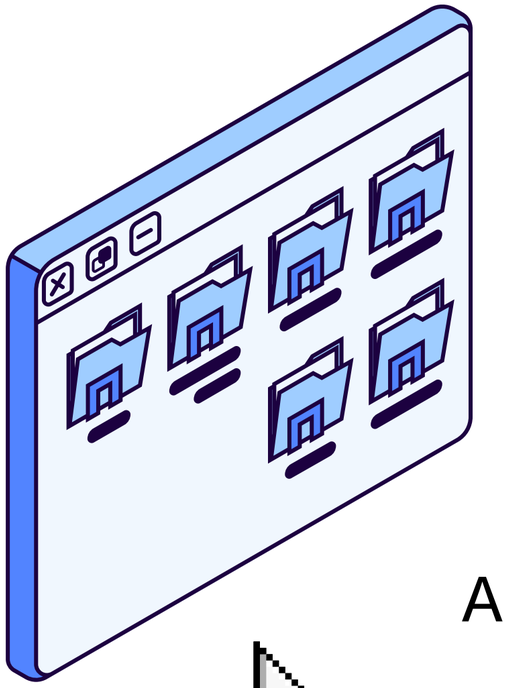
Patrón de diseño

Home Content Contact

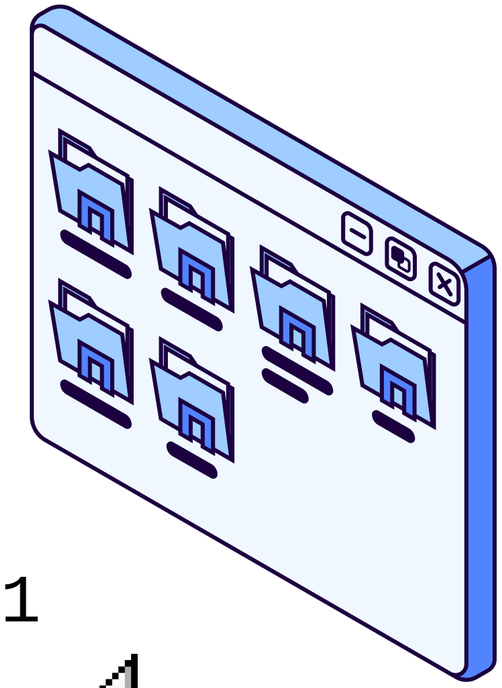


Patrón de Diseño

FACADE

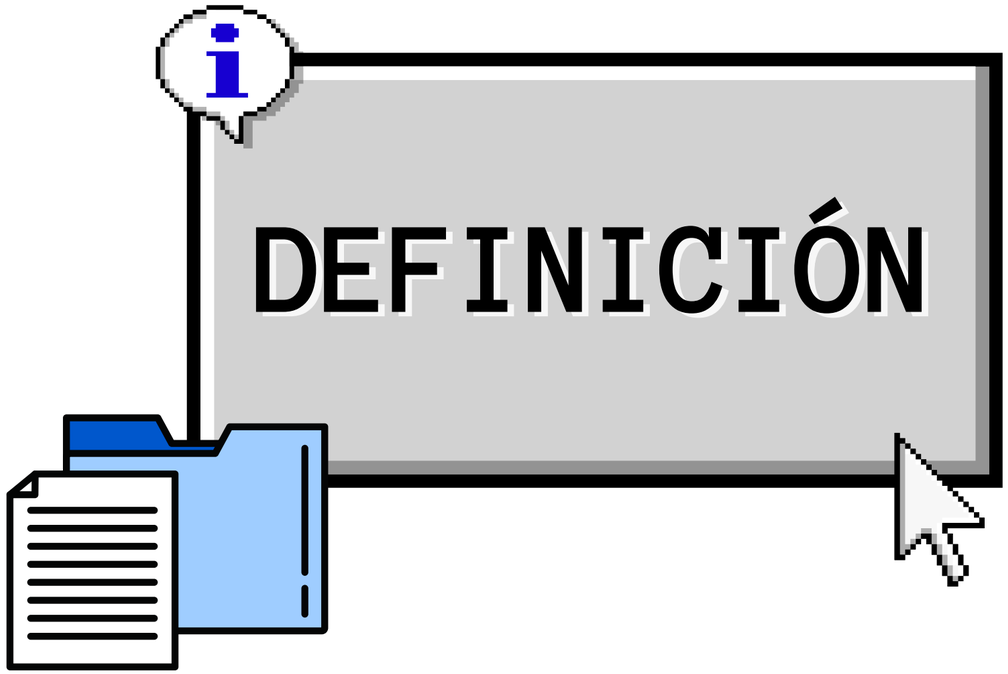
Amber Villarreal Campos C28481

Start



Patrón de diseño/Definición

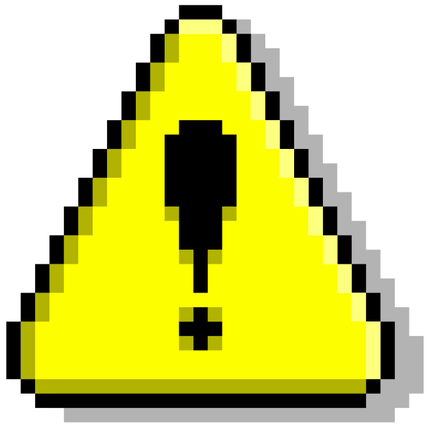
Home **Content** Contact

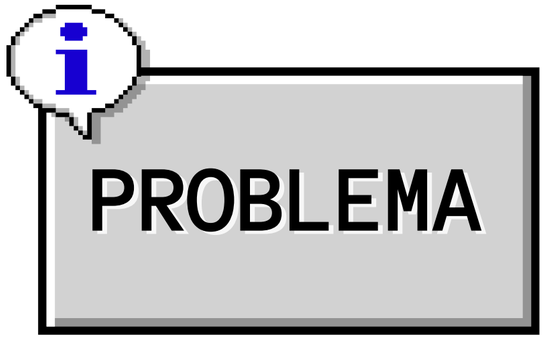


- Patrón de diseño estructural
- **Oculta** la complejidad del sistema y facilita el uso.
