



Patrón de Diseño FACADE

Simplificando Sistemas Complejos

Integrantes

Brayan Acuña

Juan Ayala

Universidad Popular del Cesar
Facultar de Ingeniería y Tecnológica
Materia: Patrones de diseño

Ing. Jairo Seoanes



Ğ Ė Ģ Ġ Ĥ Ō MF M̊M
È Ö Ò Ñ N Õ Ñ Ó M̊M Ĭ Ò Æ Ñ Ö M Æ
F Ö Ö Ó Ñ Ő Ő Æ

El patrón **Facade** (o Fachada) es un patrón estructural que proporciona una interfaz unificada y simplificada a un conjunto de interfaces en un subsistema.

Propósito

Proporcionar una interfaz simple, haciendo que el subsistema sea más fácil de usar al ocultar su complejidad detrás de una fachada.



Imagina que tienes una máquina con muchos botones, cables y palancas. Usarla es confuso si no sabes cómo funciona por dentro. Lo que hace Facade es darte **un solo botón grande y claro** para que puedas usar todo el sistema sin preocuparte por los detalles.



? ¿Por qué se creó?

Antes de que existiera este patrón, cuando un programa quería usar varios componentes (por ejemplo: cargar un archivo, descomprimirlo, decodificarlo, reproducir el video y el audio...), **tenía que conocer cómo funcionaba cada uno y llamarlos uno por uno**, en el orden correcto.

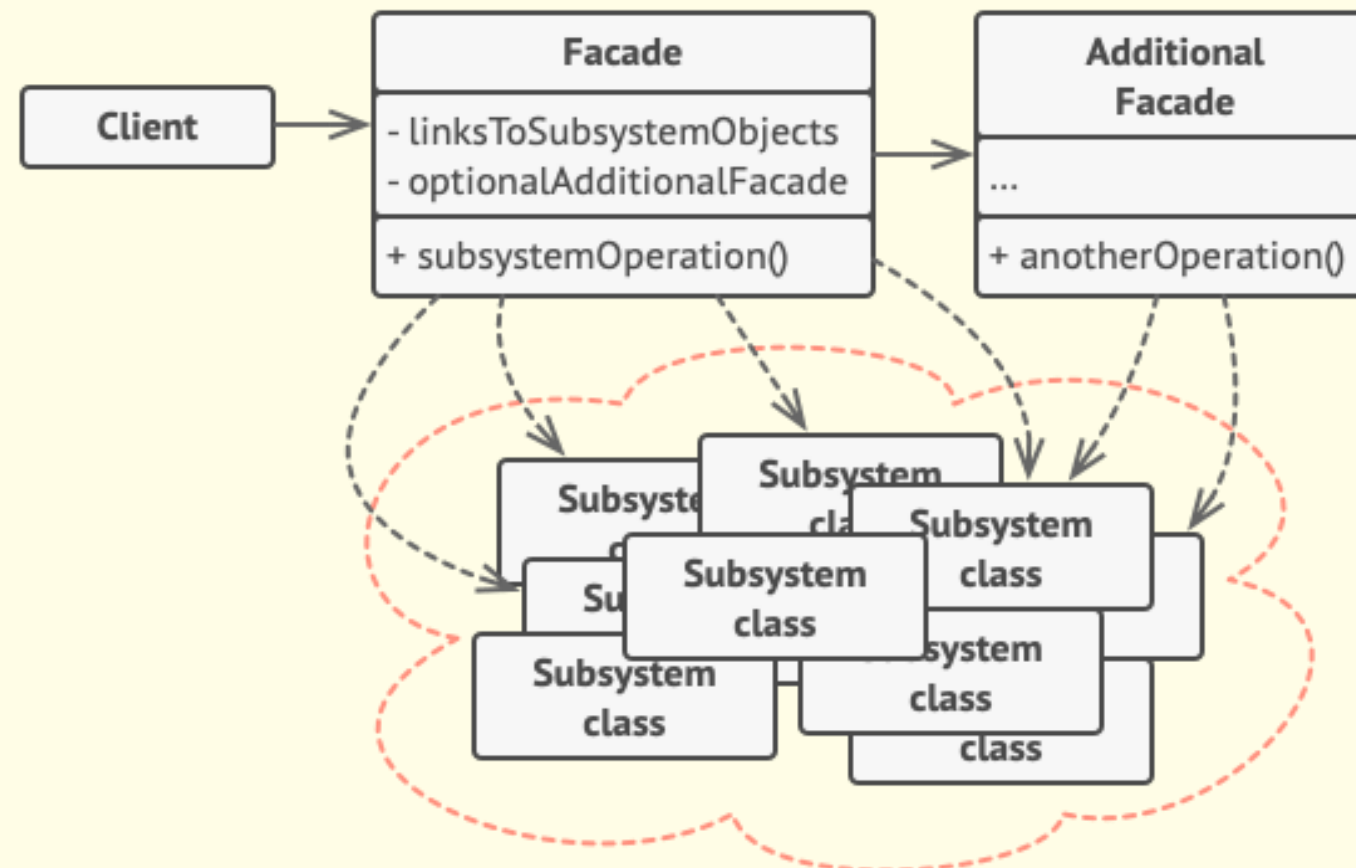
Esto era como tener que usar todos los botones de una nave espacial cada vez que querías ver una película.

