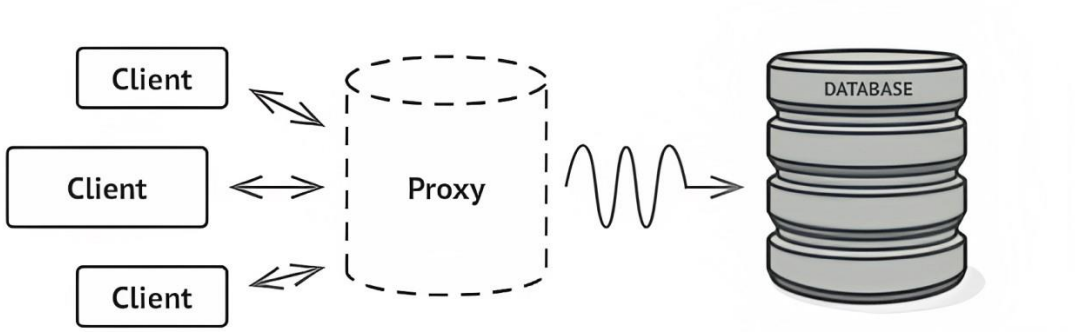
Patrón de diseño
Proxy



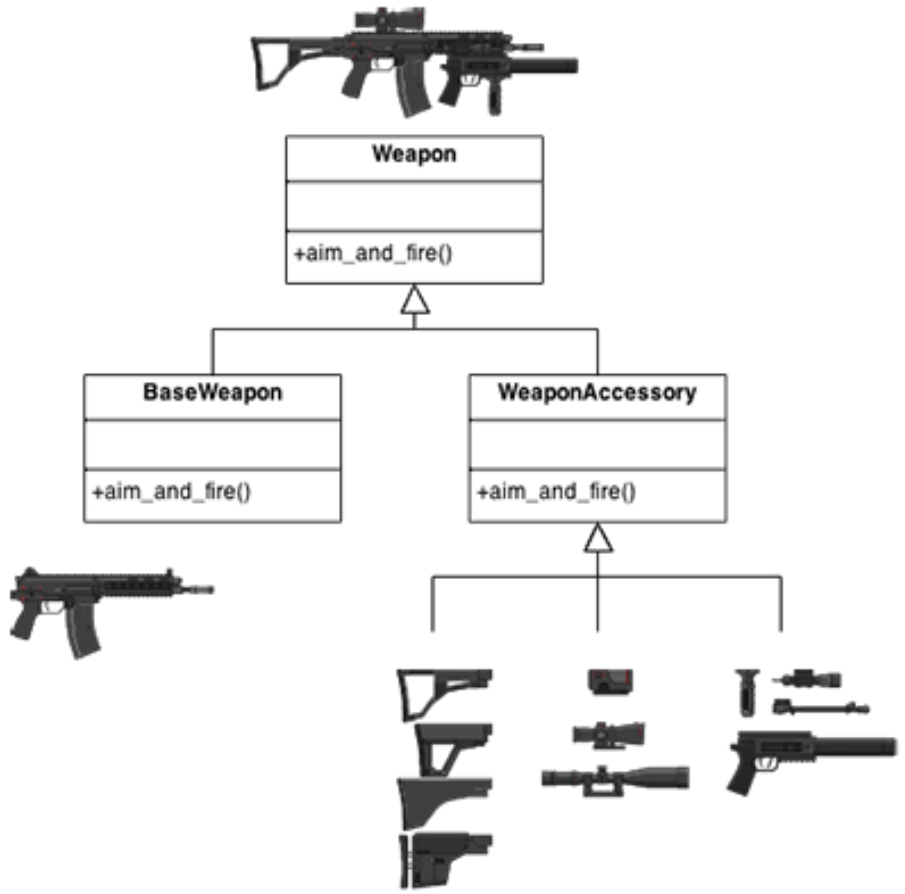
¿Qué es un patrón de diseño?

