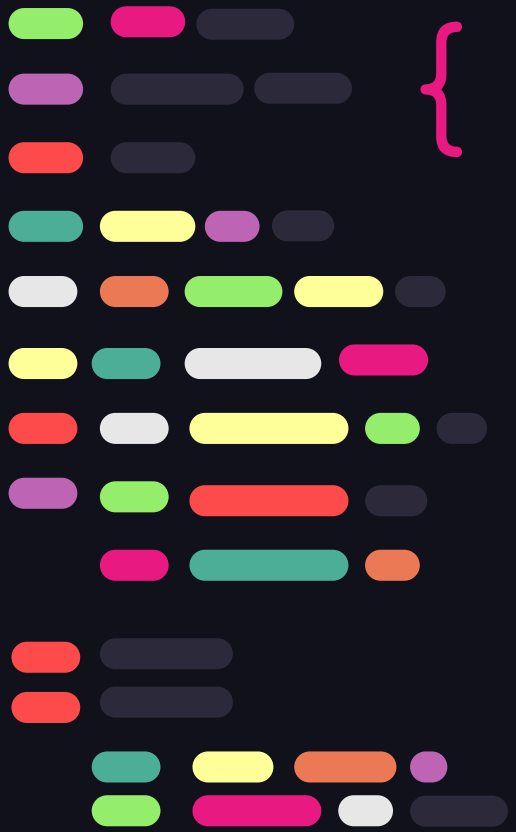


Mediator Pattern

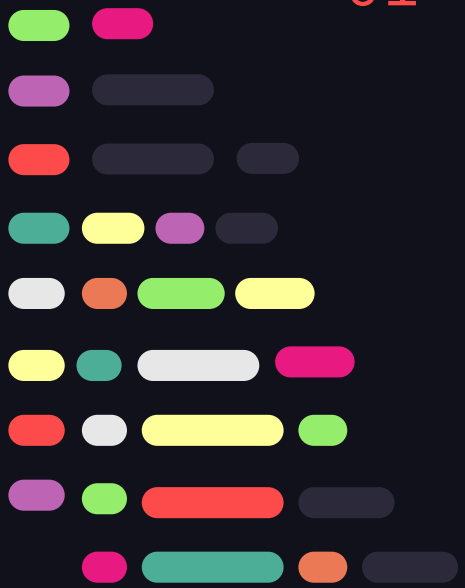
Cristopher Hernandez - C13632
Esteban Rojas - C06816

¿Que es Mediator?

Patrón de diseño el cual permite reducir las dependencias que existen entre objetos.

Fuerza a los objetos a comunicarse por un objeto mediador.

