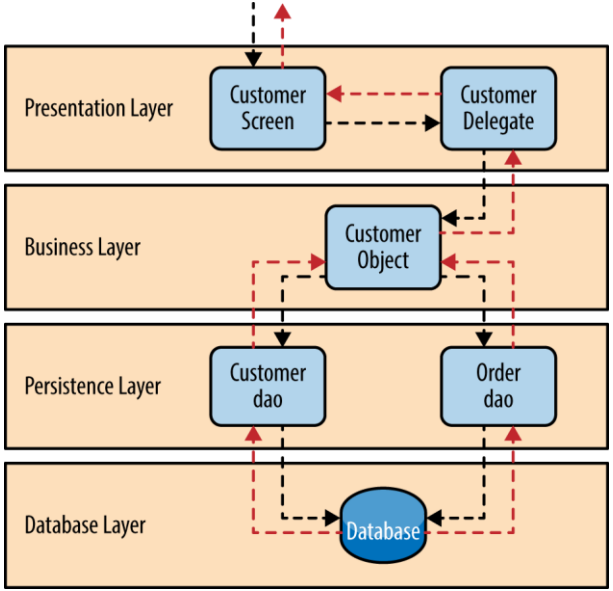
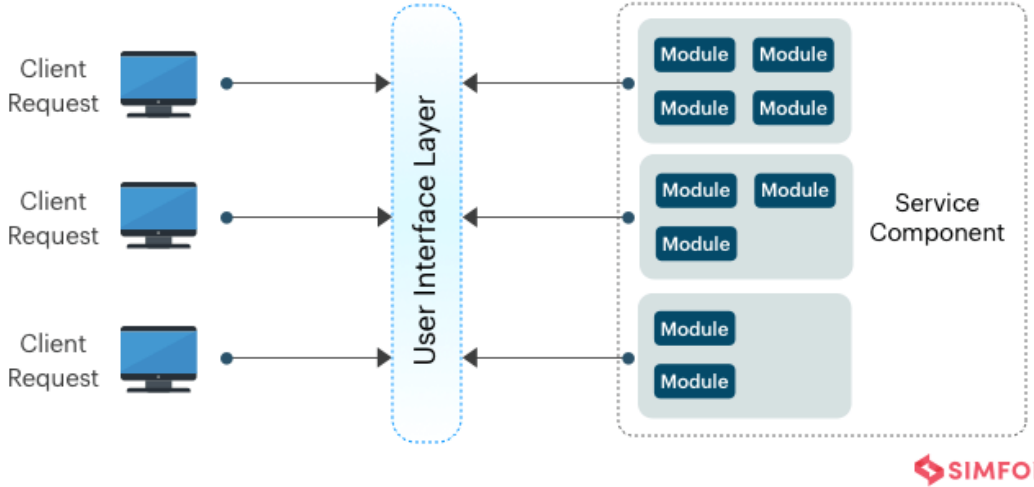
Patrones de Arquitectura de Software



