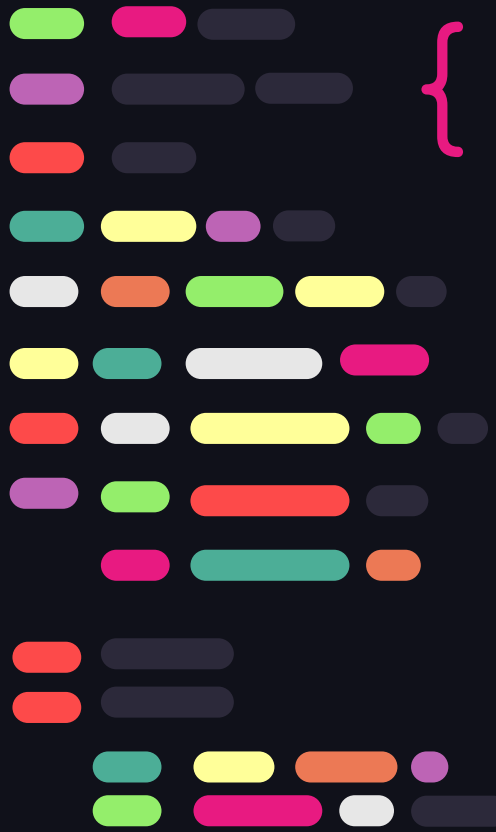


Patrón de diseño

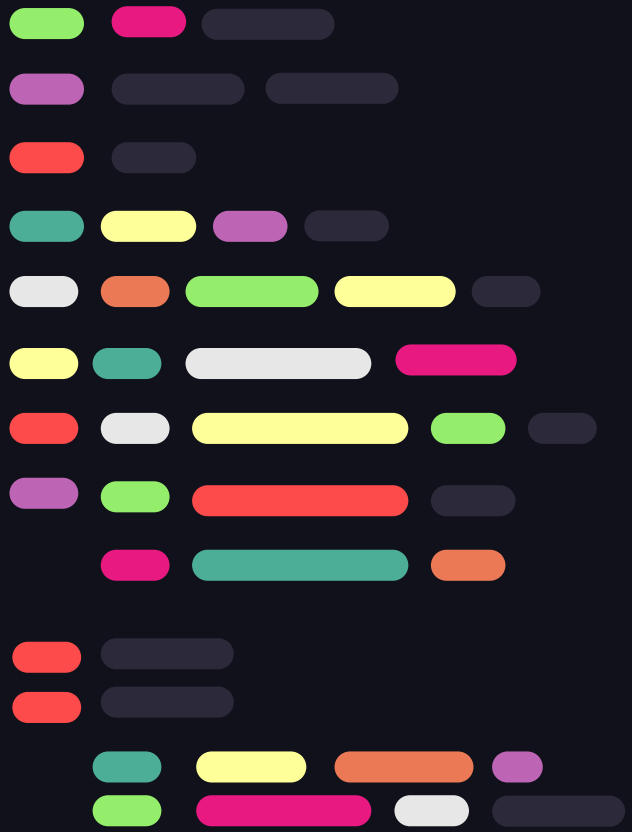
Mediator

¿Que es?

< El mediador es un patrón de diseño conductual que le permite reducir las dependencias caóticas entre objetos. >

¿Para qué sirve?

