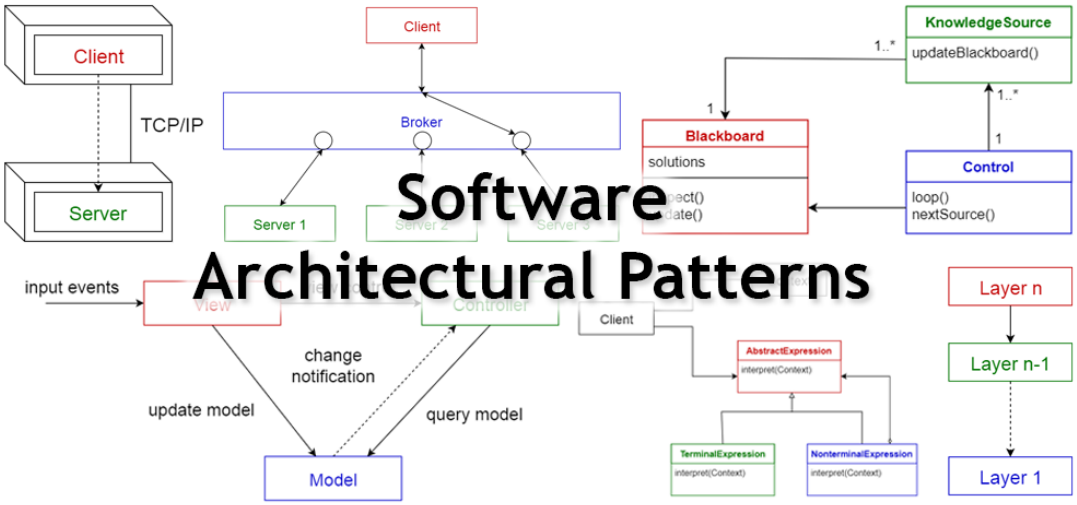


¿Alguna vez se preguntó cómo se diseñan los grandes sistemas empresariales? Antes de que comience un importante desarrollo de software, debemos elegir una arquitectura adecuada que nos proporcione la funcionalidad deseada y los atributos de calidad. Por lo tanto, debemos entender diferentes arquitecturas, antes de aplicarlas a nuestro diseño.

¿Qué es un patrón arquitectónico?

Un patrón arquitectónico es una solución general y reutilizable a un problema común en la arquitectura de software dentro de un contexto dado. Los patrones arquitectónicos son similares al patrón de diseño de software, pero tienen un alcance más amplio.

En este artículo, explicaré brevemente los siguientes 10 patrones arquitectónicos comunes con su uso, pros y contras.

1. Patrón de capas
2. Patrón cliente-servidor
3. Patrón maestro-esclavo
4. Patrón de filtro de tubería
5. Patrón de intermediario
6. Patrón de igual a igual
7. Patrón de bus de evento
8. Modelo-vista-controlador
9. Patrón de pizarra
10. Patrón de intérprete

1. Patrón de capas

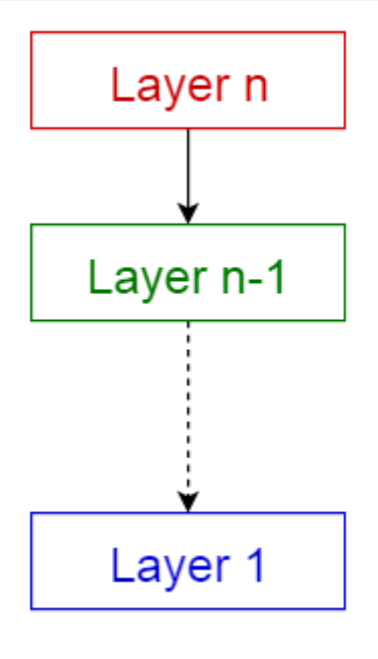
Este patrón se puede utilizar para estructurar programas que se pueden descomponer en grupos de subtareas, cada una de las cuales se encuentra en un nivel particular de abstracción. Cada capa proporciona servicios a la siguiente capa superior.

Las 4 capas más comúnmente encontradas de un sistema de información general son las siguientes.

- Capa de presentación (también conocida como capa UI)
- Capa de aplicación (también conocida como capa de servicio)
- Capa de lógica de negocios (también conocida como capa de dominio)
- Capa de acceso a datos (también conocida como capa de persistencia)

Uso

- Aplicaciones de escritorio generales.
- Aplicaciones web de comercio electrónico.