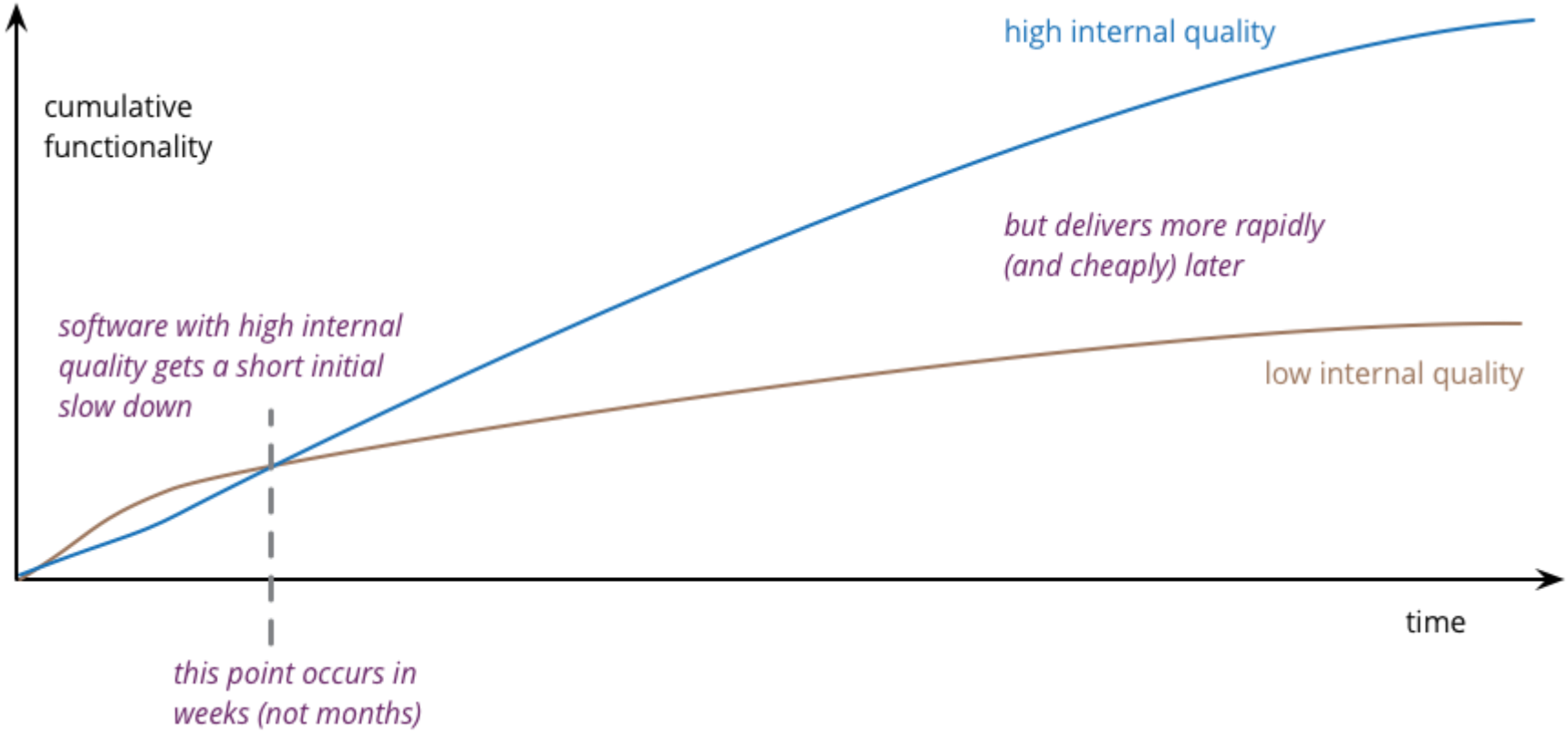


Introducción

El diseño de la arquitectura de un sistema permite observar la organización y el diseño de la estructura general del software.

- Es el puente entre el diseño y los requerimientos.
- Identifica los componentes estructurales de un sistema y las relaciones entre ellos.
- Se procura diseñar la arquitectura de una manera general en la etapa más temprana del proceso de desarrollo.
- Criticidad del proceso: los cambios en la arquitectura (refactorizaciones, etc) son muy costosos.