Problema

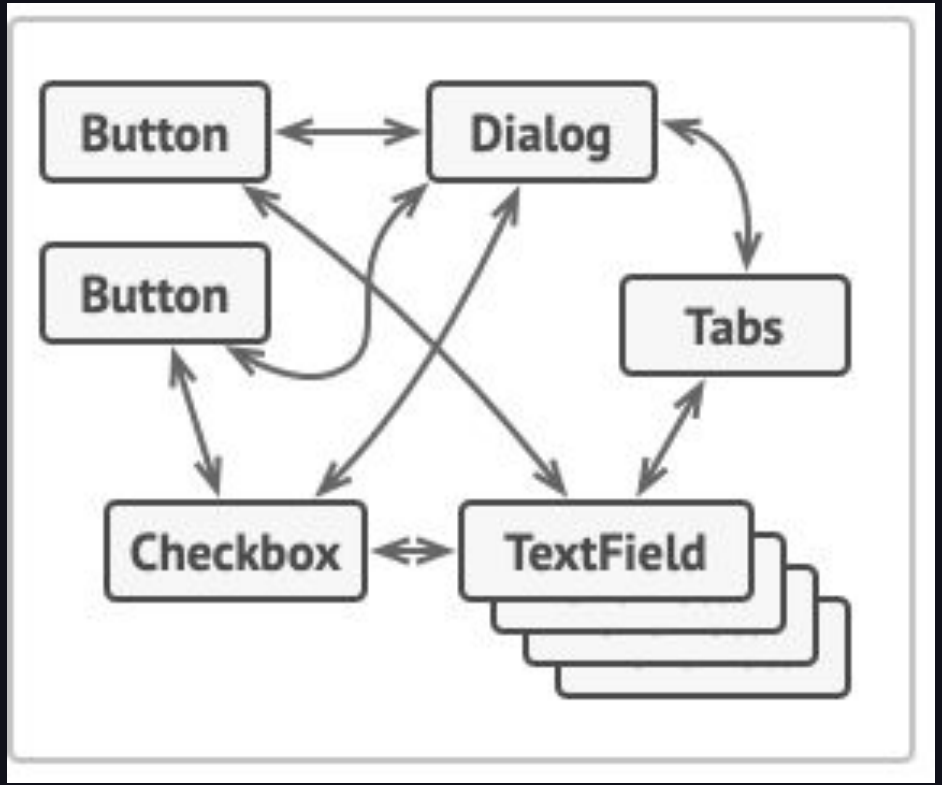


01 Clases fuertemente acopladas

02 No podemos reutilizar componentes

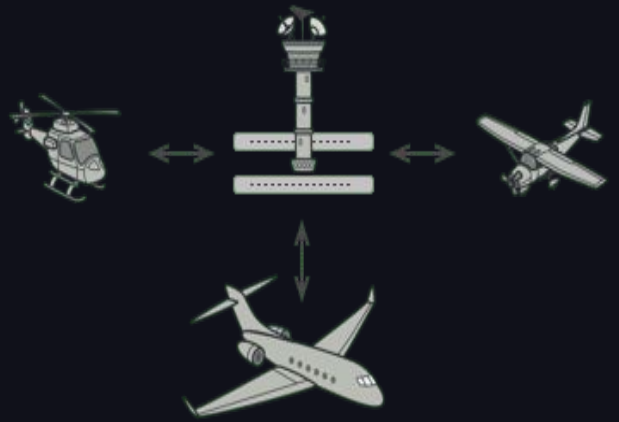
03 Creamos el mismo comportamiento múltiples veces