- El cliente accede a las funcionalidades sin la necesidad de comprender la estructura interna.
- Acoplamiento débil, mantenibilidad y escalabilidad

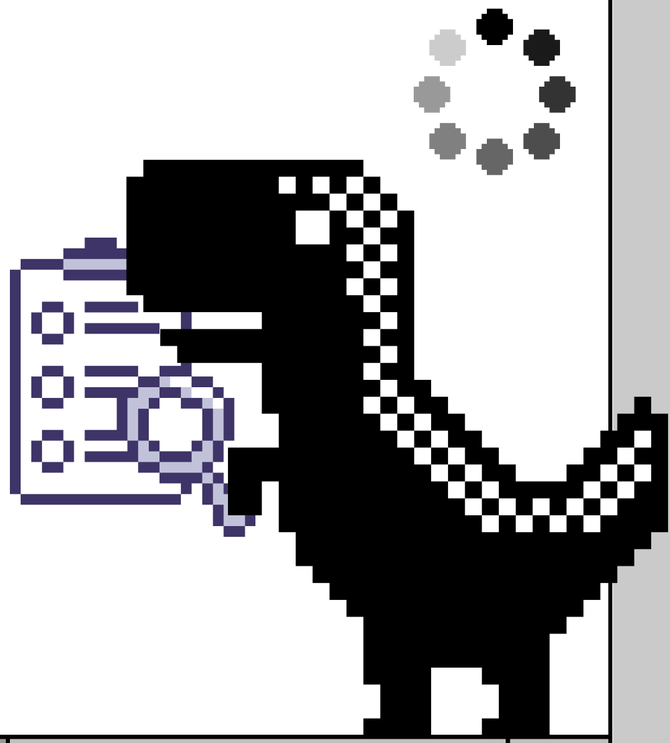
Patrón de diseño/Problema

Home **Content** Contact

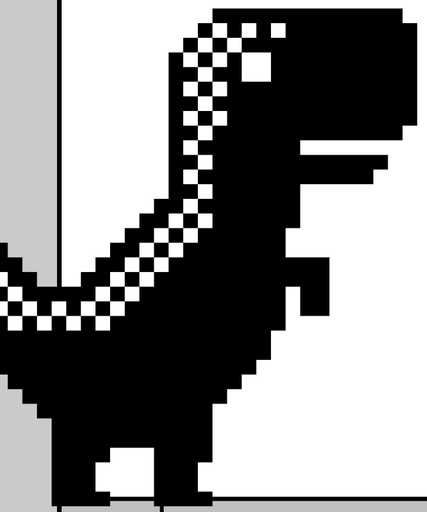




Sistemas que involucran múltiples clases y componentes de un subsistema, tienen la **complejidad** de tener que gestionar las interacciones entre estos componentes.
La complejidad resulta en **errores**, duplicación de códgio y dificultades para realizar cambios.

