Trabajo Práctico Integrador

A la caza de las vinchucas

Universidad Nacional De Quilmes | 2023

Integrantes

- Gomez, Dario Gabriel - gabigomez45@hotmail.com
- Gutierrez, Aaron Gaston - aarongastongutierrez@gmail.com
- Angiolillo, Geronimo -

Diseño

Decidimos la siguiente estructura de paquetes:

web:
En este paquete se encuentra la clase Web.

- Web: se encarga de recibir una nueva opinión o una nueva muestra a agregar. Delega responsabilidades a las muestras, a los usuarios y a las zonas correspondientemente. Además, antes de agregar una opinión a un usuario, verifica si pudo ser agregada la opinión a la muestra en cuestión. Lleva una lista para todas las muestras, otra para todos los usuarios y otra para todas las zonas.

web.criterio:
En ese paquete se encuentra todo lo relacionada con los filtros y los criterios para filtrar muestras.

web.muestra:
En este paquete se encuentran todas las clases relacionadas con las Muestras

- Muestra: posee una lista de Opiniones que utiliza para devolver su resultado, este resultado depende de su estado el cual va cambiar dependiendo de quién opine en la misma de forma dinámica. También puede dar información como su fecha, foto, usuario, ubicación y dar la fecha de la última votación.
- MuestraEstado: Es la interfaz que se usa como Estado y posee dos mensajes abstractos que son devolver el resultado actual y agregar una opinión
- MuestraEstadoNoVerificada: Este es un Estado Concreto con la capacidad de agregar una opinión, en ese proceso identifica si la opinión es de un experto y en caso de que lo sea cambia el estado. También se encarga de tomar en cuenta si un usuario ya opinó en esa muestra y de analizar todas las opiniones para definir cuál es el resultado de la muestra incluyendo el empate.
- MuestraEstadoVerificadaPorExperto: Este es un Estado Concreto con la capacidad de devolver una excepción si se quiere agregar una opinión de una persona que no es experta o si ya opino anteriormente, aparte también a la hora de devolver el resultado solo toma en cuenta a los expertos que opinaron.

web.zonaDeCobertura:
En este paquete se encuentran todas las clases relacionadas con las zonas de cobertura.

- ZonaDeCobertura: Puede registrar y desregistrar observadores para luego notificar cuando sea correspondiente. Tambien se puede registrar a las muestras de interjet, para que cuando sean validadas pueda avisar a los organizaciones correspondientes. Sabe si una Muestra esta dentro de su zona de cobertura y sabe si otra Zona solapa consigo.
- Organizacion: son observadores que realizan funciones externas según la notificación que reciban.
- FuncionDeImpresion: es una función externa que realiza un evento en específico.

- Interfaces: ObservadorOrg, ObservableOrg, ObservadorZona, Observable Zona, FuncionExterna.
- Enums: TipoDeOrganizacion.

web.opinion:
En este paquete se encuentran las clase Opinion y el enum TipoDeOpinion.
- Opinion: es la clase que representa una opinión. Se crea con un tipo de opinión, un usuario y una muestra. Además contiene la fecha en la que se creó.
- TipoDeOpinion: es un enumerativo que representa cada tipo de opinión.

web.ubicacion:
En este paquete se encuentra la clase Ubicacion.
- Ubicacion: es la clase que representa una ubicación. También sabe la distancia en kilómetros que tiene con otra ubicación, utilizando la fórmula de Haversine. Dada una lista de ubicaciones y una distancia puede retornar las ubicaciones a menos de esa distancia.

Patrones

Composite: Criterio

Al momento de hacer la búsqueda de muestras se usan filtros estos pueden ser simples o compuestos, ya que uno de los criterios de búsqueda es combinando otros criterios. Es por eso que se decidió utilizar dicho patrón.

- **cliente** -> Muestras
- **component** -> Criterio
- **composite** -> CriterioCombinado
- **leaf** -> CriterioFechaCreacion, CriterioFechaUltimaVotacion, CriterioTipoDeInsectoDetectado, CriterioNivelVerificacion

Strategy: Operador Lógico

En el caso del criterio combinado nos encontramos que dependiendo cual sea el conector (**OR** o **AND**) se siguen estrategias diferentes. Es por esta razón que se utilizó este patrón

- **Stategy** -> OperadorLogico
- **ConcreteStrategy** -> OperadorLogicoOr, OperadorLogicoAnd
- **Contex**t -> CriterioCombinado

State: EstadoUsuario

Usamos el patrón **STATE** para el estado de Usuario a raíz de que el comportamiento es distinto a partir del estado del mismo. Este puede actualizarse según la cantidad de opiniones que haga o la cantidad de muestras que publique. El caso del Estado validado,

este se valida externamente y su estado no cambia sin importar el tiempo que llevan participando, estos usuarios son siempre expertos.

- **Context** -> Usuario.
- **State** -> EstadoUsuario.
- **ConcreteState** -> EstadoBasico, EstadoExperto, EstadoValidado.

Strategy: Organizacion

Las organizaciones a la hora de registrar una muestra o validarse en una zona de interés para ellas hacen una función externa(que puede ser la misma o diferente en caso de la razón), el patrón Strategy permite cambiar la función externa.

- Stategy -> FuncionExterna
- ConcreteStrategy -> FuncionDeImpresion
- Context -> Organizacion

State: MuestraEstado

Usamos el patrón State para el estado de Muestra entendiendo que algunos métodos tiene comportamiento distinto a partir del estado de la muestra y este se puede actualizar cada vez que recibe una opinión.

- Context -> Muestra.
- State -> MuestraEstado.
- ConcreteState ->EstadoVerificada, EstadoNoVerificada.

Observer: ZonaDeCobertura

Decidimos que era necesario el uso del patrón Observer en el momento que detectamos que las Organizaciones requieren conocer cuando se valida o se agrega una nueva muestra a una de sus zonas de interés para poder hacer una función externa.

- Subject -> ObservadorOrg
- ConcreteSubject -> Organizacion
- Observer -> ObservableOrg
- ConcreteObserver -> ZonasDeCobertura

Observer: Muestra

Decidimos que era necesario el uso del patrón Observer en el momento que detectamos que las ZonasDeCobertura requieren conocer cuando se valida una nueva muestra.

- Subject -> ObservableZona
- ConcreteSubject -> Muestra
- Observer -> ObservadorZona
- ConcreteObserver -> ZonasDeCobertura

Aclaraciones

- Elegimos utilizar el enum TipoDeOpinion, para definir todos los casos de opiniones posibles. Esta implementación en ciertos casos, permitirá la utilización de un tipoDeOpinion incorrecta, pero comprendemos que la idea de incluir enums dentro de otro sería una mala práctica.
- Luego de las correcciones se decidio quitar los distintos administradores que teniamos. (AdministradoUsuario, AdministradorMuestra, AdministradorZona) ya que estos eran innecesarios, repetían código que simplemente se podía ejecutar llamando a cada usuario, muestra o zona desde la web.