Ejemplo de problema

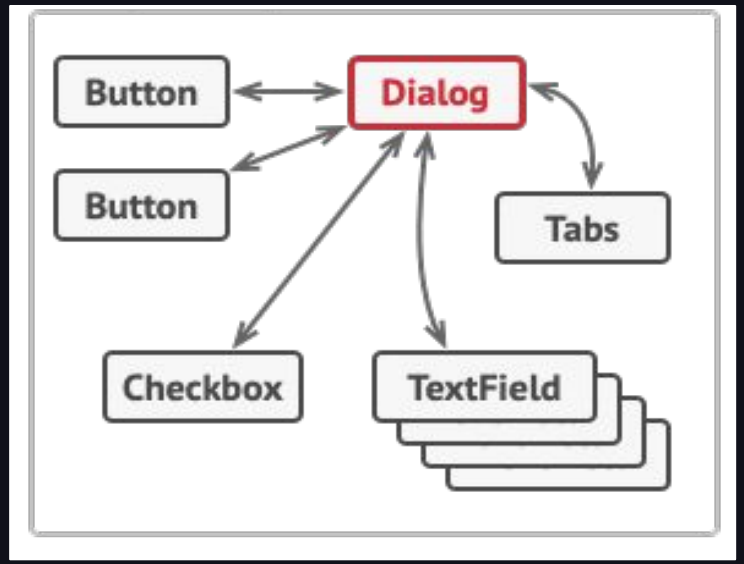


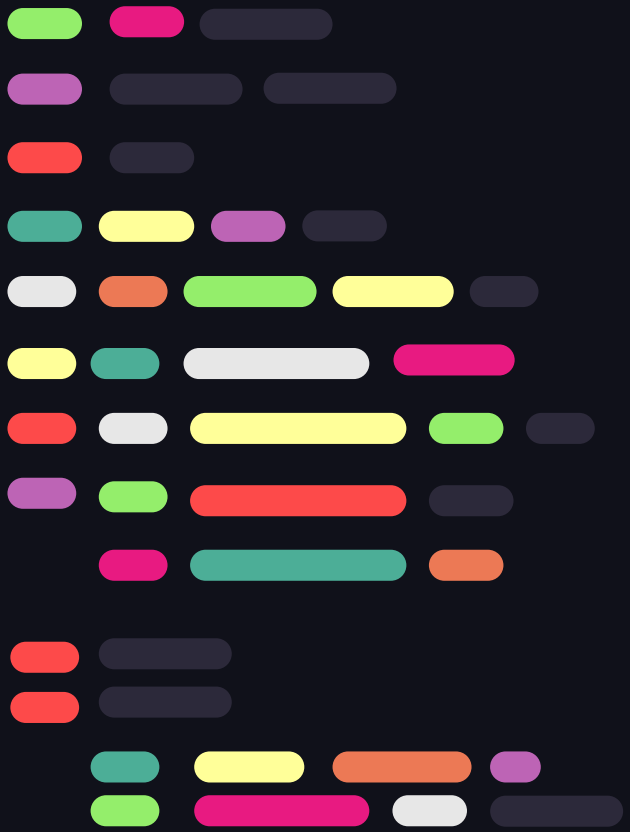
Solución

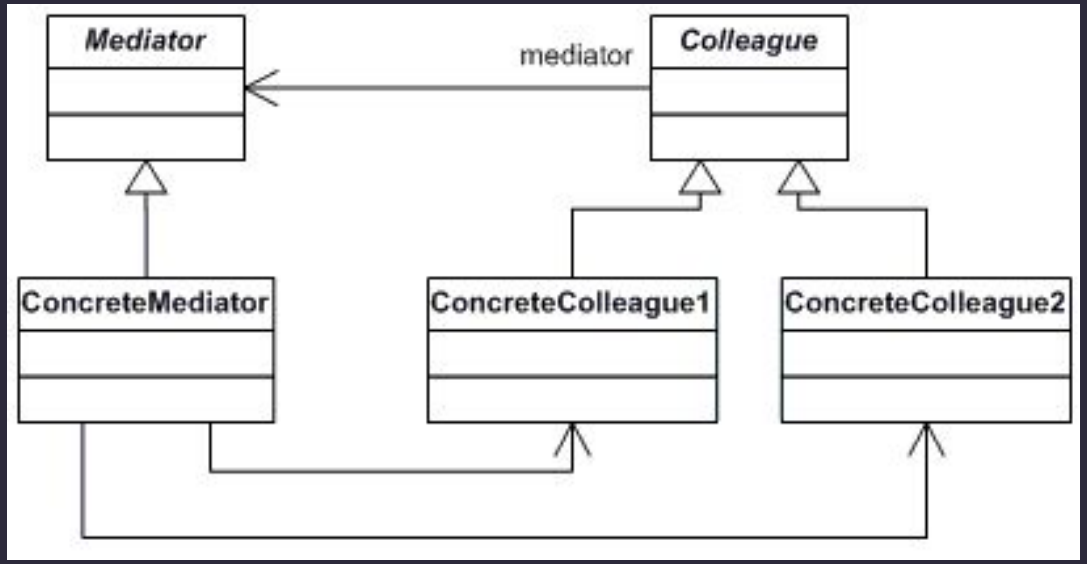
- Detenemos toda comunicación directa.
- Pasan a depender únicamente de una clase mediadora.