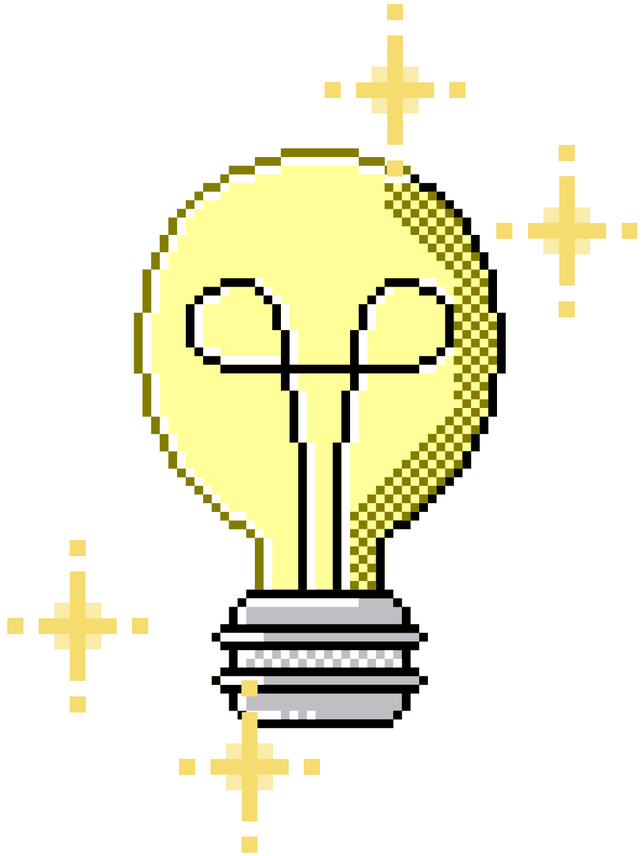


SOLUCIÓN



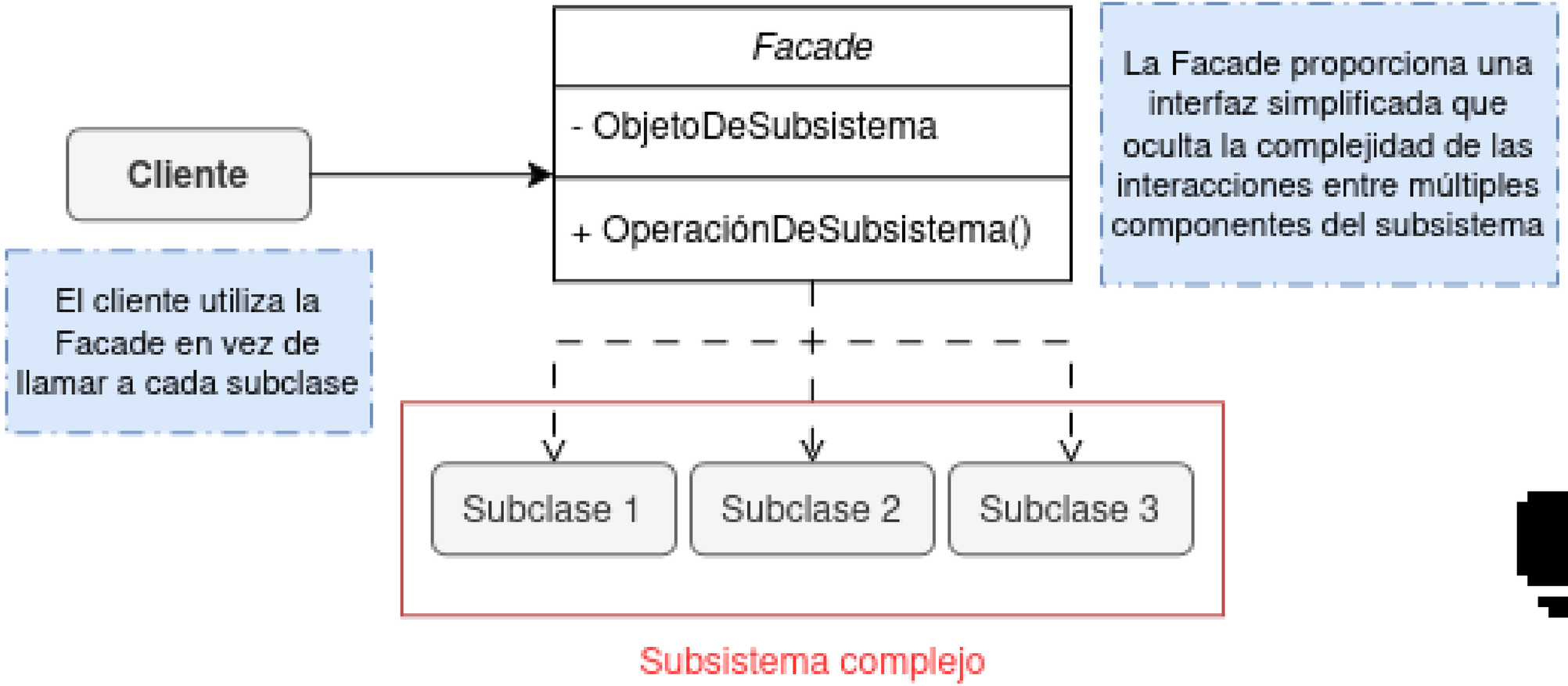
Fachada actúa como una interfaz simplificada.

- Encapsula.
- Proporciona acceso **sencillo** y directo.
- **Reduce** la complejidad, **mejora** mantenibilidad.