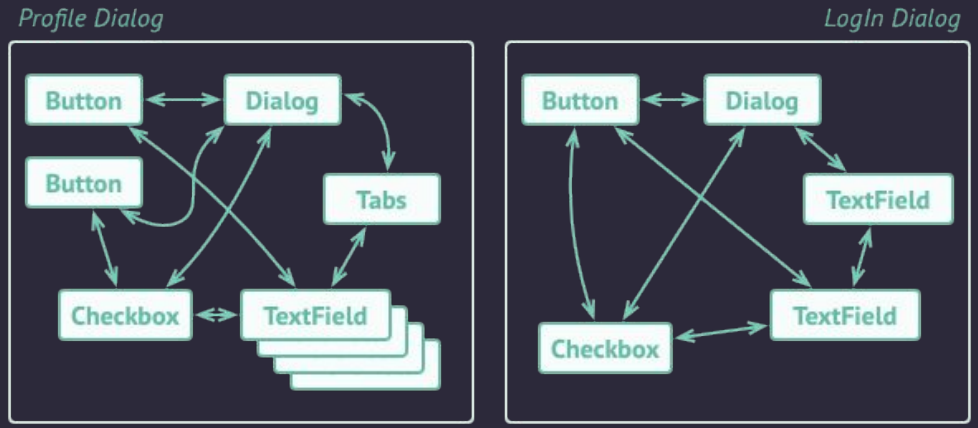
El patrón de diseño Mediator sirve para facilitar la comunicación entre diferentes componentes de un sistema, evitando que estos se comuniquen directamente entre sí

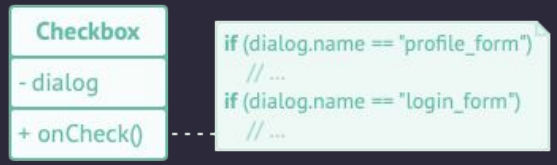


Problemas

- Escalabilidad
- Flexibilidad
- Modularidad
- Reusabilidad
- Eficiencia en el Desarrollo

Digamos que tenemos dialog para crear y editar perfiles de clientes, que contiene varios form controls como TextField, Checkbox, Buttons, etc.

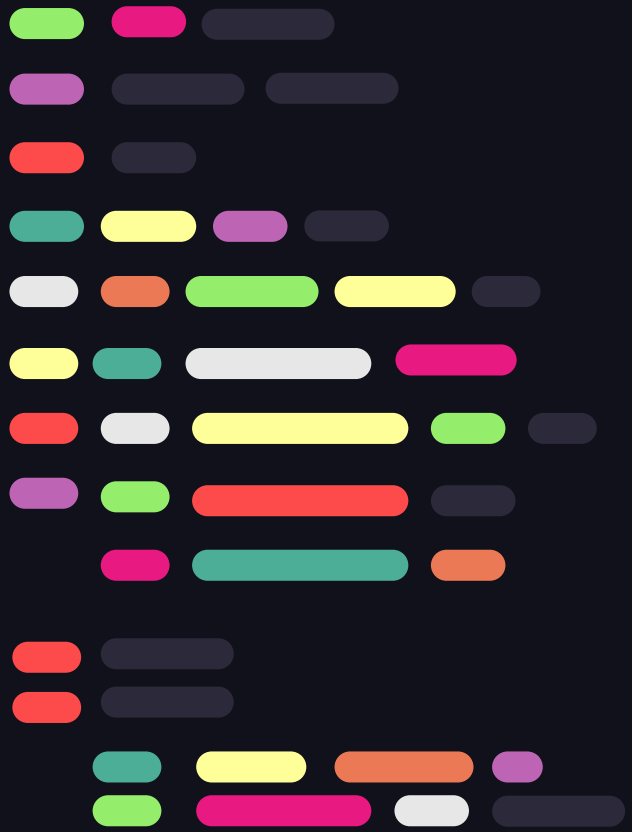




{

Algunos elementos del formulario pueden interactuar con otros, como por ejemplo un checkbox de “Tengo un perro” que despliegue un input de texto para ingresar el nombre del perro