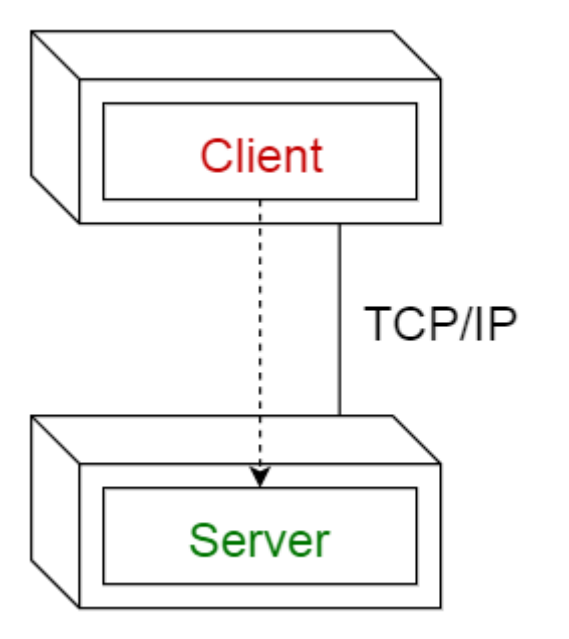
Patrón de capas

2. Patrón cliente-servidor

Este patrón consiste en dos partes; un servidor y múltiples clientes. El componente del servidor proporcionará servicios a múltiples componentes del cliente. Los clientes solicitan servicios del servidor y el servidor proporciona servicios relevantes a esos clientes. Además, el servidor sigue escuchando las solicitudes de los clientes.

Uso

- Aplicaciones en línea como correo electrónico, uso compartido de documentos y banca.



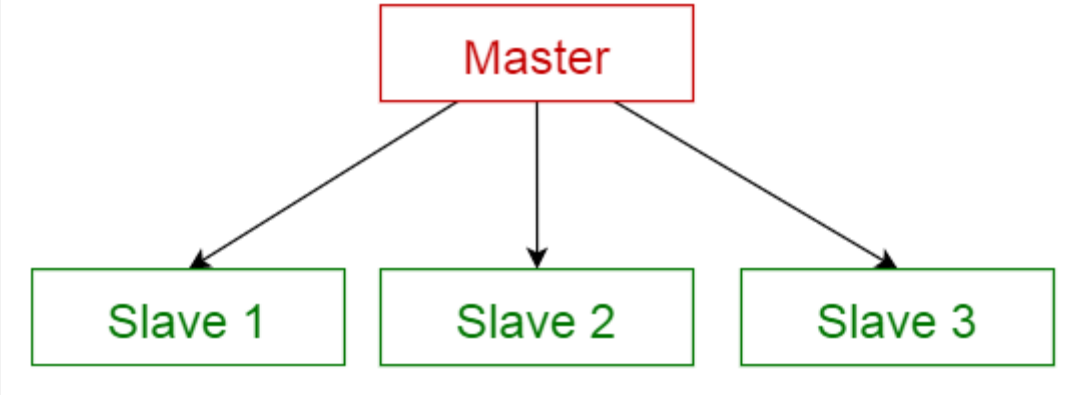
Patrón cliente-servidor

3. Patrón maestro-esclavo

Este patrón consiste en dos partes; maestro y esclavos. El componente maestro distribuye el trabajo entre componentes esclavos idénticos y calcula el resultado final de los resultados que devuelven los esclavos.

Uso

- En la replicación de la base de datos, la base de datos maestra se considera como la fuente autorizada y las bases de datos esclavas se sincronizan con ella.
- Periféricos conectados a un bus en un sistema informático (unidades maestra y esclava).