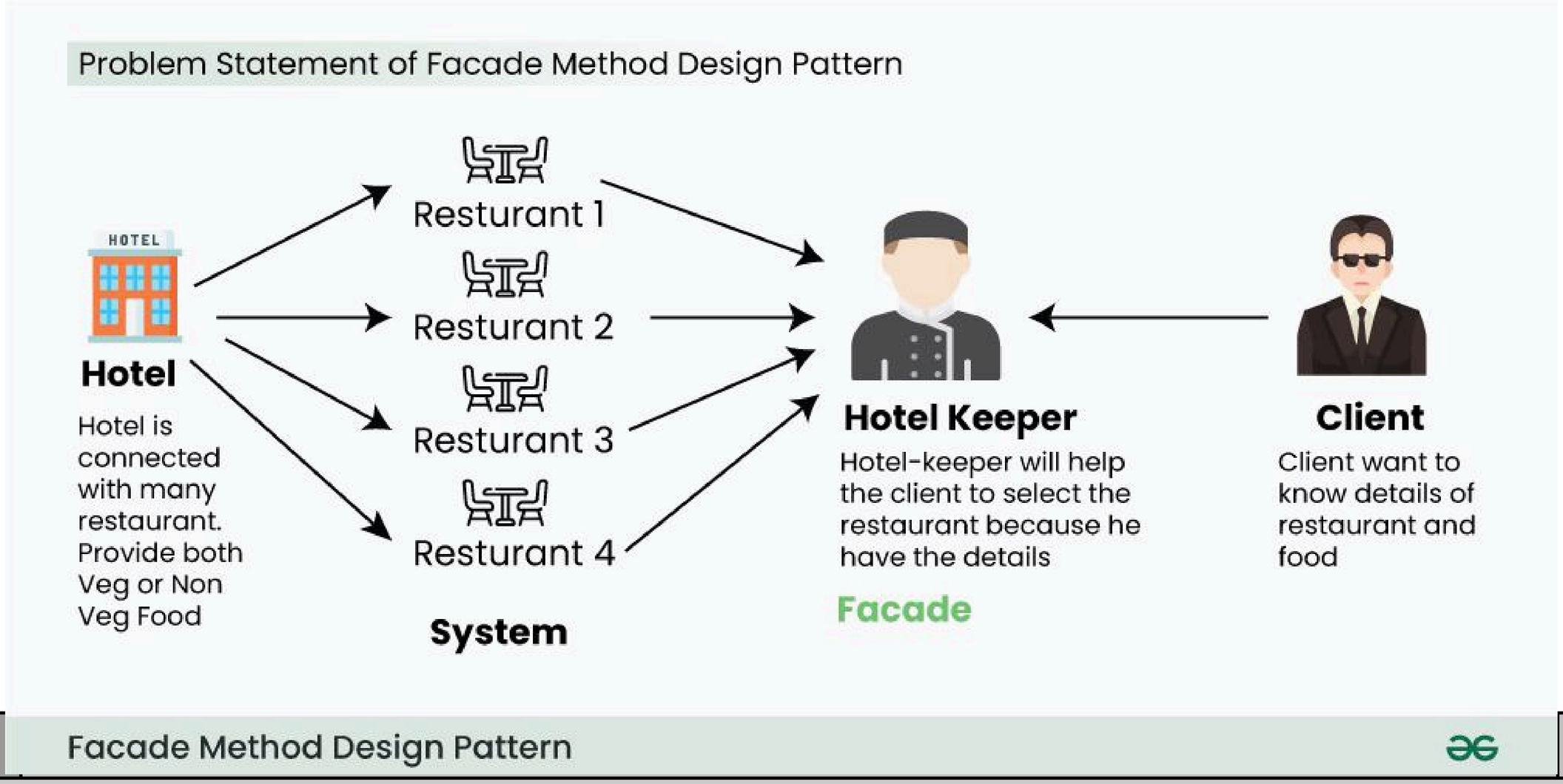
Patrón de diseño/Estructura

Home Content Contact



Patrón de diseño/Estructura

Home Content Contact



Patrón de diseño/Ventajas y Desventajas

Home **Content** Contact

VENTAJAS Y DESVENTAJAS

Ventajas:

- Mantenibilidad
- Encapsulamiento
- Acoplamiento reducido
- Simplificación de la interfaz

Desventajas:

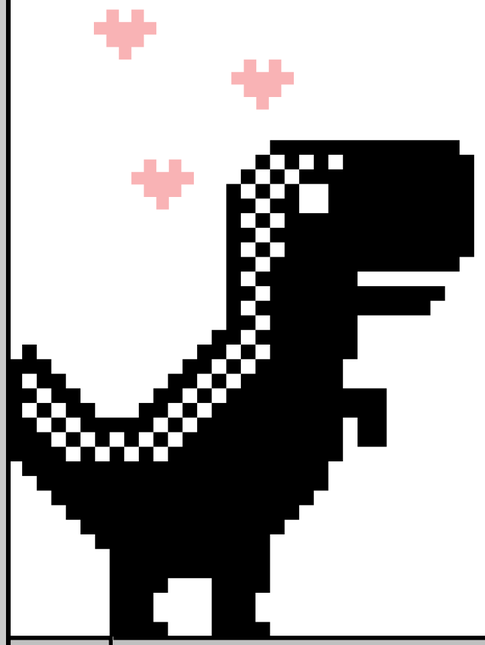
- Incremento de complejidad
- Flexibilidad reducida
- Sobreingeniería
- Sobrecarga de rendimiento