Costos en el software: Baja Calidad vs Alta Calidad

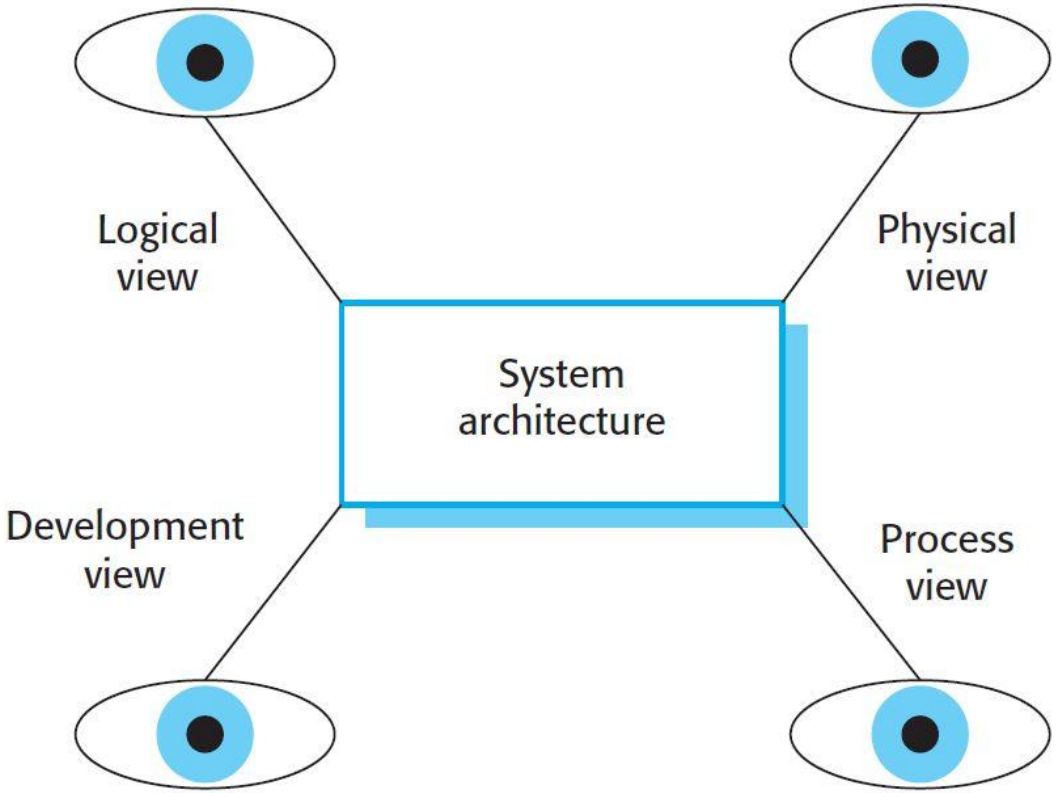


La elección de una arquitectura depende de los Req. NFunc

- Rendimiento: realizar operaciones críticas en un número reducido de componentes, para disminuir el número de interacciones entre ellos.
- Seguridad (security): favorecer una estructura por capas, protegiendo activos críticos en las capas interiores y colocando capas de validación para llegar a ellos.
- Protección (safety): las operaciones críticas deberían estar en uno o pocos componentes para disminuir costos de validación y facilitar el apagado del sistema en caso de un fallo.
- Disponibilidad: se debería incluir componentes redundantes para actualizar o reemplazarlos sin detener el funcionamiento del sistema.
- Mantenibilidad: componentes pequeños y autocontenidos.

Existen conflictos potenciales que deben ser resueltos en base a los RNF principales.

Vistas de Arquitectura



Vistas de Arquitectura (continuación)

Representan distintos puntos de vista del mismo sistema, son útiles para comprender el funcionamiento en distintos niveles y facilitan la documentación.