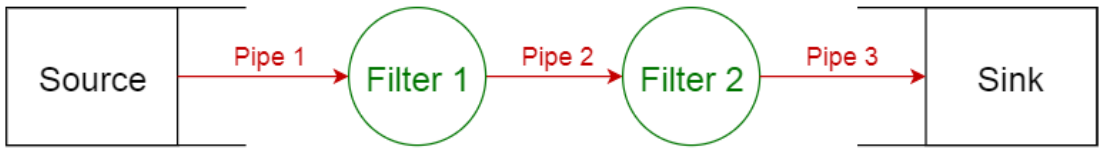
Patrón maestro-esclavo

4. Patrón de filtro de tubería

Este patrón se puede usar para estructurar sistemas que producen y procesan una secuencia de datos. Cada paso de procesamiento se incluye dentro de un componente de filtro. Los datos que se procesarán se pasan a través de las tuberías. Estas tuberías se pueden utilizar para el almacenamiento en búfer o con fines de sincronización.

Uso

- Compiladores Los filtros consecutivos realizan análisis léxico, análisis sintáctico y generación de código.
- Flujos de trabajo en bioinformática.



Patrón de filtro de tubería

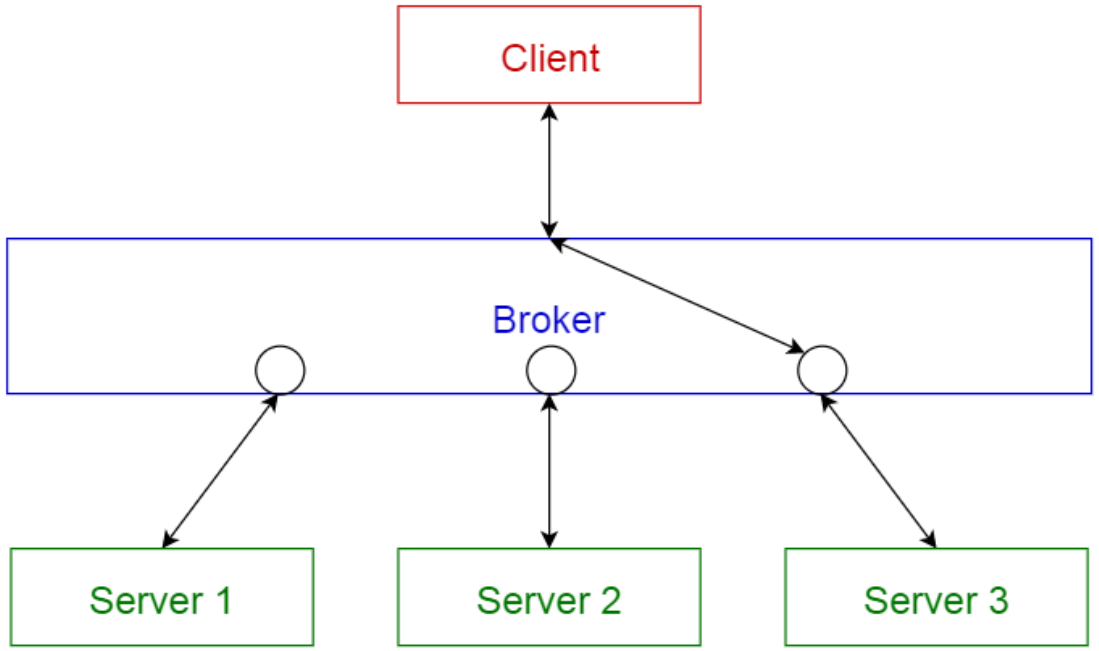
5. Patrón del agente

Este patrón se usa para estructurar sistemas distribuidos con componentes desacoplados. Estos componentes pueden interactuar entre sí mediante invocaciones de servicios remotos. Un componente de intermediario es responsable de la coordinación de la comunicación entre los componentes.

Los servidores publican sus capacidades (servicios y características) a un intermediario. Los clientes solicitan un servicio del intermediario y el intermediario redirecciona al cliente a un servicio adecuado desde su registro.

Uso

- Software de Message Broker como Apache ActiveMQ, Apache Kafka , RabbitMQ y JBoss Messaging .