Yes

Cancel

Patrón de diseño/Testing

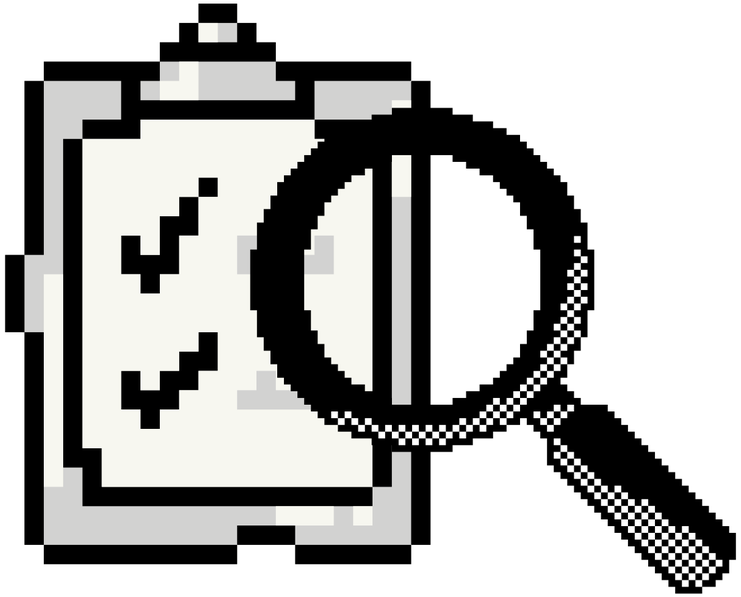
Home **Content** Contact



TESTING

Facilita las pruebas unitarias:

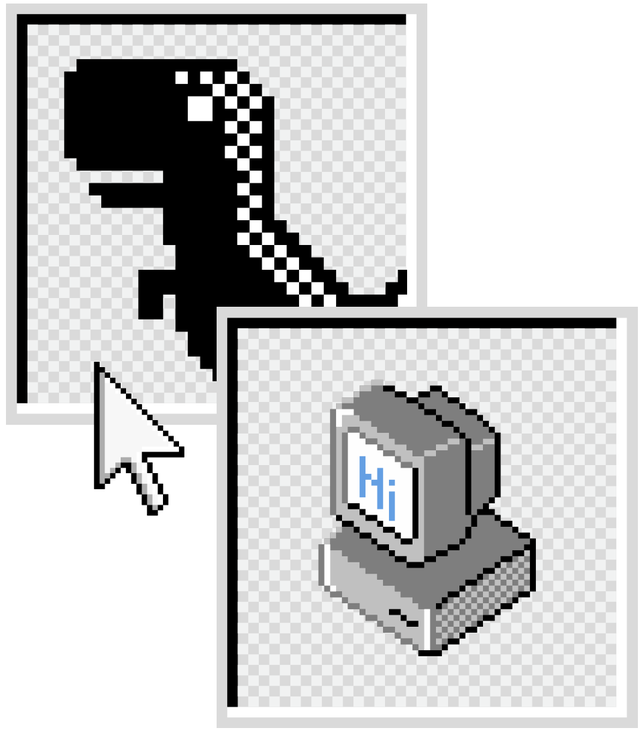
- Mocking simplificado
- Reducción de dependencias
- Pruebas más rápidas



