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 al administrador de muestras, usuarios y zonas correspondientemente. Además, antes de agregar una opinión a un usuario, verifica si pudo ser agregada la opinión a la muestra en cuestión.

web.administradorMuestra:
Este es el encargado de interactuar con la web. Es quien conoce las muestras del sistema y quien itera sobre ellas y ejecuta los métodos adecuados sobre cada una.

web.administradorUsuario:
Este es el encargado de interactuar con la web. Es quien conoce los usuarios del sistema y quien itera sobre los usuarios y ejecuta los métodos adecuados sobre cada uno de estos. El usuario contiene un registro de Muestras las cuales subió y un registro de las Opiniones hechas por el mismo. El mismo también es capaz de actualizar su estado (EstadoBasico, EstadoExperto y EstadoValidado) según corresponda.

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

- AdministradorZona: se encarga de agregar zonas y verifica si esta nueva zona solapa con las ya existentes, para agregarla a la lista de zonas que solapan de cada zona. Agrega muestras a las zonas correspondientes según su ubicación. Recibe mensajes de una muestra validada para poder informar a las zonas correspondientes.
- ZonaDeCobertura: lleva una lista de las muestras que le corresponden y cada vez que se agrega una notifica a las organizaciones correspondientes. Sabe si una muestra se encuentra dentro de la zona. Sabe si solapa con otra zona y lleva una lista de muestras que solapan. Puede registrar y desregistrar observadores para luego notificar cuando sea correspondiente.
- OrganizacionNoGubernamental: son observadores que realizan funciones externas según la notificación que reciban.
- FuncionDeImpresion: es una función externa que realiza un evento en específico.
- Interfaces: Observador, Observable, FuncionExterna.
- Enums: TipoDeOrganizacion.

web.extras:
En este paquete se encuentran las clases Opinion y Ubicacion.

- 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.
- 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** -> AdministradorDeMuestras
- **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: OrganizacionNoGubernamental

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 -> el trabajo menciona que hay pero no especifica cuáles
- Context -> OrganizacionNoGubernamental

State: EstadoMuestra

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 -> EstadoMuestra.
- ConcreteState ->EstadoVerificada, EstadoNoVerificada.

Observer: ZonasDeCobertura

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

(No cumple el patrón de diseño estándar)

- Subject -> OrganizacionNoGubernamental
- ConcreteSubject -> OrganizacionNoGubernamental
- Observer -> ZonasDeCobertura
- 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.