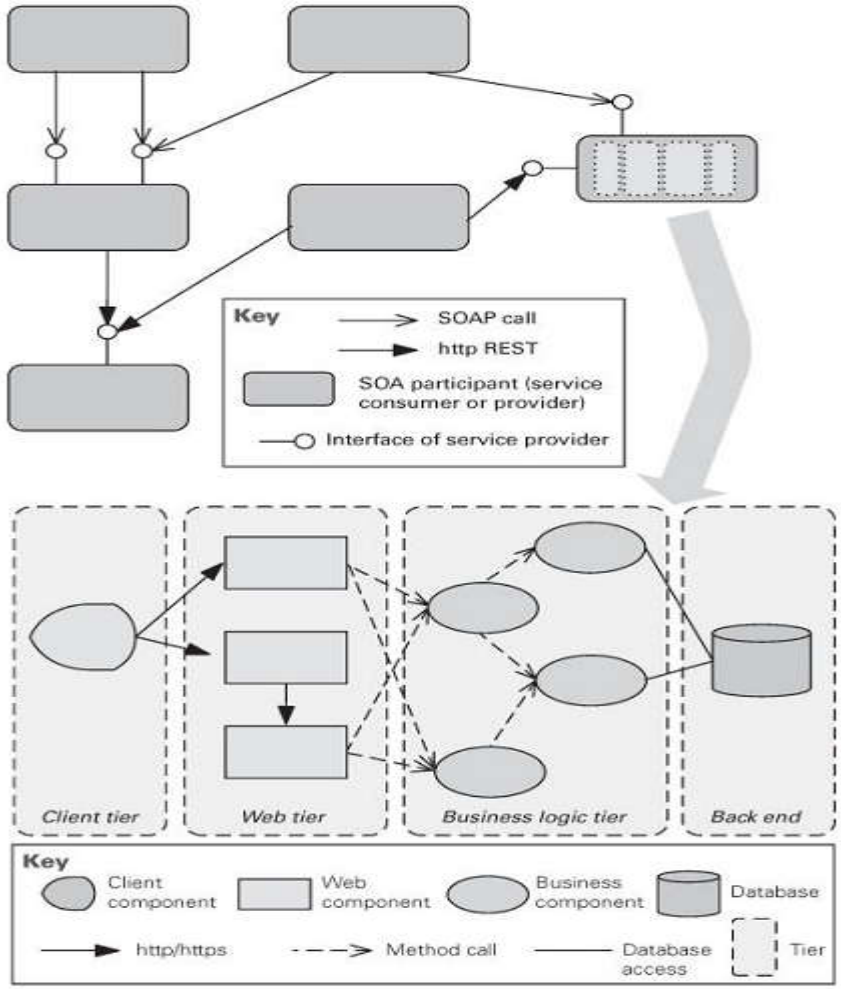
Estilos y patrones arquitectónicos

Aplicación de los Estilos Arquitectónicos

- Diferentes áreas de un sistema pueden exhibir diferentes estilos
- Un elemento que actúa como parte de un estilo puede descomponerse en elementos de otro estilo



