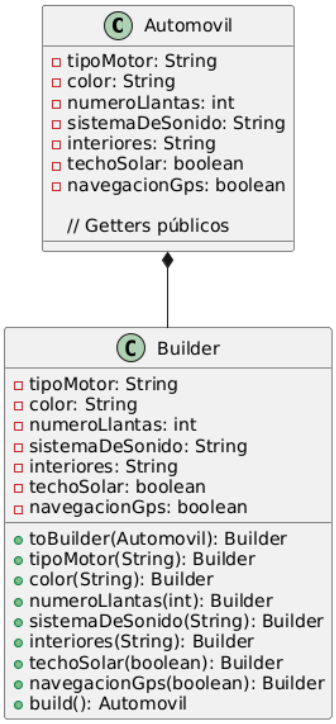
Taller patrones de diseño – Grupo 2

Jesus Enrique Jimenez Posada
Anderson Florez Gutierrez
Myriam Andrea Martinez Fontecha
Marlon Andres Nustes Hernandez
Oscar Dario Malagon Murcia
Fredy Orlando Pulido Quintero
Robinson David Cely Rincon

Escenario 1

- Tipo de patrón: creacional
- Patrón seleccionado: Builder
- Diagrama de clases:

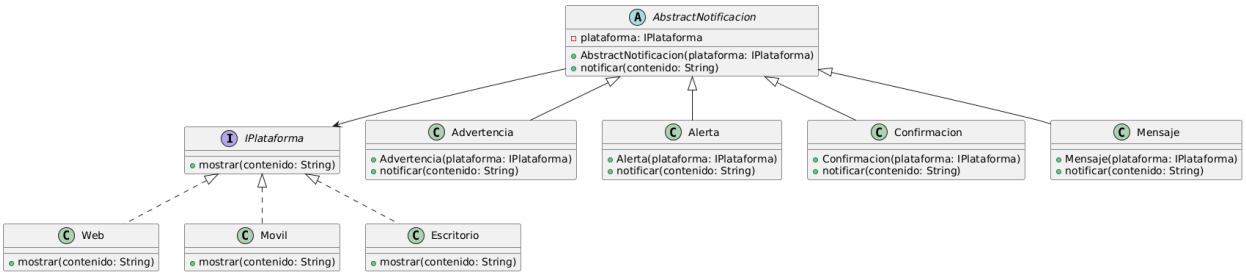


- Razones del diseño/implementación:
    - Permite crear objetos de tipo Automovil con muchos parámetros opcionales sin necesidad de tener muchos constructores
    - Una vez creado el objeto Automovil no se pueden cambiar los valores de sus atributos (inmutable). Sin embargo, se implementa un método toBuilder disponible en el Builder, pero de todas maneras se crearía una nueva instancia de Automovil.

    - A diferencia del patrón builder complejo no se necesita una interface *IBuilder* o una clase *Director* ya que:
        - no tenemos la necesidad de crear otros objetos diferentes a Automovil usando la misma estructura del builder
        - no tenemos la necesidad de crear diferentes configuraciones de tipo Automovil con diferentes implementaciones de builder
- Código: https://github.com/oscar-malagon/Arquitectura-de-Software/tree/main/Taller1_PatronesDiseno/src/main/java/edu/unisabana/patrones/scenario1

Escenario 2

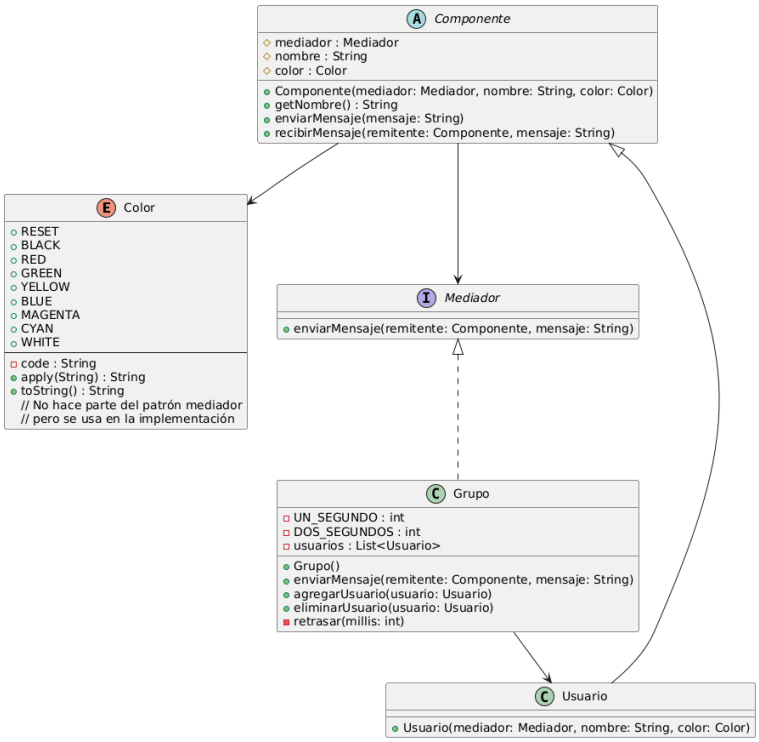
- Tipo de patrón: estructural
- Patrón seleccionado: Bridge
- Diagrama de clases:



- Código: https://github.com/oscar-malagon/Arquitectura-de-Software/tree/main/Taller1_PatronesDiseno/src/main/java/edu/unisabana/patrones/scenario2

Escenario 3

- Tipo de patrón: Comportamental
- Patrón seleccionado: Mediator / Intermediary / Controller
- Diagrama de clases:



- Código: https://github.com/oscar-malagon/Arquitectura-de-Software/tree/main/Taller1_PatronesDiseno/src/main/java/edu/unisabana/patrones/scenario3