Patrón de diseño/src/main.py

Home **Content** Contact

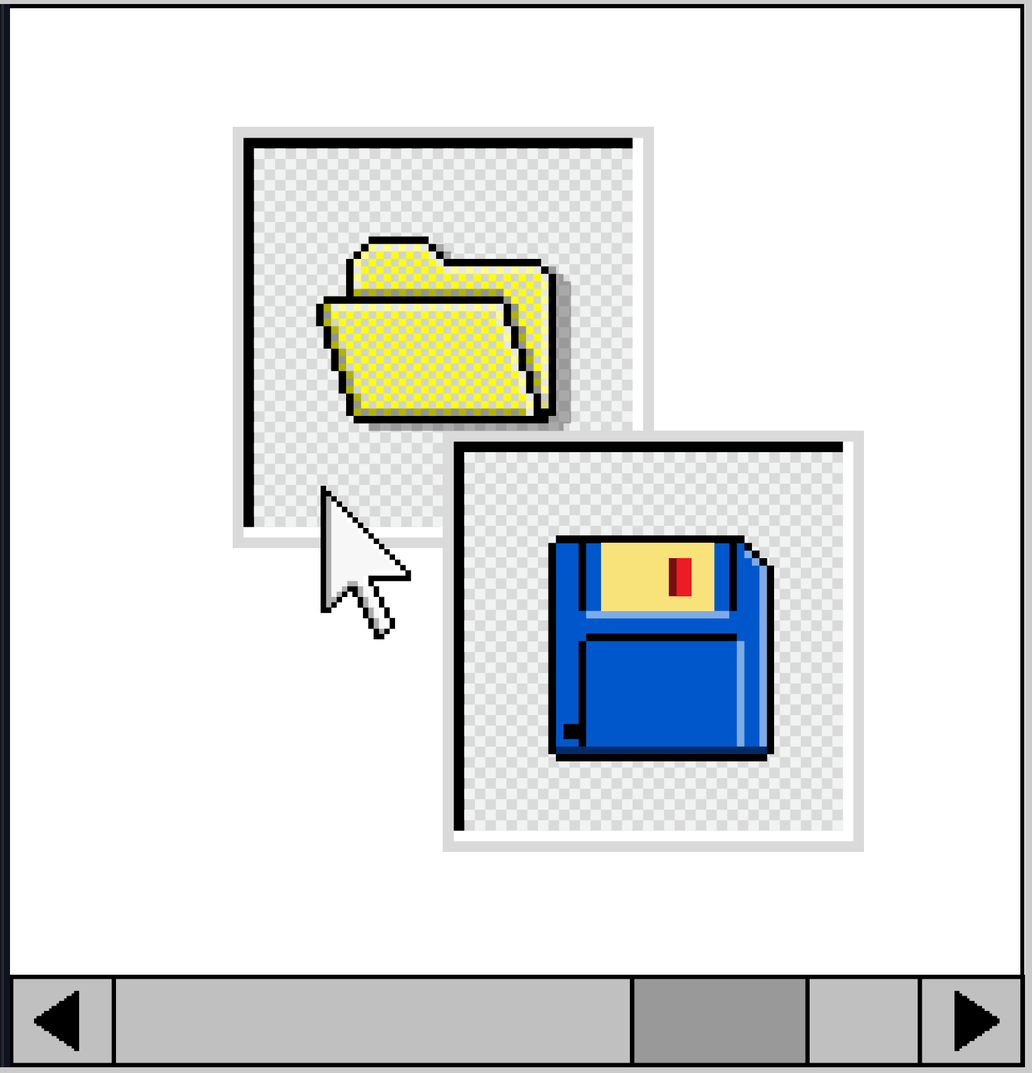
```python
from RecipeAPI import RecipeAPI


def main():
    api = RecipeAPI()

    # Cliente solo requiere llamar a un método simple (la fachada)
    result = api.get_recipe_for_user("user123", "recipe456")
    print(f"Receta adaptada: {result}")


if __name__ == "__main__":
    main()
```



Patrón de diseño/src/RecipeAPI.py

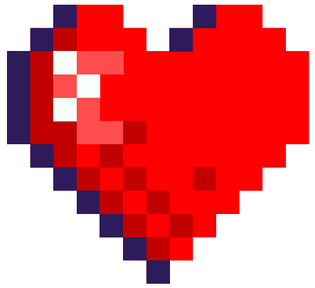
Home **Content** Contact

```python
class RecipeAPI:
    """
    Fachada que simplifica el acceso a los subsistemas de recetas y usuarios.
    Proporciona una interfaz unificada para operaciones complejas del recetario.
    """

    def __init__(self):
        # Interfaces a los subsistemas
        self._recipes: IGetRecipe = GetRecipe()
        self._users: IUserProfile = UserProfile()