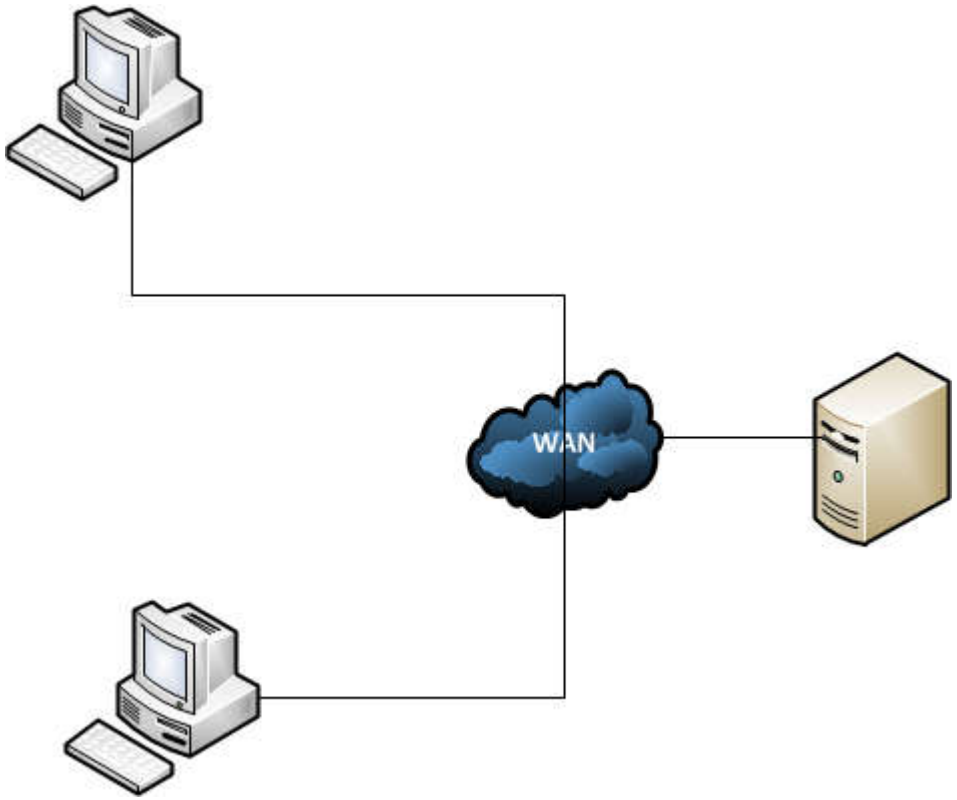
Modelo Cliente-Servidor

Cliente/Servidor son construidos para que tengan un elemento central, conocido como servidor que es compartido con múltiples usuarios que lo acceden a través de un cliente

Elementos:

- Servidores que ofrecen servicios a otros subsistemas
- Clientes que llaman a los servicios ofrecidos por los servidores
- Una red que permita a los cliente acceder estos servicios

Estilos y Patrones Arquitectónicos: Cliente-Servidor



Estilos y Patrones Arquitectónicos: Cliente-Servidor

- Usualmente presente como:

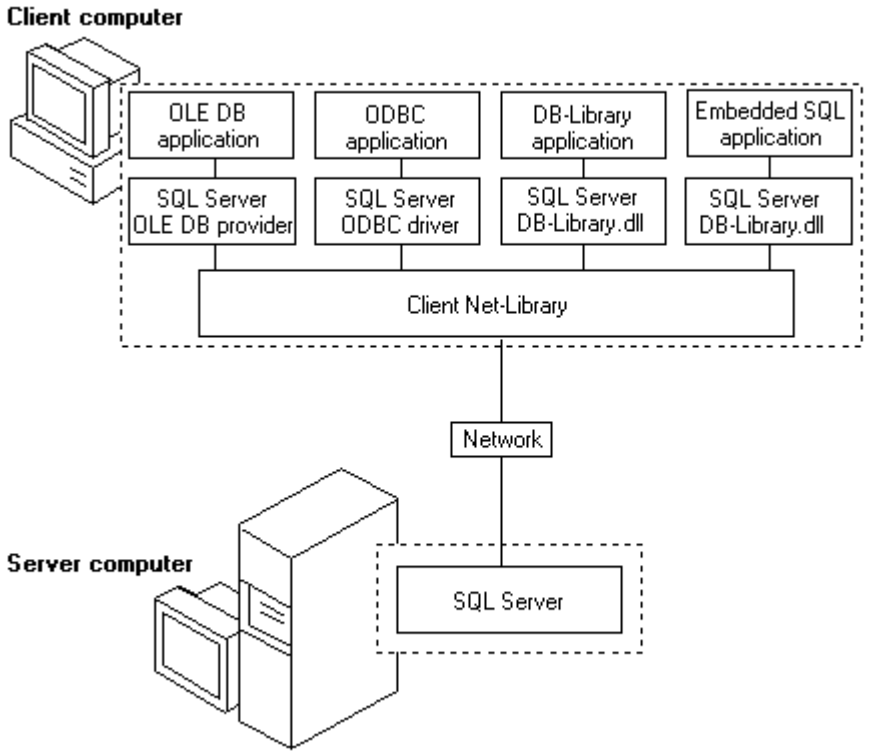
Two-Tier Client/Server

- Cliente ejecuta su aplicación en la computadora local la cual se conecta a un servidor
- El servidor típicamente implementado con un servidor de BD

Multitier Client/Server

- Un cliente liviano que se ejecuta en una computadora local enfocado en el despliegue de los datos
- La lógica de negocio se encuentra ubicada en un servidor de aplicación . El servidor de aplicación establece conexiones al servidor de BD

Estilos y Patrones Arquitectónicos: Cliente-Servidor



Estilos y Patrones Arquitectónicos

Ventajas

- Arquitectura distribuida. Se puede hacer uso efectivo de los sistemas en red con muchos procesadores distribuidos
- Fácil añadir un nuevo servicio e integrarlo con el resto del sistema
- Facilidad para recuperación y respaldo. Esto debido a la centralización

Desventajas

- Puede ser necesario realizar cambios a los clientes y servidores existentes para obtener los mayores beneficios de la integración de un nuevo servidor. Con costos asociados
- Problemas de congestionamiento en la red por un volumen no anticipado de solicitudes