→ **Problema:** el código se volvía complicado, difícil de entender, mantener y modificar.

→ **Ejemplo:** si cambiabas la forma de decodificar el audio, tenías que buscar todos los lugares del código donde se usaba ese paso y actualizarlos.

✓ ¿Cuál es la solución que trae Facade?

Facade crea una clase “intermediaria” que se encarga de hablar con todos esos componentes complejos por ti.



Tú solo llamas a la función, y el patrón Facade se encarga de hacer todo lo necesario **por dentro**

¿Cómo se ve esto en la vida real?

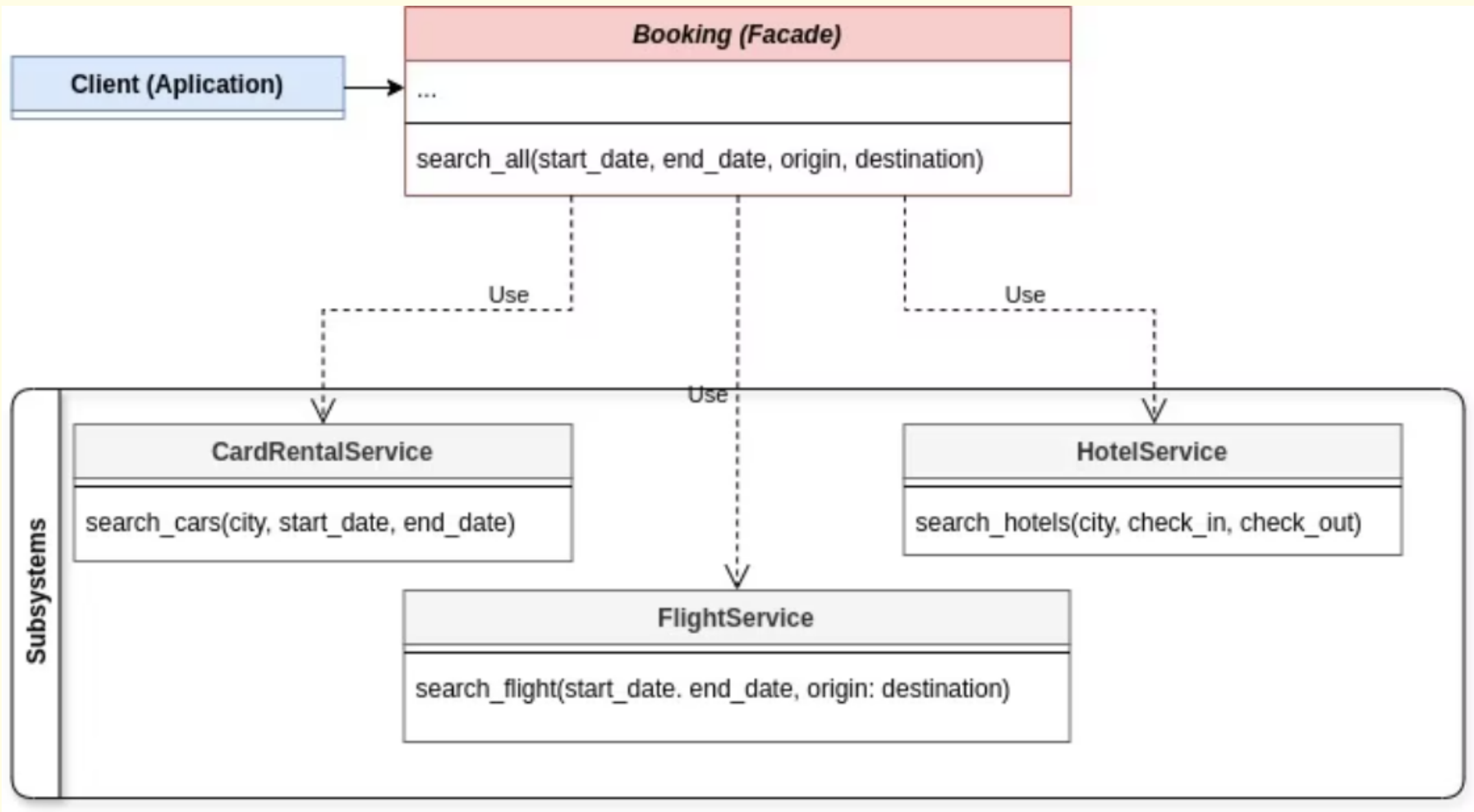
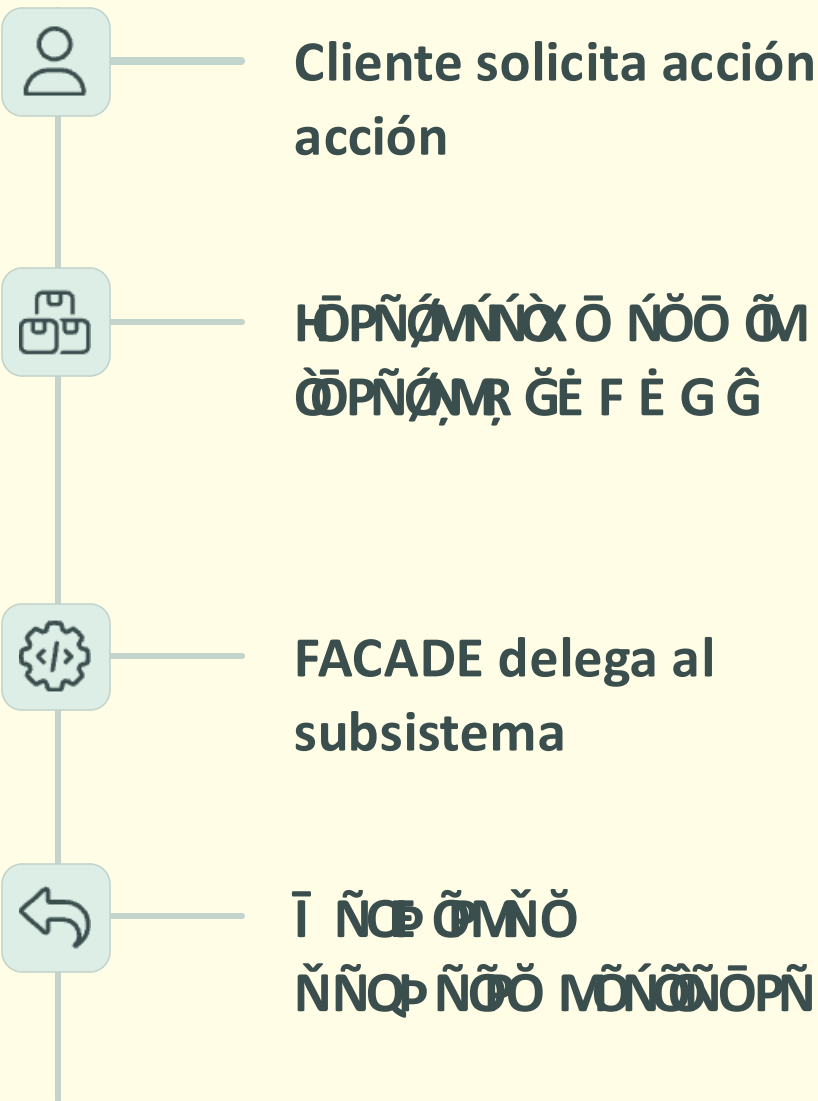
Imagina que usas Netflix.