}

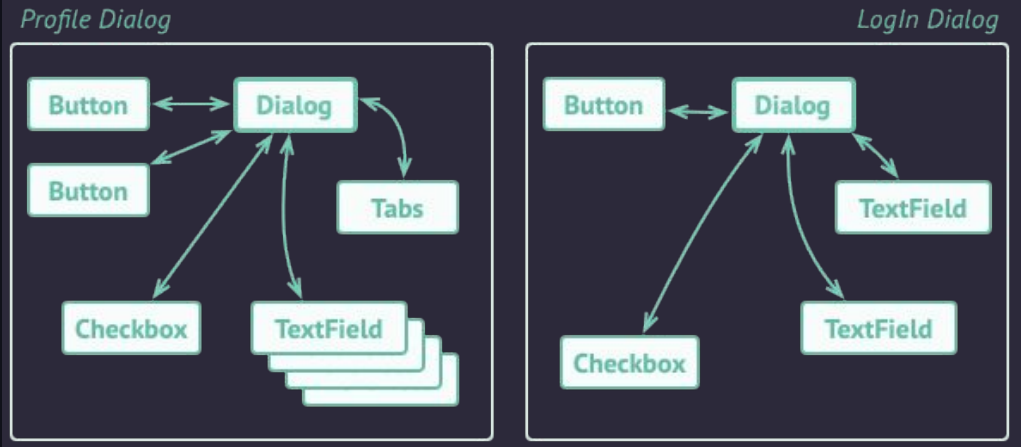


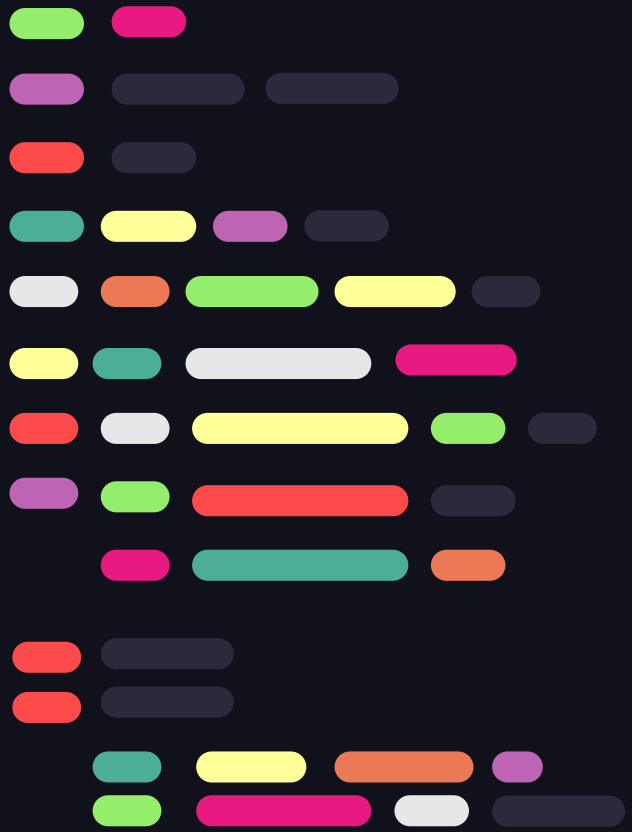
{ ..

Solucion

} ..

El patrón de Mediador sugiere que debes cesar toda comunicación directa entre los componentes que quieres hacer independientes el uno del otro.