Modelo Repositorio

Un almacén de datos en el centro de esta arquitectura
Los otros componentes tienen acceso a él y cuentan con la opción de actualizar, agregar, eliminar o modificar estos datos

Modelo Repositorio

- El software cliente accede a los datos independientemente de cualquier cambio hecho a los datos o las acciones de otros software

- Promueve la capacidad de integración: es posible cambiar componentes existentes y agregar nuevos componentes cliente a la arquitectura sin preocuparse por otros clientes

Modelo Repositorio

- Dos formas:
  - ✓ Todos los datos compartidos se almacenan en una base de datos central a la que puede acceder por todos los subsistemas
  - ✓ Cada subsistema mantiene su propia base de datos. Los datos se intercambian con otros subsistemas mediante el paso de mensajes entre ellos

- Adecuado para aplicaciones en las que los datos son generados por un subsistema y usados por otro

Estilos y Patrones Arquitectónicos

Ventajas

- Forma eficiente de compartir grandes cantidades de datos. No hay necesidad de transmitir datos explícitamente de un subsistema a otro
- Los subsistema que producen datos no necesitan conocer como se utilizan sus datos por otros subsistemas
- Las actividades de seguridad, protección, control de acceso y recuperación de errores están centralizados

Desventajas

- Los subsistemas deben estar acordes con el modelo de datos del repositorio. El rendimiento puede verse afectado de forma adversa por el compromiso entre las necesidades específicas de cada herramienta.
- Puede ser difícil integrar nuevos subsistemas si sus módulos no se ajustan al esquema acordado
- La evolución puede ser difícil a medida que se genera un gran volumen de información
- Diferentes subsistemas pueden tener distintos RQ de protección, recuperación y políticas de seguridad
- Difícil distribuir el repositorio sobre varias máquinas

Estilos y Patrones Arquitectónicos

Modelo Capas

Organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios

Modelo Capas

Escoja la estrategia de capas a utilizar:
Estrategia de agrupamiento, separar en pocas o en muchas. Determinar la granularidad.

Considerar la posible separación lógica de las capas

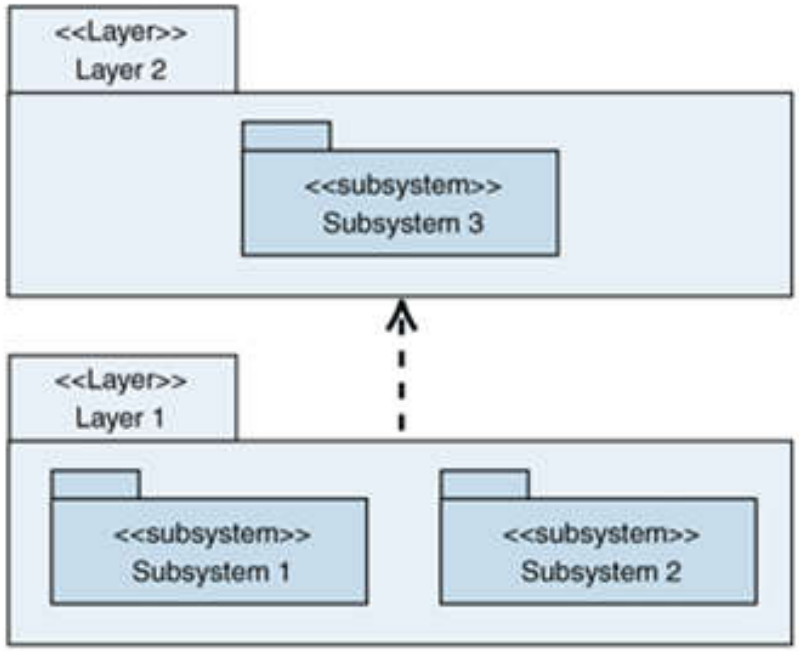
- Determine las capas que necesitará: Se puede iniciar utilizando patrones de arquitectura.
- Decida como distribuir las capas y los componentes:

Modelo Capas

- Determinar las reglas de interacción entre capas:
  - Top Down: Las capas superiores pueden interactuar con las inferiores pero las inferiores no pueden hacerlo con las superiores
  - Interacción Estricta: Cada capa debe interactuar únicamente con la inmediatamente inferior.
  - Interacción Suelta: Las capas superiores pueden pasar a otras capas inferiores. Permite mejorar el rendimiento

- Definir Interfaces entre las capas

Estilos y Patrones Arquitectónicos: Capas



Estilos y Patrones Arquitectónicos

Ventajas

- Soporta el desarrollo incremental de sistemas
- Se puede entender una capa como un conjunto coherente sin tener que conocer las otras capas
- Soporta bien los cambios y es portable
- En la medida que la interfaz permanezca sin cambios una capa puede cambiarse por otra equivalente