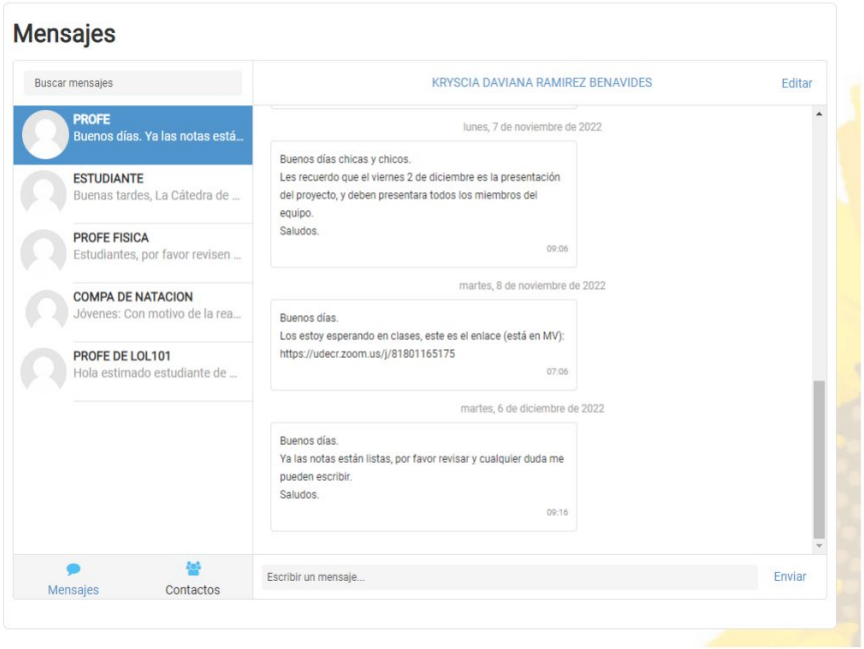
Ejemplo de la solución

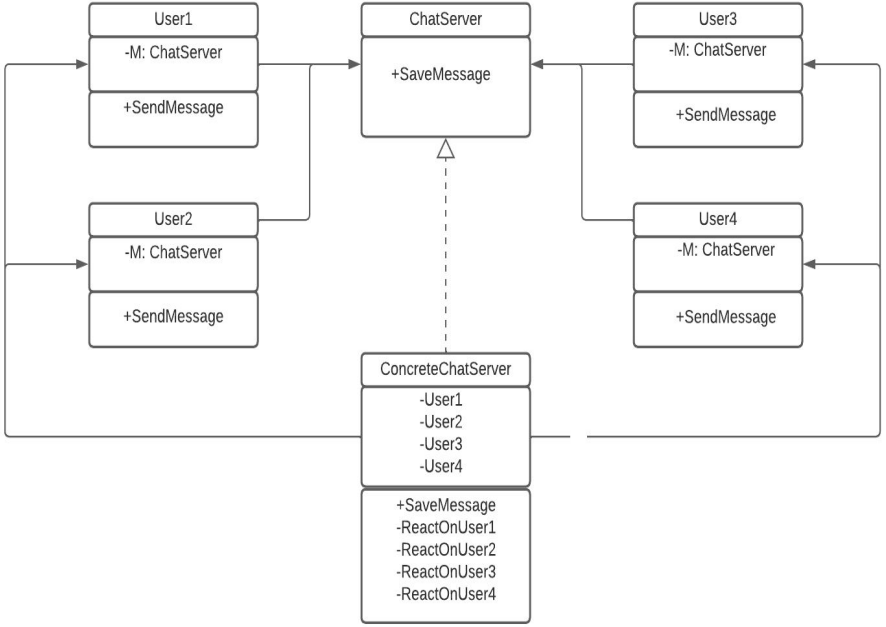





Mediator
mediator
Colleague
ConcreteMediator
ConcreteColleague1
ConcreteColleague2

Ejemplo en Mediación Virtual





Ejemplo en código

```python
from __future__ import annotations
from abc import ABC

class Mediator(ABC):

    def notify(self, sender: object, event: str) -> None:
        pass
```