Solución general y reutilizable aplicable a problemas comunes en el diseño del software.

Acelera el proceso de desarrollo proporcionando paradigmas de desarrollo testeados y probados.

- Patrón creacional
- **Patrón estructural**
- Patrón de comportamiento

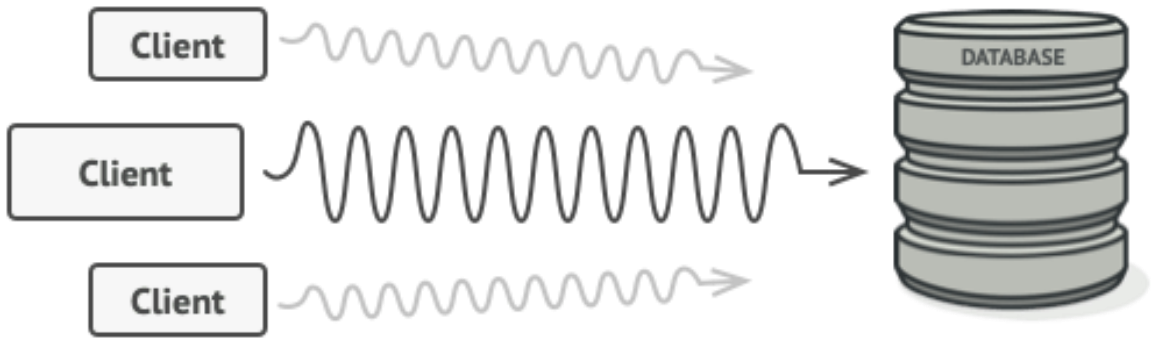
Patrón estructural

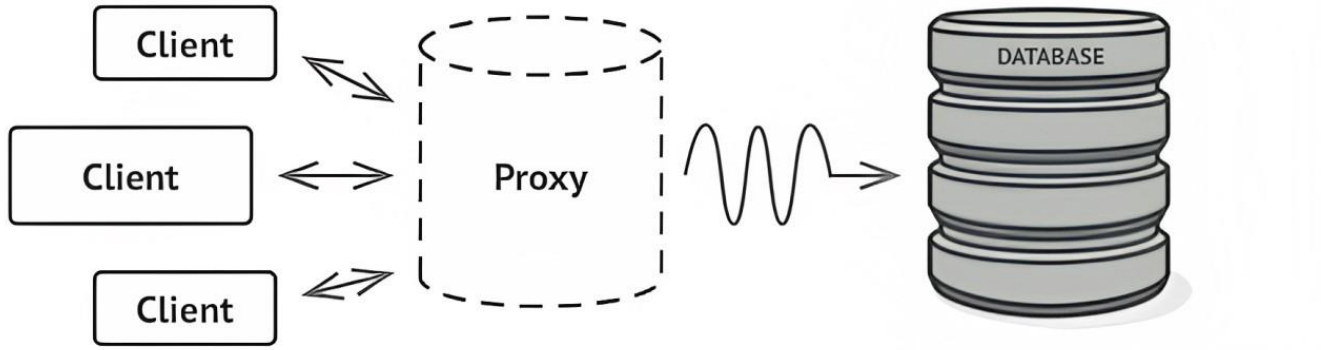


Permite identificar y realizar relaciones entre entidades.

La composición de clases y objetos facilita la creación de estructuras más grandes.