Desventajas

- La estructuración de los sistemas parece resultar difícil
- En algunos casos los servicios requeridos por un usuario del nivel superior puede requerir servicio de los niveles inferiores, atravesando las capas adyacentes para tener acceso a los servicios trastocando el modelo ya que la capa externa no solo depende de su predecesora
- El rendimiento puede ser un problema si hay múltiples niveles de interpretación

Modelo Orientado a Flujos (Pipe and Filter)

Estructura de tuberías y filtros: componentes denominados filtros conectados por tuberías que transmiten datos de un componente al siguiente.

Cada filtro funciona sin tomar en cuenta si los componentes tiene un flujo ascendente o descendente

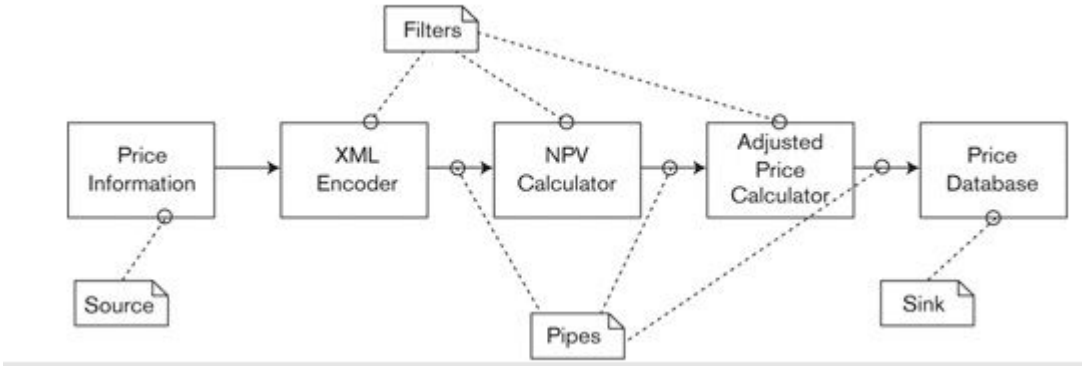
Modelo Orientado a Flujos (Pipe and Filter)

- Utilizada cuando los datos de entrada se habrán de transformar en datos de salida mediante una serie de componentes para el cálculo o la manipulación

- Diseñado para esperar la entra de datos con cierta forma u producir la salida. No es necesario que el filtro conozca el funcionamiento de los filtros vecinos

Modelo Orientado a Flujos (Pipe and Filter)

- Soluciona el problema de implementar un sistema que debe procesar datos en una secuencia de pasos donde un único proceso no es posible y donde los requerimientos de pasos procesamiento pueden cambiar a través del tiempo.
- Reordenamiento, cambio o recombinación de los pasos
- Pasos pequeños son más fáciles de reutilizar



Estilos y Patrones Arquitectónicos

Ventajas

- Permite la reutilización de transformaciones
- Es intuitiva puesto que muchas personan piensan en su trabajo en términos de procesamiento de entradas y salidas
- Generalmente se pueden hacer evolucionar de forma directa el sistema añadiendo nuevas transformaciones
- Es sencilla de implementar ya sea como un sistema concurrente o como uno secuencial

Desventajas

- Tiene que haber un formato común para transferir los datos de esta forma que puedan ser reconocidos por todas las transformaciones.
- Cada transformación debe estar acorde con las transformaciones con las que se comunica sobre el formato de los datos a procesar o debe imponer un formato estándar
- Los sistema interactivos son difíciles de describir usando el modelo de flujo de funciones debido a la necesidad de un flujo de datos a procesar

Peer-to-Peer

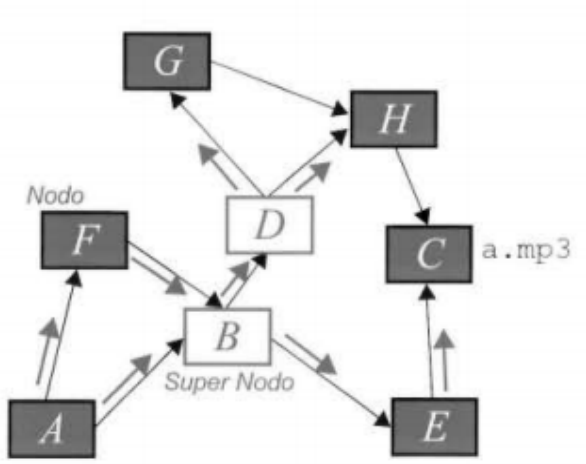
El procesamiento es logrado mediante la cooperación de pares que solicitan y proveen servicios a otros a través de la red.
Cada uno de los pares es un componente independiente corriendo en la red