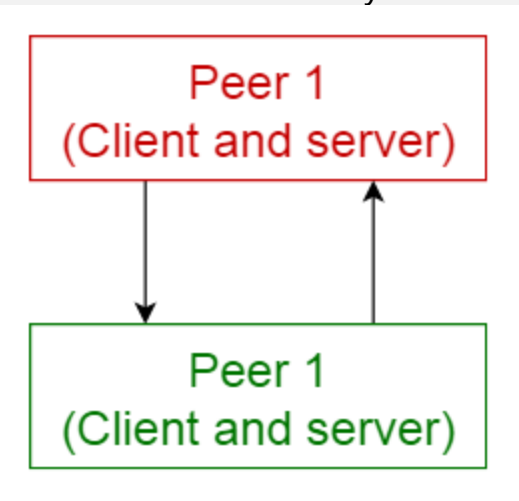
Patrón de intermediario

6. Patrón de igual a igual

En este patrón, los componentes individuales se conocen como pares. Los pares pueden funcionar tanto como un cliente, solicitando servicios de otros pares, y como un servidor, proporcionando servicios a otros pares. Un par puede actuar como un cliente o como un servidor o como ambos, y puede cambiar su rol dinámicamente con el tiempo.

Uso

- Redes de intercambio de archivos como Gnutella y G2)
- Protocolos multimedia como P2PTV y PDTP .



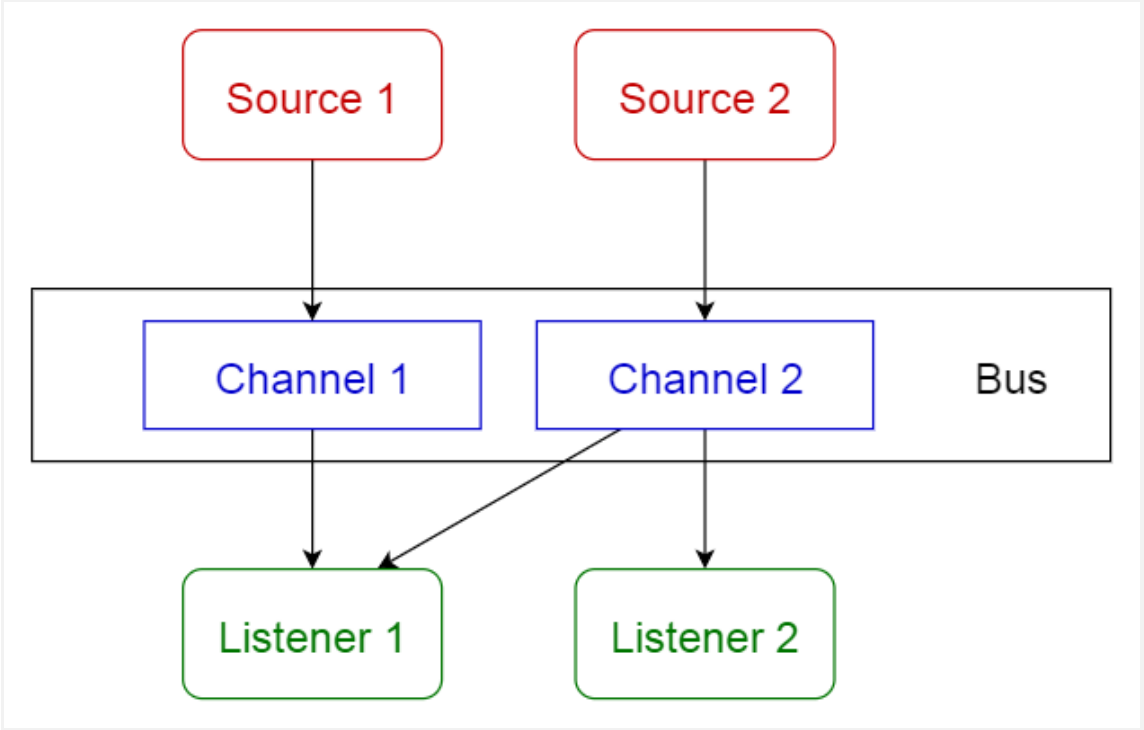
Patrón de igual a igual

7. Patrón de bus de evento

Este patrón trata principalmente con eventos y tiene 4 componentes principales; fuente de evento, escucha de evento, canal y bus de evento. Las fuentes publican mensajes en canales particulares en un bus de eventos. Los oyentes se suscriben a canales particulares. Los oyentes son notificados de los mensajes que se publican en un canal al que se han suscrito anteriormente.