    def get_recipe_for_user(self, user_id, recipe_id):
        """Obtiene toda la información de una receta adaptada al usuario"""
        user_prefs = self._users.get_user_preferences(user_id)
        recipe_details = self._recipes.get_recipe_details(recipe_id)
        ingredients = self._recipes.get_ingredients(recipe_id)

        # Filtrar ingredientes según preferencias del usuario
        filtered_ingredients = self._filter_ingredients(
            ingredients, user_prefs)

        return {
            "recipe": recipe_details,
            "ingredients": filtered_ingredients,
            "adapted_for": user_prefs
        }
```

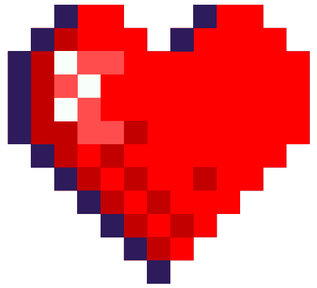


Patrón de diseño

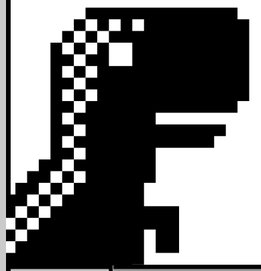
Home Content Contact



MUCHAS GRACIAS



Por su atención :)



Ok