Ejemplo en código

```
class ConcreteMediator(Mediator):
    def __init__(self, component1: Component1, component2: Component2) -> None:
        self._component1 = component1
        self._component1.mediator = self
        self._component2 = component2
        self._component2.mediator = self

    def notify(self, sender: object, event: str) -> None:
        if event == "A":
            print("Mediator reacts on A and triggers following operations:")
            self._component2.do_c()
        elif event == "D":
            print("Mediator reacts on D and triggers following operations:")
            self._component1.do_b()
            self._component2.do_c()
```

Ejemplo en código

```python
class BaseComponent:

    def __init__(self, mediator: Mediator = None) -> None:
        self._mediator = mediator

    @property
    def mediator(self) -> Mediator:
        return self._mediator

    @mediator.setter
    def mediator(self, mediator: Mediator) -> None:
        self._mediator = mediator
```

Ejemplo en código

```
class Component1(BaseComponent):
    def do_a(self) -> None:
        print("Component 1 does A.")
        self.mediator.notify(self, "A")

    def do_b(self) -> None:
        print("Component 1 does B.")
        self.mediator.notify(self, "B")


class Component2(BaseComponent):
    def do_c(self) -> None:
        print("Component 2 does C.")
        self.mediator.notify(self, "C")

    def do_d(self) -> None:
        print("Component 2 does D.")
        self.mediator.notify(self, "D")
```

Ejemplo en código

```
if __name__ == "__main__":
    # The client code.
    c1 = Component1()
    c2 = Component2()
    mediator = ConcreteMediator(c1, c2)

    print("Client triggers operation A.")
    c1.do_a()

    print("\n", end="")

    print("Client triggers operation D.")
    c2.do_d()
```

Salida esperada

```
Client triggers operation A.
Component 1 does A.
Mediator reacts on A and triggers following operations:
Component 2 does C.

Client triggers operation D.
Component 2 does D.
Mediator reacts on D and triggers following operations:
Component 1 does B.
Component 2 does C.
```

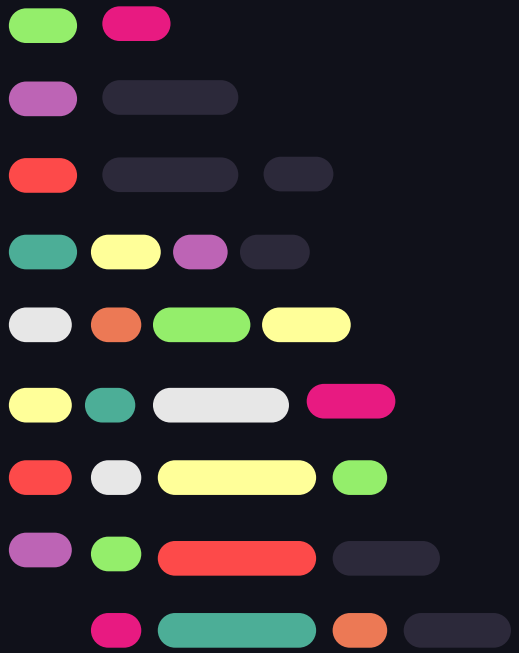
Ventajas

1. Todas las comunicaciones se centralizan para tener un mejor control.

2. Se puede cambiar la interacción sin la necesidad de cambiar componentes.

3. Mayor desacoplamiento entre componentes.

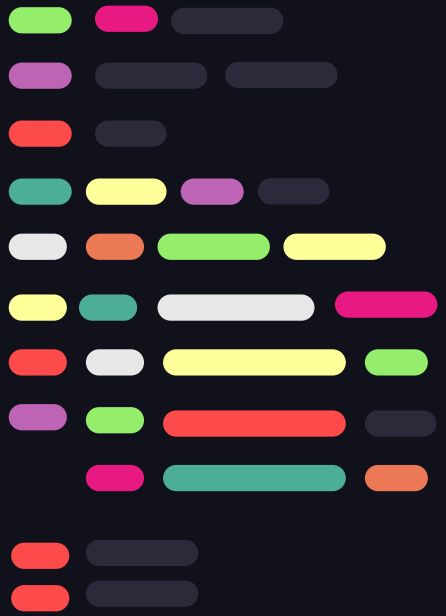
Desventaja



La clase mediator podría convertirse en un God Object.

