

# **Programación con Objetos 2**

## **Trabajo Final**

### **Ciencia participativa y juegos**

- Agustin Campos -  
Email : agustin.campos17@outlook.com
- Ezequiel Gonzalez -  
Email: ezegonzalez912@gmail.com
- Federico Hernan Sanchez -  
Email : federicosanchezunq@gmail.com

### **Patrones de diseño**

- **State**

Se utilizó el patrón State en la clase DesafioDeUsuario . La clase se comporta diferente dependiendo del estado en el que se encuentra . Se

modeló la clase DesafioState con sus respectivos estados y su comportamiento

Los participantes son :

-Contexto : DesafioDeUsuario

-State : DesafioState

-ConcreteStates : EstadoCompleto y EstadoIncompleto

- **Strategy**

Se utilizó el patrón Strategy para la recomendación de desafíos para el usuario . El usuario posee un atributo con la estrategia que desea utilizar

Los participantes son :

-Contexto : Usuario

-Strategy : Recomendación

-ConcreteStrategy : Favoritos y PreferenciaEnJuego

- **Composite**

Se utilizó patrón Composite para la clase IFiltro

Participantes Del Patrón.

- IFiltro : es la interfaz que declara el protocolo común.

- FiltroIncluirTexto, FiltroIncluirCategorias, FiltroExcluirCategorias: son los objetos primitivos del patrón (Leaf 's).

- FiltroCompuesto: Composite, este es la clase abstracta que abstrae el comportamiento común de los filtros compuestos, de este extienden FiltroAND y FiltroOR.

## Decisiones de diseño

### Usuario

La clase Usuario posee una lista de proyectos, en los cuales está , una lista de muestras que recolectó , su perfil de usuario con sus diferentes atributos que sirven para poder comparar con los desafíos y calcular la coincidencia de ese desafío para esto tenemos la clase Recomendación en la cual implementamos un strategy, esto nos facilita el desarrollo permitiéndonos poder cambiar la estrategia de recomendación de desafíos fácilmente. También dentro de usuario existe un GestorDeDesafios el cual se encarga de filtrar y dar algunos datos sobre los mismos.

### Desafío

Decidimos optar por crear una clase DesafioUsuario la cual se encarga de manejar gestionar todas las funcionalidades de un desafío el cual está como atributo. Dentro de esta se encuentra un state, el cual fue elegido ya que nos permite alterar el comportamiento, ya sea cuando está completo o incompleto, esto nos permite evitar varios if's y hacer el diseño más escalable.

### Filtros

Para crear la lógica de Filtros, decidimos implementar el patrón **Composite** el cual nos permite agrupar de forma uniforme objetos primitivos (Leaf) y complejos (Composite). Llegamos a la conclusión de

que deberíamos usar este patrón porque aparte de tener filtros básicos, también existen filtros compuestos por las operaciones lógicas AND y OR.