Laboratorio 3 Diseño de Software

PATRÓN
FACADE

Isabella Rodríguez Sánchez
C26701

ÍNDICE DE CONTENIDOS

DEFINICIÓN

- El patrón Facade fue formalizado por el **"Gang of Four"** (GoF), en su libro de 1994: **Design Patterns: Elements of Reusable Object-Oriented Software.**

- El término "Facade" viene de la arquitectura real: es la cara externa de un edificio. Así como una fachada cubre estructuras internas complejas, el patrón Facade **oculta sistemas complejos detrás de una interfaz sencilla.**

- Es un **patrón estructural**, que significa:
  - No crea nuevos objetos.
  - No define nuevos comportamiento.
  - **Sí reorganiza cómo interactúan múltiples clases complejas.**

PROBLEMA QUE RESUELVE

En sistemas complejos, el cliente (usuario o código consumidor) debe interactuar con muchas clases o módulos, lo que **aumenta el acoplamiento y la complejidad del sistema**.

¿CÓMO?

El **Facade** actúa como una **API simplificada** que agrupa operaciones internas en un conjunto reducido de métodos de alto nivel. Así el cliente interactúa con **un solo punto de entrada**, sin necesidad de conocer los detalles del sistema.

- **Separación de responsabilidades**: el cliente no tiene que coordinar pasos complejos.
- **Abstracción limpia**: el sistema puede cambiar por dentro sin afectar al consumidor.
- **Mejora del principio de inversión de dependencias**: el cliente depende de una abstracción (el Facade), no de múltiples clases concretas.

VENTAJAS

- **Reduce el acoplamiento:**

El cliente solo depende de la fachada, no de múltiples clases internas.

- **Simplifica el uso del sistema:**

Proporciona una API de alto nivel fácil de entender y usar.

- **Facilita el mantenimiento y escalabilidad:**

Los cambios en los subsistemas no afectan al cliente y permite añadir nuevas funcionalidades fácilmente.

- **Mejora la organización modular:**

Separa la lógica de coordinación u organización de los detalles de implementación.

- **Favorece la testabilidad:**

Permite probar la lógica principal desde un solo punto de acceso.

DESVENTAJAS

- **Oculta funcionalidades avanzadas**:

Puede limitar el acceso a operaciones específicas del subsistema.

- **Riesgo de sobrecarga:**

Una fachada mal diseñada puede asumir demasiadas responsabilidades y volverse inflexible.

- **Sensación falsa de simplicidad:**

Puede ocultar la complejidad real del sistema para desarrolladores nuevos.

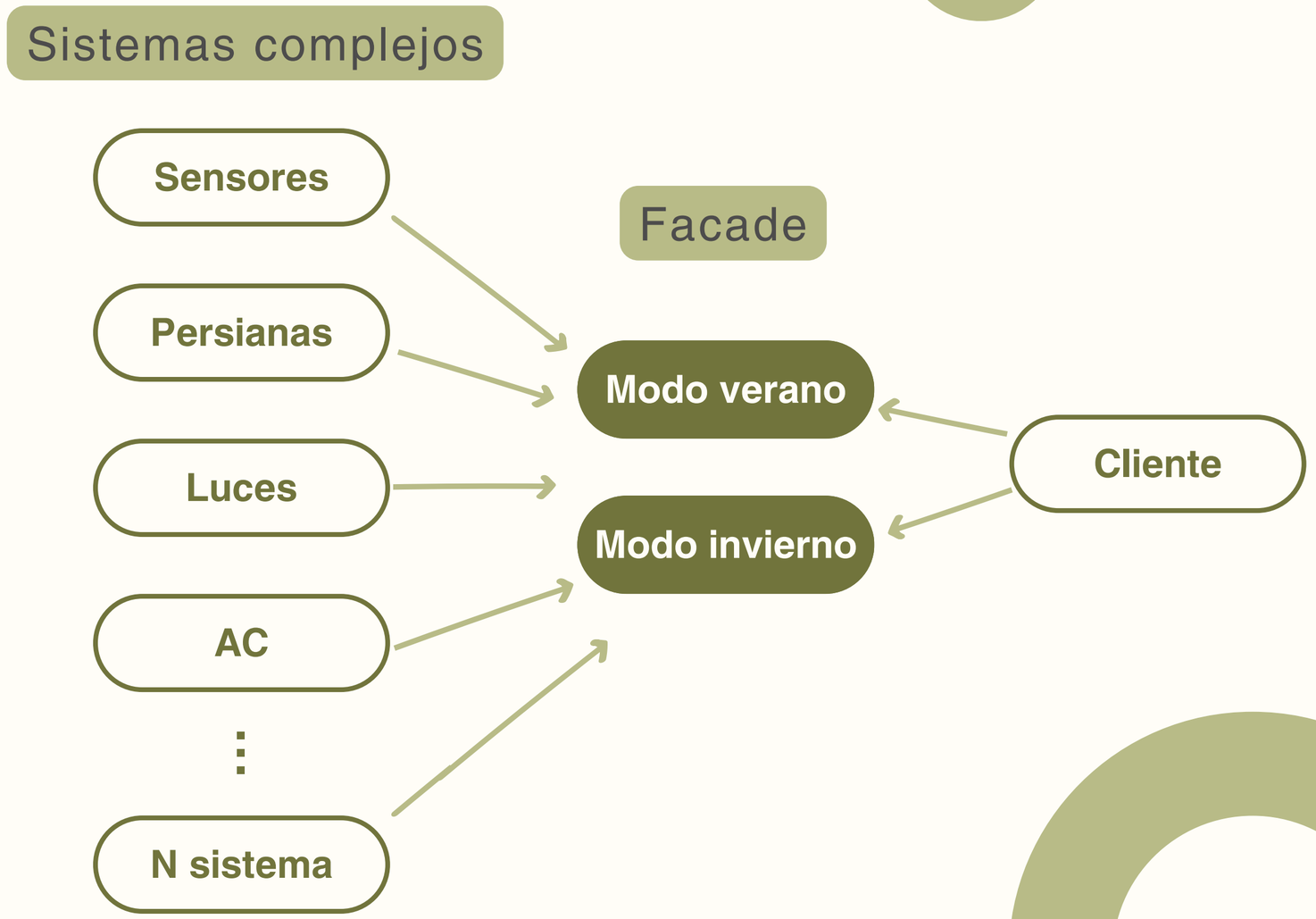
- **Agrega una capa adicional**

En sistemas simples, puede ser innecesario y aumentar la complejidad sin beneficio.

EJEMPLO

SMART HOME

- Sistema de **automatización y el control inteligente de una casa.**
- El usuario quiere modos como **"verano" o "invierno".**
- **Subsistemas:** sensores, persianas, luces, aire acondicionado.
- Se **llama solo al modo** que tiene por debajo todos los subsistemas.



CONCLUSIÓN

El patrón Facade es uno de los **más usados** en la ingeniería de software porque **simplifica la interacción con sistemas complejos**. De hecho, muchos lo usamos a diario sin saberlo, en librerías, frameworks o APIs, lo que vuelve más accesible herramientas importantes, como **Flask** en el proyecto

GRACIAS

¿PREGUNTAS?