- God object es un componente en un sistema que posee muchas funcionalidades y que conoce demasiado de otras clases.

Recomendaciones al implementar



1. El mediador se puede encargar de crear y destruir.

2. Los componentes deben referenciar al objeto mediador correcto.

3. Solo comunicar componentes a través del mediador.

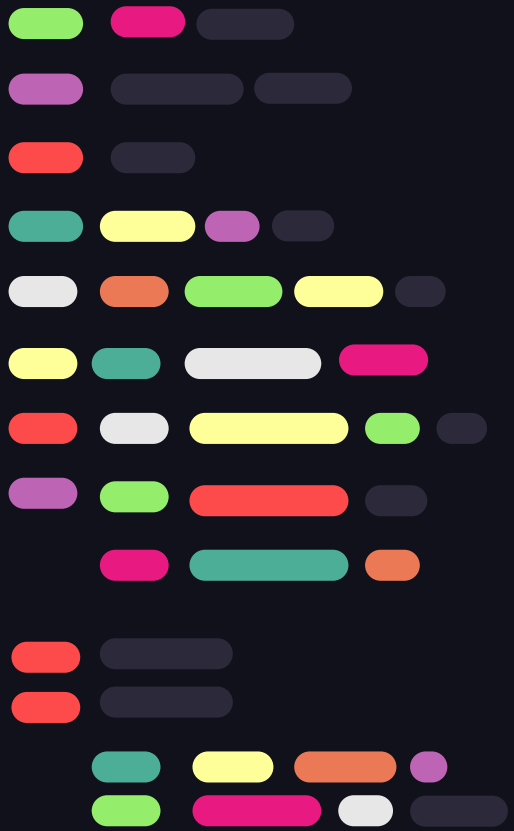
Patrones relacionados

Facade

- Facade: interacción entre el usuario y los sistemas
- Mediator: interacción entre componentes.

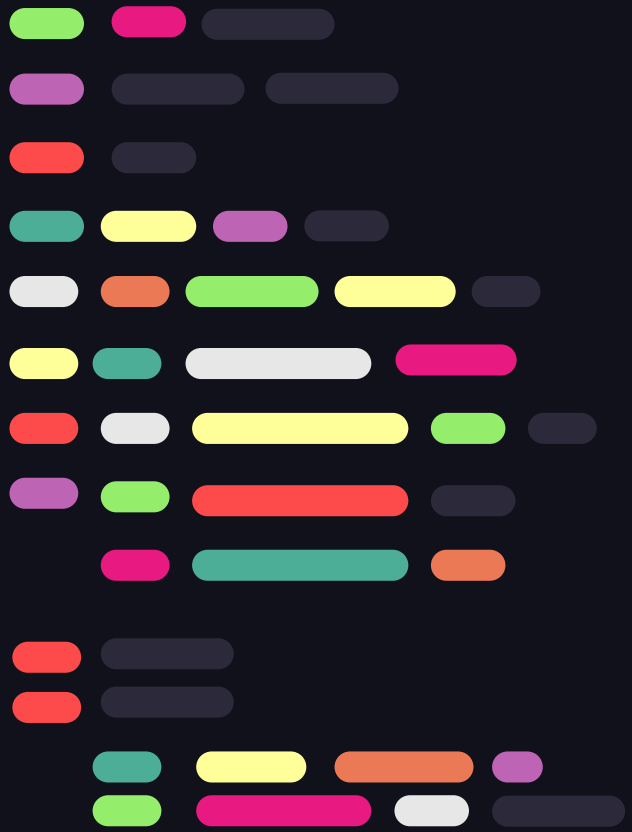
Observer

El patrón Mediator puede funcionar como Observer, si trabaja como notificador.



Gracias

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**



Actividad