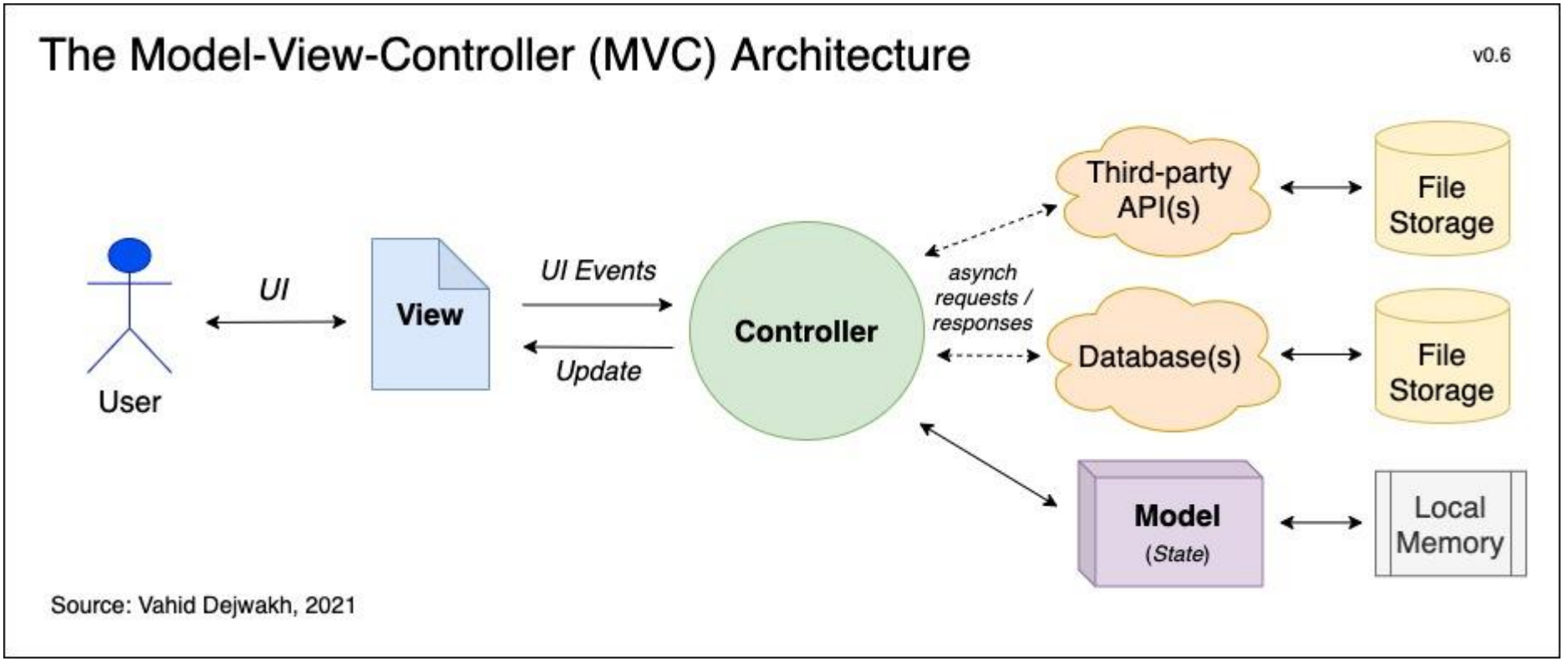
- Vista lógica: permite relacionar requerimientos del sistema con entidades. Muestra abstracciones como objetos y clases de objetos
- Vista de procesos: muestra en “tiempo real” las interacciones de los procesos del sistema.
- Vista de desarrollo: muestra el despiece del sistema en componentes, es el utilizado en el desarrollo del software.
- Vista física: muestra el hardware del sistema y como el software se encuentra distribuido en él.

Patrones de Arquitectura: Model View Controller

Se utiliza para separar la lógica de presentación de la lógica de negocio. El **modelo** representa los datos y la lógica empresarial, la **vista** representa la lógica de presentación y el **controlador** administra la interacción entre el modelo y la vista.

- Se utiliza cuando hay múltiples formas de ver e interactuar con los datos o cuando estos pueden cambiar en el futuro.
- Ventajas: permite procesar los datos independientemente de la lógica de la vista y viceversa.
- Desventajas: incrementa la complejidad del código si el modelo de datos y la interacción son simples.

Model View Controller (continuación)