Uso

- Desarrollo de Android
- Servicios de notificación



Patrón de bus de evento

8. Patrón de modelo-vista-controlador

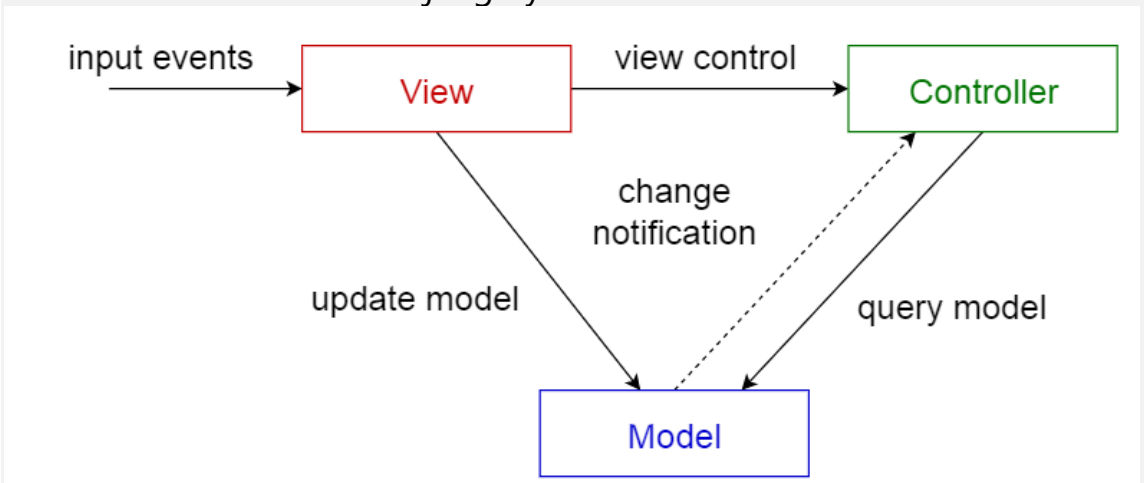
Este patrón, también conocido como patrón MVC, divide una aplicación interactiva en 3 partes, como

1. modelo: contiene la funcionalidad y los datos básicos
2. vista: muestra la información al usuario (se puede definir más de una vista)
3. controlador: maneja la entrada del usuario

Esto se hace para separar las representaciones internas de información de las formas en que se presenta y acepta la información del usuario. Desacopla los componentes y permite la reutilización eficiente del código.

Uso

- Arquitectura para aplicaciones World Wide Web en los principales lenguajes de programación.
- Marcos web como Django y Rails .



Modelo-vista-controlador

9. Patrón de pizarra

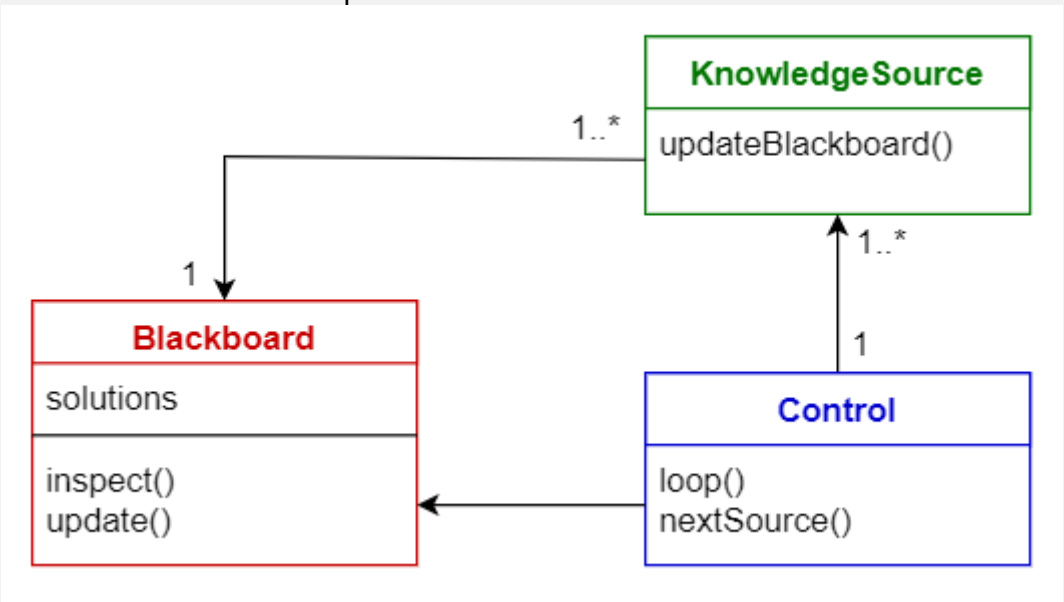
Este patrón es útil para problemas para los que no se conocen estrategias de solución deterministas. El patrón de pizarra consta de 3 componentes principales.