Peer-to-Peer

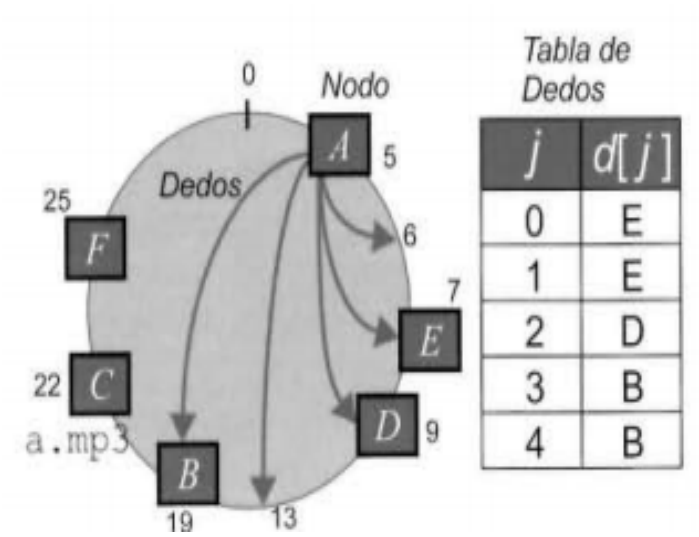
- Cada entidad distribuida es considerada igualmente importante en términos de iniciar una interacción y cada una proveerá sus propios recursos.
- Los conectores proveen interacción bidireccional.
- Los peers primero se conectan a la red peer-to-peer para descubrir otros elementos pares con los que puedan interactuar e iniciar acciones
- En algunas ocasiones existen nodos especializados (supernodos) que tienen capacidades de indexación o enrutamiento para asegurar que los otros nodos puedan acceder un número mayor de nodos pares

Peer-to-Peer No Estructurados

- Llamados no estructurados porque no hay restricción alguna del lugar donde un archivo particular debe ser almacenado



Peer-to-Peer Estructurados



- Los sistemas P2P estructurados logran minimizar el número de mensajes requeridos para satisfacer una búsqueda, pero imponen un régimen menos flexible en cuanto a la ubicación de los objetos.