Tú solo haces clic en “**Reproducir**”, y ya: el sistema carga la película, ajusta la calidad, sincroniza el audio, pone los subtítulos, etc.

Tú no ves todo ese trabajo interno, y no necesitas saber cómo se hace.

Eso es exactamente lo que hace Facade

El FACADE en Acción



Ejemplo: Búsqueda de Gestión de Reservas de Vuelos.

bjxus/Facade-Flights

A React based project for searching and managing flight bookings.

1Contributor

0Issues

0Stars

0Forks


A stylized blue icon representing a facade or building structure.


GitHub

GitHub - bjxus/Facade-Flights: A React based project for searching an...


A React based project for searching and managing flight bookings. - bjxus/Facade-Flights

juadadev/**FACADE-API-Flights**




 1


Contributor

 0


Issues


 0


Stars

 0

Forks



 GitHub



GitHub - juadadev/FACADE-API-Flights

Contribute to juadadev/FACADE-API-Flights development by creating an account on GitHub.

Ventajas



Interfaz Simplificada

Facilita el uso y comprensión del sistema.



Reduce Acoplamiento

Minimiza dependencias entre cliente y subsistema.



Ocultar Complejidad

Encapsula detalles intrincados del subsistema.



Mejora Mantenimiento

Facilita refactorización y mantenimiento del código.

Desventajas



Limitaciones

Puede ocultar funcionalidades necesarias del subsistema.



Facade puede volverse demasiado complejo si no se controla.

FACADE puede volverse demasiado complejo si no se controla.

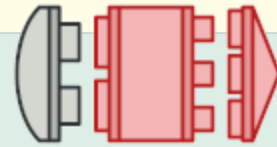


Puede afectar rendimiento por capa adicional.

Puede afectar rendimiento por capa adicional.

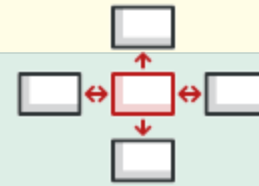
FACADE y sus Amigos

Adapter



Convierte interfaces para clientes.

Mediator



Centraliza interacción entre objetos.

Singleton



Garantiza una única instancia global.

Abstract Factory

Oculto creación de objetos del subsistema.



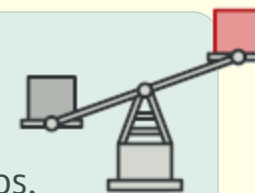
Proxy

Gestiona acceso a entidades complejas.



Flyweight

Muestra cómo crear muchos objetos pequeños, mientras que [Facade](#) muestra cómo crear un solo objeto que representa un subsistema completo.



Resumen y Claves del Patrón **FACADE**

FACADE simplifica sistemas complejos con una interfaz única.

Reduce dependencias, oculta detalles y mejora mantenimiento.