- pizarra: una memoria global estructurada que contiene objetos del espacio de solución
- fuente de conocimiento: módulos especializados con su propia representación
- componente de control: selecciona, configura y ejecuta módulos.

Todos los componentes tienen acceso a la pizarra. Los componentes pueden producir nuevos objetos de datos que se agregan a la pizarra. Los componentes buscan tipos particulares de datos en la pizarra, y pueden encontrarlos por coincidencia de patrones con la fuente de conocimiento existente.

Uso

- Reconocimiento de voz
- Identificación y seguimiento del vehículo
- Identificación de la estructura proteica
- Sonar señala la interpretación.



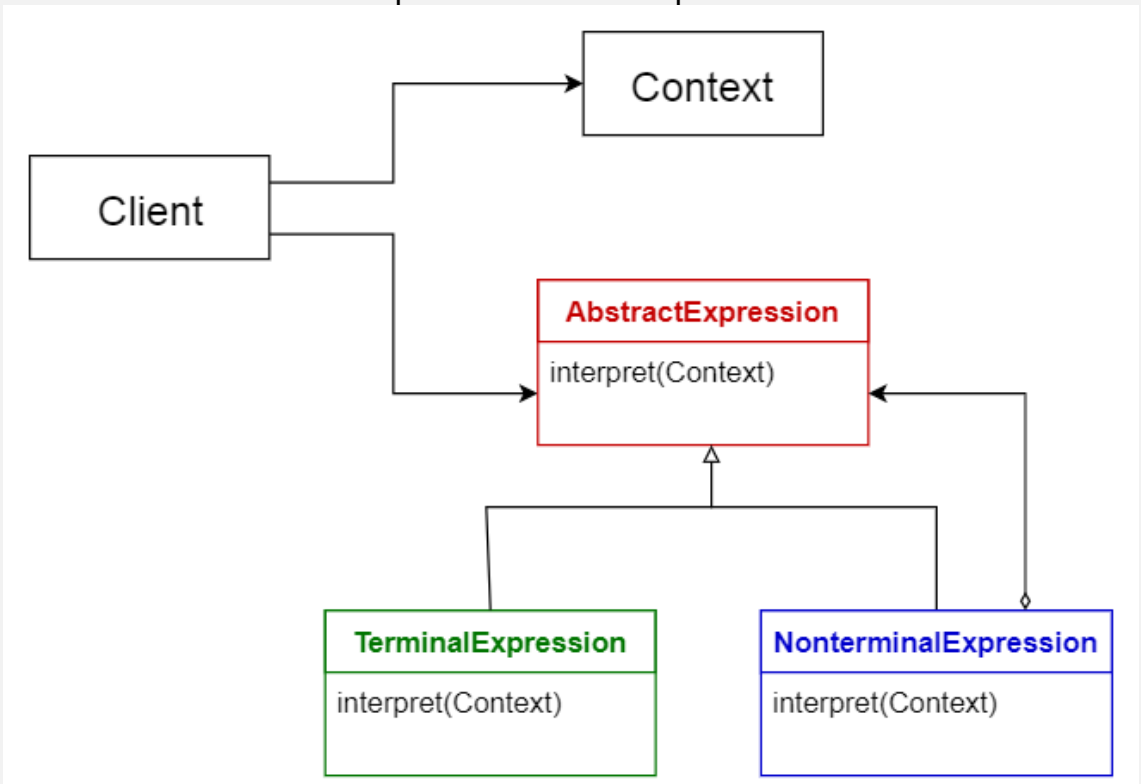
Patrón de pizarra

10. Patrón de intérprete

Este patrón se usa para diseñar un componente que interpreta programas escritos en un lenguaje dedicado. Especifica principalmente cómo evaluar las líneas de programas, conocidas como oraciones o expresiones escritas en un idioma particular. La idea básica es tener una clase para cada símbolo del idioma.

Uso

- Lenguajes de consulta de base de datos como SQL.
- Idiomas utilizados para describir los protocolos de comunicación.



Patrón de intérprete

Autor: YNCatari