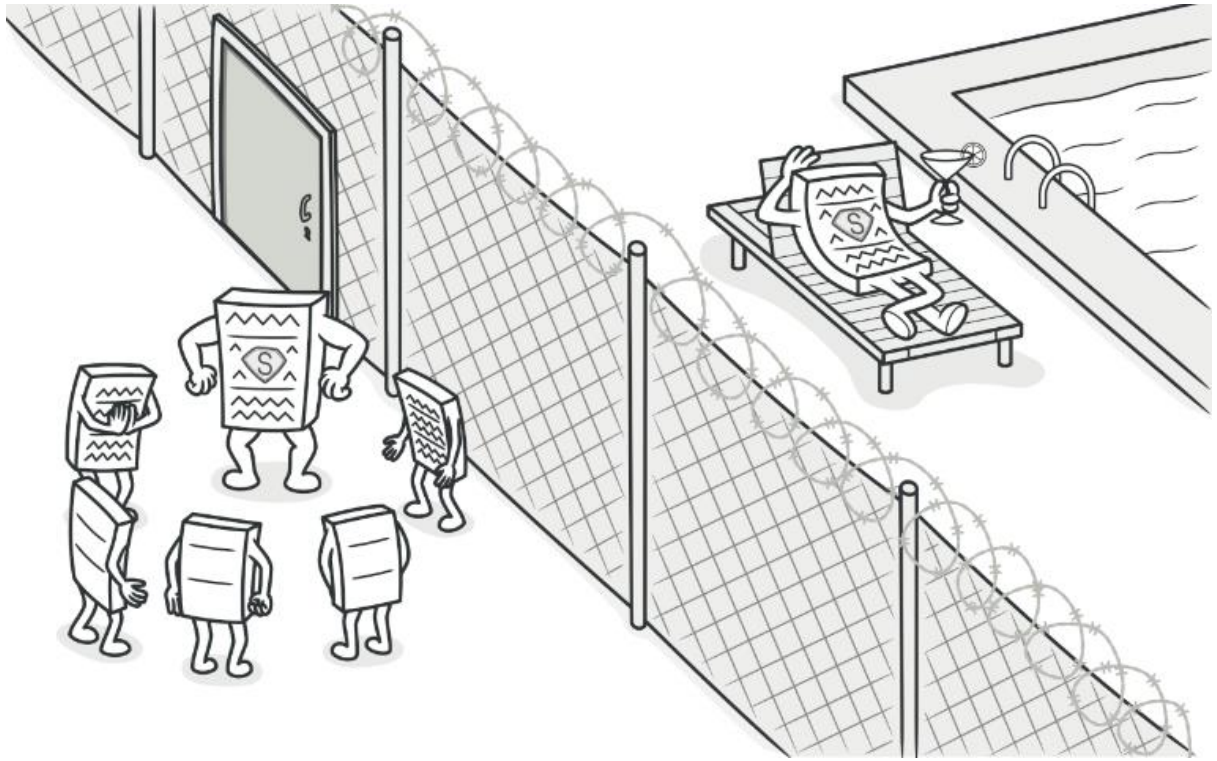
¿Qué es un Proxy?





Patrón de diseño Proxy

- Permite representar un objeto con un objeto sustituto, controlando el acceso a la información.
- Puede dar funcionalidad adicional a un método de una clase sin modificar su estructura.
- Se compone de un Subject, un RealSubject y el Proxy.



Tipos de patrones Proxy

Proxy remoto	Usado para que los recursos de la interfaz estén disponibles a distancia, como un servicio web.
Proxy virtual	Usado para reducir el costo en procesos de alto costo.
Proxy protector	Usado para proveer seguridad y privacidad de los datos.
Proxy inteligente	Usado cuando se requiere una capa adicional de seguridad para un objeto específico.

Ventajas y desventajas del patrón Proxy

Ventajas
• Mejor seguridad. • Mejor desempeño. • Interfaz simplificada. • Comunicación remota. • Diseño modular.

Desventajas
• Introduce un nivel de indirección adicional. • Problemas de sincronización. • Mayor trabajo requerido en el proceso de desarrollo.

Cómo implementar un Proxy

1. Si no hay una interfaz de servicio preexistente, crear una para que los objetos de proxy y de servicio sean intercambiables.

2. Crear la clase proxy.

3. Implementar los métodos del proxy según sus propósitos.

4. Introducir un método de creación que decida si el cliente obtiene un proxy o un servicio real.

5. Implementar la inicialización diferida para el objeto de servicio.