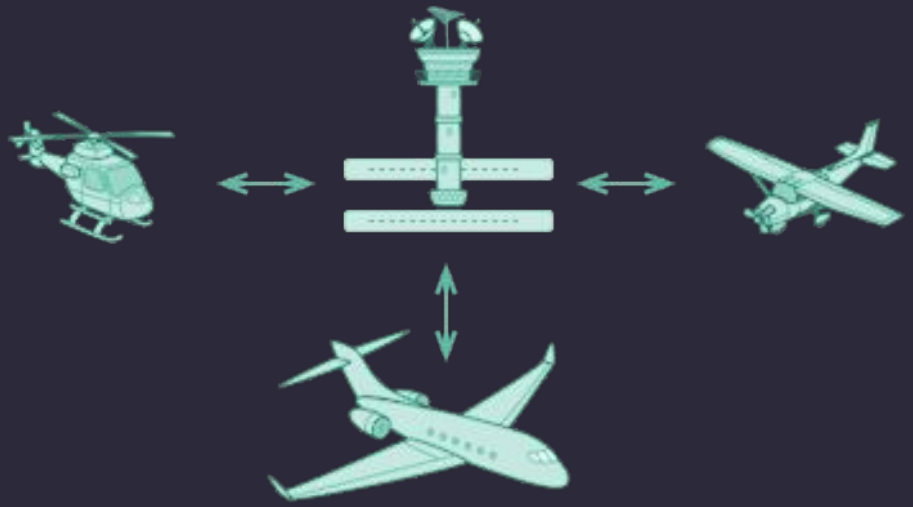




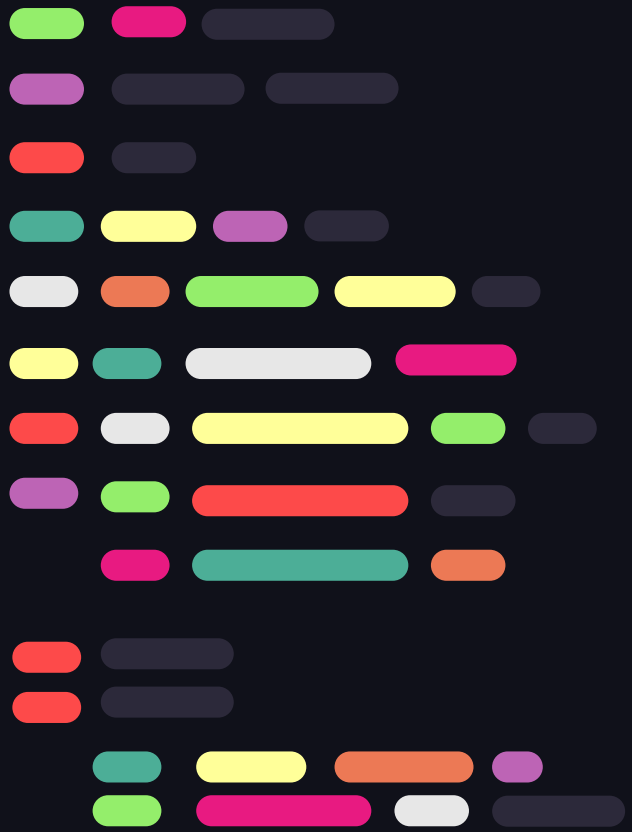
{ ..

Analogia del mundo real

} ..



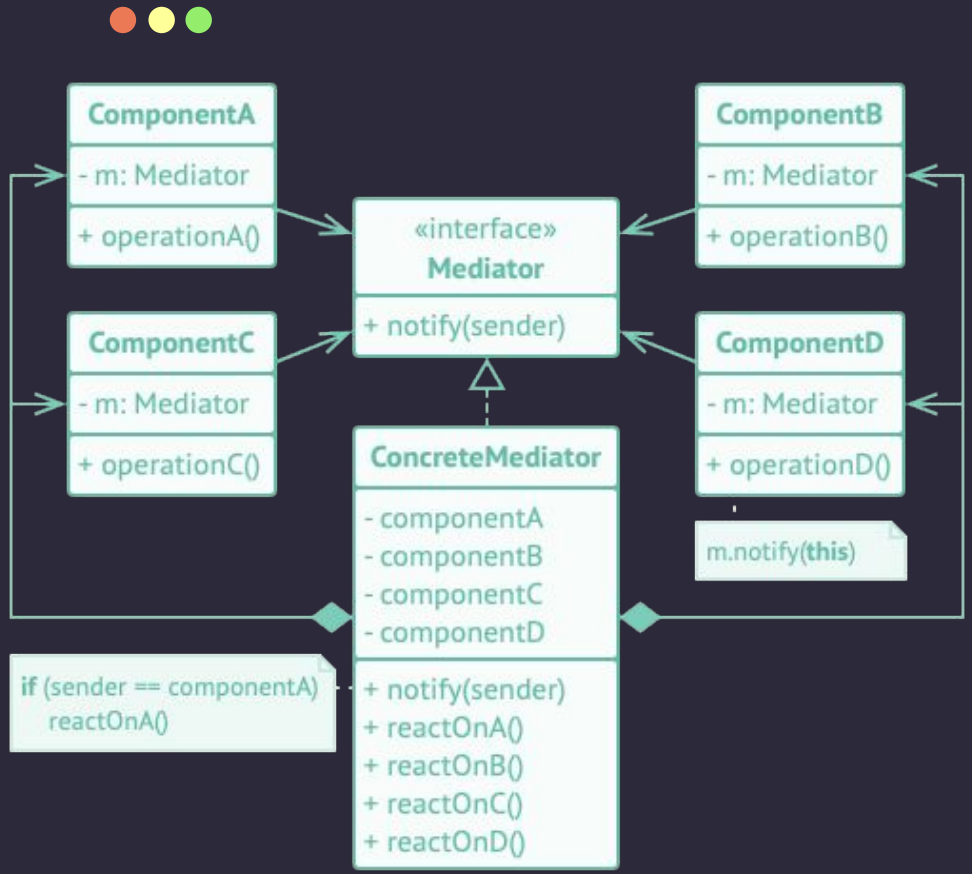
La analogía del mundo real se basa en una torre de control de tráfico aéreo. En este contexto, los pilotos de aviones no se comunican directamente entre sí para coordinar el aterrizaje.