Estilos y Patrones Arquitectónicos

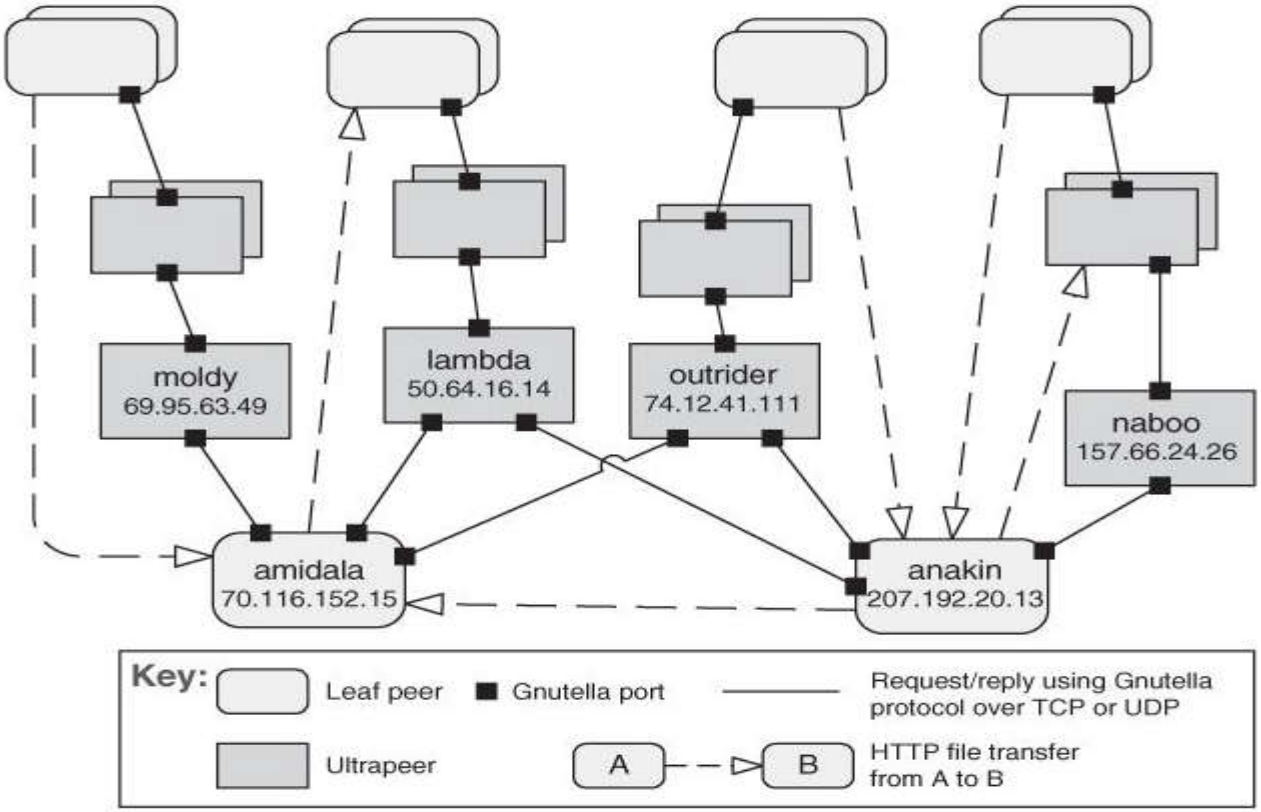
Peer-to-Peer

1
- Los elementos del estilo peer-to-peer son similares a los del cliente-servidor.
- Sin embargo el estilo cliente-servidor impone los roles de cliente y de servidor, mientras que en el peer-to-peer cada conector tiene el mismo rol

2
- Peer-to-Peer es igualitario
- Cliente Servidor es jerárquico

3
- En contraste con el estilo Cliente-Servidor, peer-to-peer no tiene un punto único de fallo

Peer-to-Peer



Bass, Len; Clements, Paul; Kazman, Rick; Software Architecture in Practice

Estilos y Patrones Arquitectónicos

Ventajas

- Facilidad de escalabilidad debido a que los nodos pueden ser agregados o eliminados de la red sin crear un impacto mayor en el sistema
- El solapamiento en las capacidades provee redundancia en caso de uno de los elementos pares deje de estar disponible, los otros elementos pueden completar la tarea
- Normalmente genera mejoras en el rendimiento debido a que las responsabilidades que necesitan esa capacidad e infraestructura son distribuidas.

Desventajas

- El manejo de las tareas de seguridad, consistencia, respaldo y recuperación es complejo
- Sistemas pequeños peer-to-peer comúnmente presentan problemas para lograr las meta de calidad y rendimiento

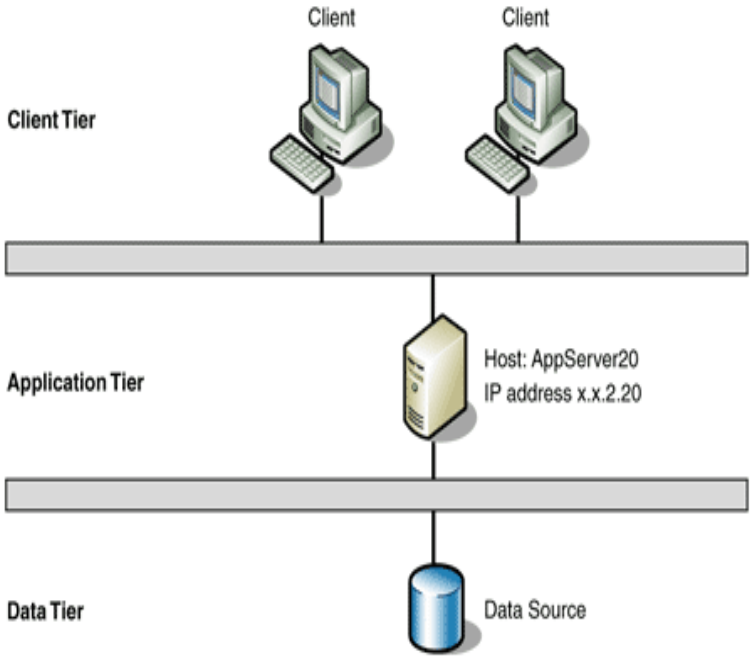
Arquitecturas de Alta disponibilidad

"SOA includes practices and processes that are based on the fact that networks of distributed systems are not controlled by single owners"