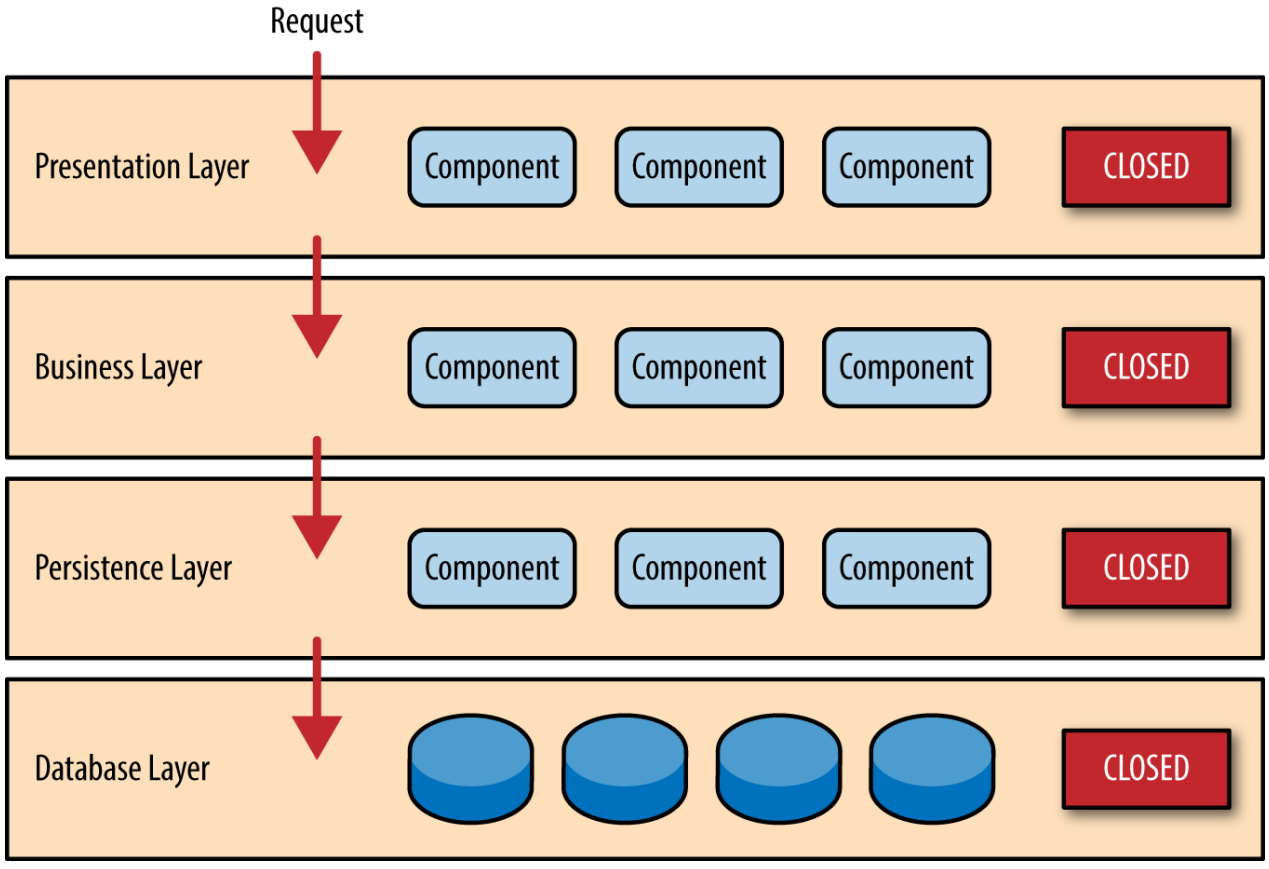


Patrones de Arquitectura: Arquitectura de Capas

Organiza al sistema dividiéndolo en diferentes capas, cada una de las cuales es responsable de una tarea específica. Este patrón ayuda a mejorar la escalabilidad, la mantenibilidad y la flexibilidad del sistema.

- Se utiliza cuando se construyen nuevas funcionalidades arriba de sistemas existentes, cuando el desarrollo está distribuido entre muchos equipos o cuando es necesario un requerimiento de seguridad multinivel.
- Ventajas: permite el reemplazo de capas enteras mientras la interfaz se mantenga. Se pueden crear funciones redundantes (por ej autenticación) en cada capa para aumentar la confianza en el sistema.
- Desventajas: dificultad en la separación entre las capas. Puede disminuir el rendimiento al aumentar el nivel de procesamiento requerido.

Arquitectura de Capas (continuación)