{ ..

Estructura

} ..



El patrón se compone de tres elementos principales:

- Componentes
- Mediador
- Mediador concreto

¿Como implementar?

Pasos para implementar el patrón de diseño:

1. Identificación de Clases Acopladas.
2. Declaración de la Interfaz Mediadora.
3. Reutilización mediante Interfaz Genérica.
4. Implementación del Mediador.
5. Referencia del Componente al Mediador.
6. Centralización de la Comunicación.

Ventajas y desventajas

Ventajas

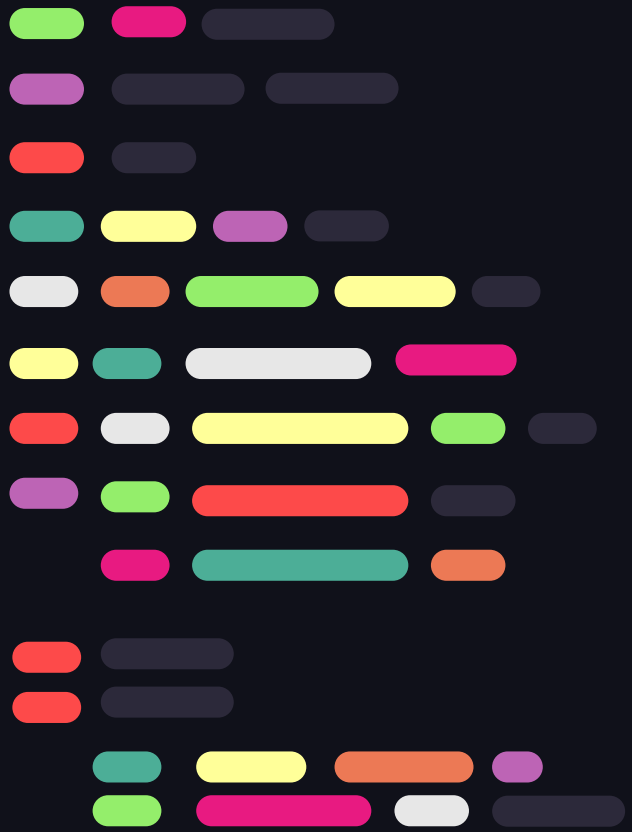
- Principio de responsabilidad unica
- Principio de abierto/cerrado
- Puede reducir el acoplamiento entre varios componentes de un programa.
- Puede reutilizar componentes individuales más fácilmente.

Desventajas

- Con el tiempo el mediador puede evolucionar a objeto que hace referencia a un gran número de tipos distintos, puede tener demasiados métodos no relacionados o no categorizados, o alguna combinación de ambos.

Casos de uso

- Sistemas GUI (Interfaz Gráfica de Usuario)
- Sistemas de Chat en Tiempo Real
- Juegos Multijugador
- En editores gráficos
- Sistemas de Control de Tráfico
- Sistemas de Gestión de Eventos



{ ..

Gracias!

} ..