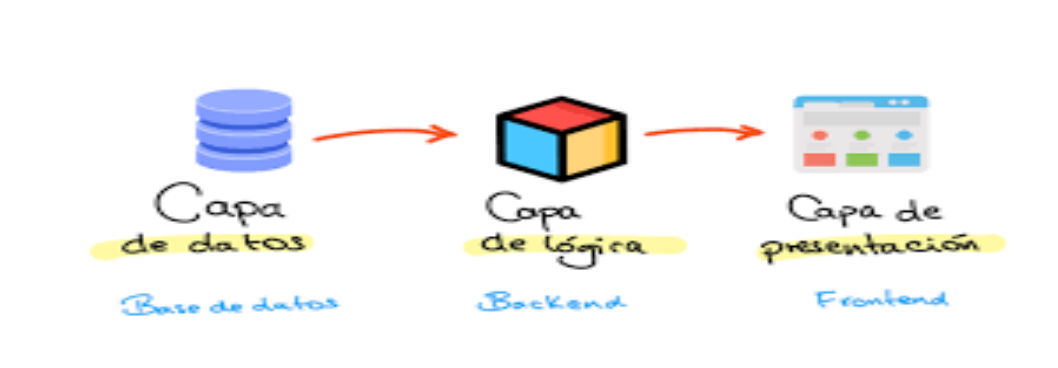


Tabla comparativa de modelos de arquitectura

Modelo	Características	Ventajas	Desventajas
Cliente-Servidor	Centralización de los servicios en un servidor que atiende múltiples solicitudes de clientes.	Control centralizado - Seguridad y administración - Facilidad de mantenimiento	Dependencia del servidor central - Punto único de fallo - Escalabilidad limitada por el servidor
Proxy	Actúa como intermediario entre el cliente y el servidor, gestionando las solicitudes y respuestas.	Ahorro de ancho de banda - Mejora de seguridad - Caché de contenidos para mayor eficiencia	Puede introducir latencia adicional - Complejidad en la configuración
Peer-to-Peer (P2P)	Cada nodo actúa como cliente y servidor, compartiendo recursos directamente entre los pares sin un servidor central.	- Distribución eficiente de la carga - Escalabilidad alta - Reducción del punto único de fallo	- Difícil de gestionar la seguridad - Problemas de redes - Propenso a ataques
Grid	Computación distribuida que utiliza recursos de múltiples computadoras para resolver problemas complejos	- Aprovechamiento de recursos - Alta capacidad de procesamiento - Flexibilidad geográfica	- Complejidad de implementación - Problemas de latencia y seguridad - Dependencia de una infraestructura de red confiable
Arquitectura en Capas	Sistema dividido en capas lógicas, cada una responsable de una funcionalidad específica (presentación, lógica de negocio, acceso a datos,	Separación de responsabilidades - Fácil mantenimiento y escalabilidad - Reutilización de componentes	Interdependencia entre capas puede generar latencia - Complejidad de implementación en sistemas grandes
Middleware	Software intermediario que facilita la comunicación y gestión de datos entre aplicaciones distribuidas	- Interoperabilidad entre sistemas heterogéneos - Simplifica la integración de sistemas - Escalabilidad	- Sobrecarga de rendimiento - Implementación y mantenimiento costoso - Complejidad en la gestión de seguridad

Conclusiones

LOS MODELOS DE ARQUITECTURAS SON ESENCIALES PARA DISEÑAR Y COMPRENDER CÓMO LOS SISTEMAS INTELIGENTES OPERAN. LA ELECCIÓN DEL MODELO ADECUADO DEPENDE DE LOS REQUISITOS DEL SISTEMA, LAS EXPECTATIVAS DE RENDIMIENTO, LA COMPLEJIDAD Y LA FACILIDAD DE GESTIÓN, LO QUE DESTACA SU PAPEL CRUCIAL EN LOS SISTEMAS DISTRIBUIDOS.

Gracias