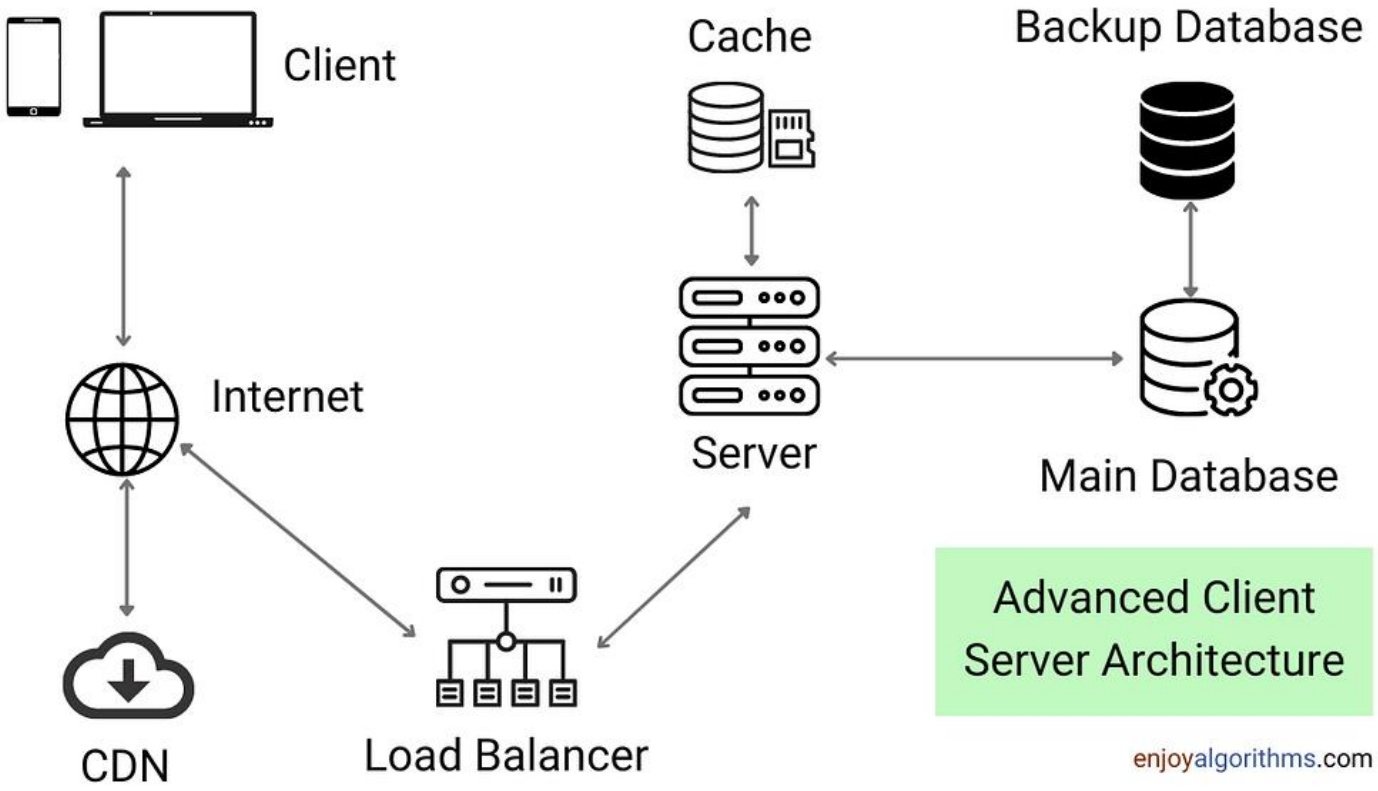


Patrones de Arquitectura: Cliente Servidor

El sistema es presentado como un conjunto de servicios, donde cada uno de estos es mantenido por un servidor separado. Los clientes son usuarios de los servicios y acceden a los servidores para utilizarlos.

- Se usan cuando la información está en bases de datos compartidas tiene que ser accedida desde una amplio rango de localizaciones. También se utiliza para administrar la carga del sistema, mediante la replicación de servidores.
- Ventajas: los servidores pueden estar distribuidos en una red y se pueden implementar funcionalidades generales para todos los clientes.
- Desventajas: cada servicio es un potencial punto de fallo individual y por lo tanto susceptible a ataques DoS o DDoS. Por otro lado, el rendimiento depende de la calidad de conexión de la red y la administración puede ser compleja si hay más de una organización para los servidores.