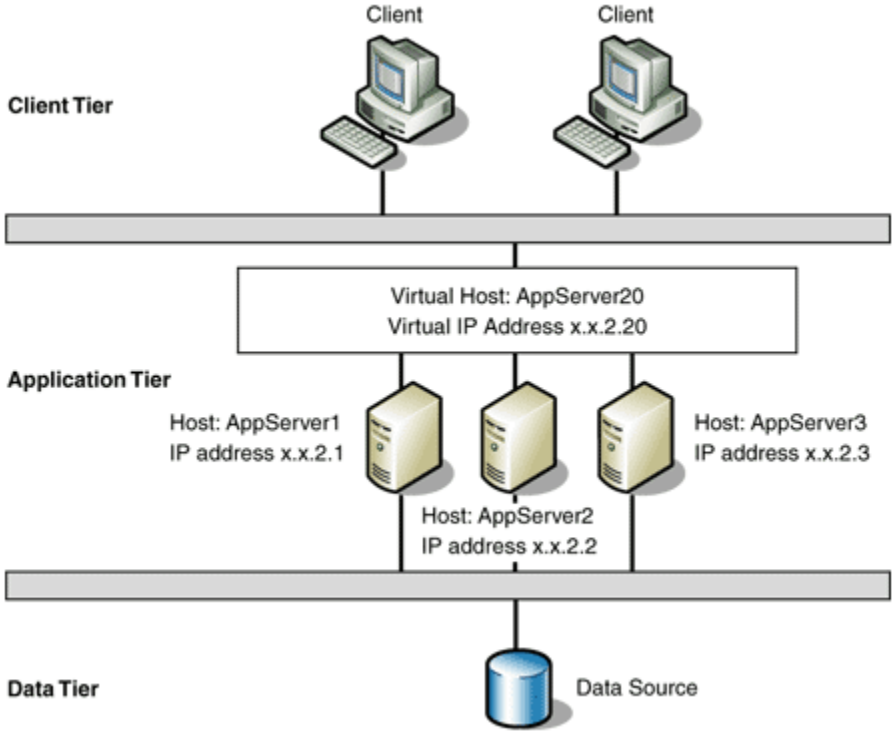
Josuttis, Nicolai M. (2009-02-09). SOA in Practice

