# A la caza de las Vinchucas

Trabajo Practico Integrador

Programación con Objetos II - 2023 - 1er Semestre


## Integrantes

Agüero, Mauro.

Quaglia, Sebastian.

Agüero, Fernando.

# Decisiones

- Asumimos que las organizaciones no tienen una funcionalidad externa (por defecto) para cada evento al que se pueden registrar. Y que pueden suscribirse a una Zona sin necesariamente tener una funcionalidad definida para el evento.

- Decidimos que una **organización** puede suscribirse a la carga y validación de muestras en una **Zona de Cobertura** de forma independiente, es decir, a uno de los dos o ambos a la vez.

- Creamos una clase **PaginaWeb** que contiene los posteos de cada usuario y las zonas de cobertura para facilitar que el usuario opine y actualice todo el sistema.

- Para los operadores de los filtros de búsqueda utilizamos String para representarlos y los matcheamos mediante un switch case; dado a que es un sistema que no pensamos que se expandirá por ese lado, y el crear clases o enums para solventar el problema creemos que es sobrediseño.

- Utilizamos un **Optional** de **Opinion** en **PostMuestra** para solucionar el problema que este tiene al momento de que no haya un resultado definido o hayan opiniones con la misma cantidad de votos. En esos casos el Optional estará vacío y por lo tanto el **PostMuestra** estará como no definido.

- “Dado una muestra, conocer todas las muestras obtenidas a menos de x metros o kilómetros”. Decidimos no implementar dicha funcionalidad porque creemos que no aporta al comportamiento del sistema. Ya que las Zonas de Cobertura se encargan de esto mismo.

- Se decidió que la función de búsqueda de muestras trabajaría a futuro con una clase que no está implementada actualmente, ya que la responsabilidad será del cliente el decidir cómo se usará esa funcionalidad.

## Patrones de diseño aplicados

1. Observer:
    a. Subject:  ZonaDeCobertura
    b. Observer: ObserverZona (interfaz)
    c. Concrete Observer: Organizacion
2. State 1
    a. Context: PostMuestra
    b. State: EstadoDePost
    c. ConcreteState:
        i. EstadoDePostBasico
        ii. EstadoDePostExperto
        iii. EstadoDePostVerificado
3. State 2
    a. Context: Usuario
    b. State: EstadoUsuario
    c. ConctreteState:
        i. EstadoBasico
        ii. EstadoExperto
4. Composite
    a. Component: FiltroBusqueda
    b. Composite: ConectoresBusqueda
    c. Leaf: FiltroBusqueda
    d. Client: BusquedaDeMuestra
5. Template Method
    a. AbstractClass: ConectoresBusqueda
    b. ConcreteClass:
        i. ConectorAnd implementa:
            1. aplicar(List<PostMuestra>, List<PostMuestra>): List<PostMuestra>
        ii. ConectorOr implementa:
            1. aplicar(List<PostMuestra>, List<PostMuestra>): List<PostMuestra>
    c. TemplateMethod: filtrar(List<PostMuestra>): List<PostMuestra>
    d. PrimitiveOperation:
        i. aplicar(List<PostMuestra>, List<PostMuestra>): List<PostMuestra>

## Problemas durante el desarrollo

1. Al comienzo del desarrollo del proyecto se estaba usando un repositorio distinto del cual se usa en la entrega. Durante la primera semana, debido a un problema con las librerías de test, cualquier test que se programaba fallaba automáticamente a pesar de que para el que miembro original funcionasen de forma correcta. Debido a esto, se tomó la decisión de mudar todo el progreso a otro repositorio, el cual no presentó ninguna anormalidad similar al anterior.

2. Durante la semana final del desarrollo, uno de los miembros se encontró en la situación de no tener acceso a su computadora de trabajo debido a una falla técnica. Si bien esto no afectó el resultado final ya que se había completado todo el desarrollo previamente, se decidió que es algo que merece ser mencionado.