Cliente Servidor (continuación)

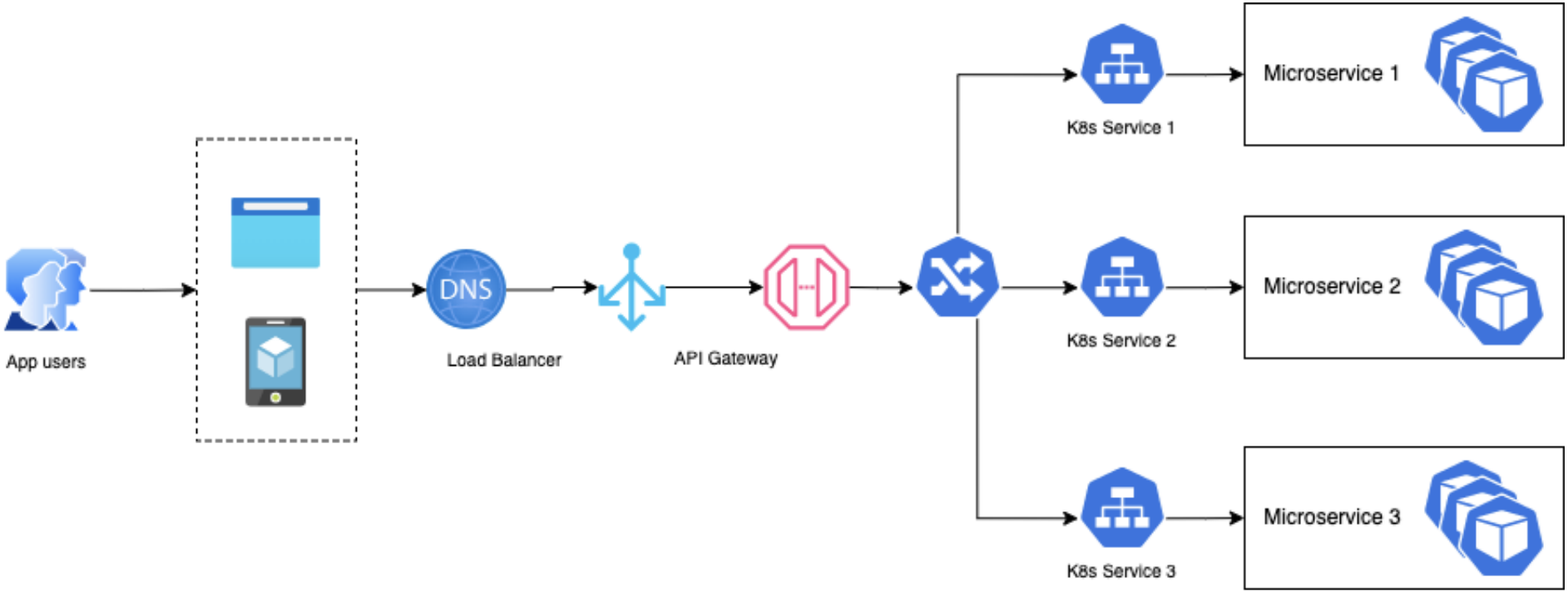


Patrones de Arquitectura: Microservicios

En este patrón, el sistema se divide en servicios más pequeños e independientes que se comunican entre sí a través de protocolos ligeros (HTTP, HTTPS, etc). Cada servicio se centra en una lógica de negocio específica.

- Usos:
  - Grandes sistemas con muchos componentes interdependientes.
  - Sistemas de alto tráfico.
  - Sistemas de evolución rápida.
- Ventajas: escalabilidad, flexibilidad, resiliencia, facilidad de mantenimiento y posibilidad de utilizar un stack tecnológico diverso.
- Desventajas: pueden presentar altos niveles de complejidad (cada uno tiene su deploy, administración, monitoreo, etc), altos costos de comunicación, desafíos de los sistemas distribuidos y mayor costo que un sistema monolítico.

Microservicios (continuación)



Patrones de Arquitectura: Pipe & Filter

Cuando un sistema organiza el procesamiento de datos en un componente (filtro) discreto que solo lleva a cabo un tipo de transformación de datos. Los datos fluyen (pipe / tubería) de un componente a otro en el proceso de procesamiento.

- Usos: sistemas operativos (también en arquitectura de hardware) y aplicaciones de procesamiento segmentadas.
- Ventajas: fáciles de entender y reusar. Se puede implementar en sistemas concurrentes o lineales
- Desventajas: es necesario que el formato de transferencia de datos (inputs y outputs) esté acordado estrictamente. Diseño específico para cada tipo de datos

Pipe & Filter (continuación)