Arquitecturas de Alta disponibilidad

Solución sin balanceo

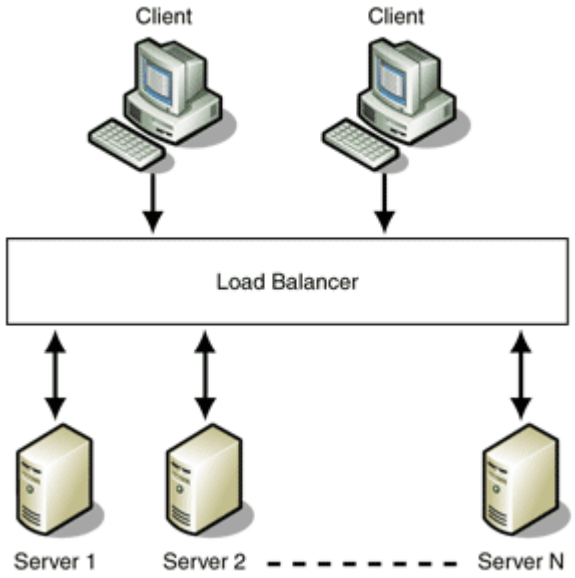


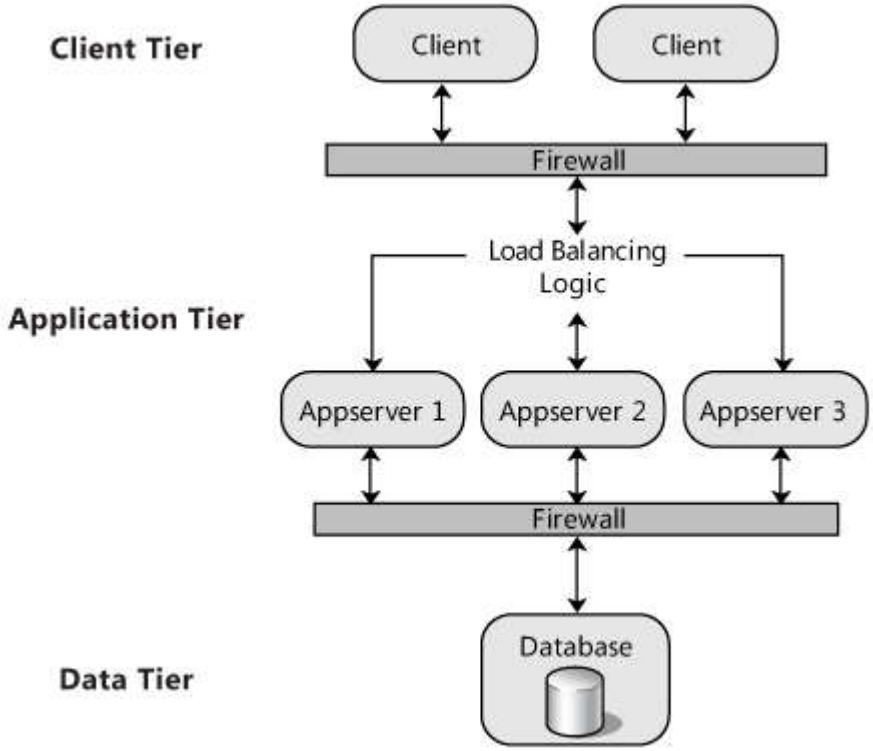
Solución con balanceo



Estilos y Patrones Arquitectónicos

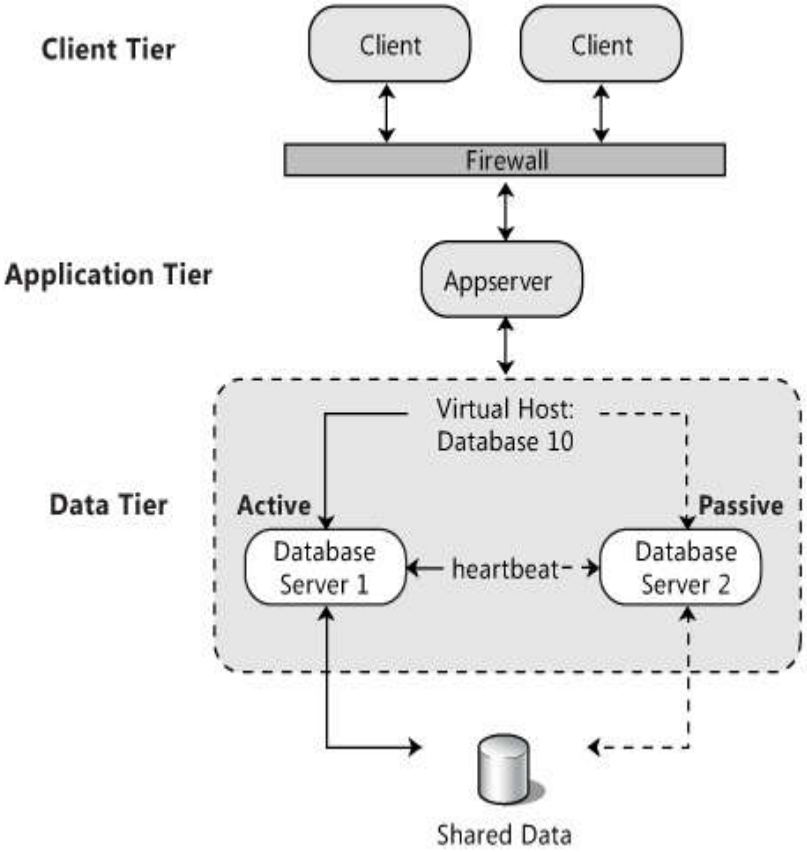
Arquitecturas de Alta disponibilidad

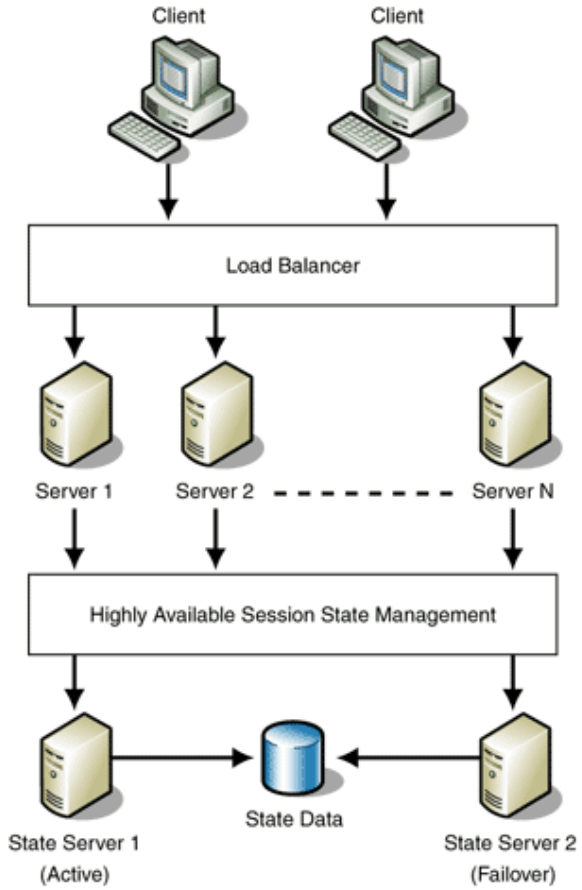




Estilos y Patrones Arquitectónicos

Arquitecturas de Alta disponibilidad





"A software architecture is never right or wrong, but at most better suited for certain situations. It involves making a large number of trade-offs between concerns of different stakeholders. There may be different *acceptable* solutions, and the solution eventually chosen depends on how the balancing between stakeholder concerns is made"