TP Integrador POO2

A la caza de las Vinchucas

12.06.2022

Integrantes

Leonardo Criado - leonardoac31@gmail.com

María de los Ángeles Gattone - mariadelosangelesgattone@gmail.com

Héctor Villavicencio - fvillavicencio2@gmail.com

Descripción general

En el siguiente documento se documentaran los patrones y decisiones de diseño tomadas para la confección del Trabajo Práctico Integrador “A la caza de las vinchucas”.

Lineamientos generales

Decidimos partir de una clase principal que jugará el rol de main llamada AplicacionWeb.

De la misma colabora con colecciones de las siguientes clases que modelan los requerimientos del trabajo práctico:

- Usuario
- Muestra
- ZonaDeCobertura
- Organizacion
- Buscador

A su vez, estas interactúan con distintas clases que serán descriptas en el siguiente detalle

Usuario

Decidimos Utilizar el patrón State en usuario ya que entendemos que usuario cambió sus métodos según el rango al que pertenece, además que cuando llega a una cierta cantidad de requisitos para cambiar el rango lo haga de manera automática, eso pasa cuando cambia de “básico” a “experto” y de “experto” a “básico” cambia de manera random, también está el rango “siempre experto” que se crea directamente con esa categoría y NUNCA cambia.

Sus roles serían:

Context es el “Usuario”

State es “EstadoUsuario”

ConcreteState son “Ubasico”, “UExperto” y “UExpertoPorSiempre”

Muestra

La clase Muestra modela una muestra. Posee dos estados de comportamiento diferente modelados por las clases:

- ValidacionMuestraBasica: Aplicado cuando la muestra no posee revisiones de usuarios expertos
- ValidacionMuestraExperto: Aplicando cuando la muestra fue revisada por lo menos un expertos

Se utilizó un patrón **State**:

Sus roles serían:

Context es el “Muestra”

State es “MuestraState”

ConcreteState son “ValidacionMuestraBasica”, y “ValidacionMuestraExperto”

La opinión de un usuario:

La clase **Opinion** modela la opinión de un usuario emitida en un momento particular, con una clasificación específica para la muestra.

La clasificación de las muestras

La clasificación de las muestras se representó mediante la interfaz ClasificacionMuestra, que se implementó en tres enum:

- EspecieVinchuca: para clasificaciones de vinchucas
- NoEsVinchuca: para clasificaciones que no fueran vinchuca
- EspecieNoDefinida: que es la clasificación retornada en el estado de una muestra cuya clasificación no está definida porque los votos empatan.

Zona de cobertura

Esta clase modela las distintas zonas de cobertura y su interacción con las muestras, otras zonas de cobertura y las organizaciones.

Además colabora con la clase Ubicacion que modela un punto geográfico que puede calcular la distancia en Kms a otras ubicaciones que a su vez colabora con DistanciaHaversine para realizar dichos cálculos.

Se utilizó el patrón Observer, donde ella misma cumple el rol de **Observable** y avisa a Organizacion (**Observer**) por medio de la interfaz ObserverZona (**Observer Concreto**) cuando corresponda según si la muestra que se modifica está o no dentro de su cobertura.

Organizacion

Las organizaciones, como se menciona en Zonas De Cobertura, son un **Observer** concreto de esa clase.
La misma guarda en variables de instancia las acciones que debe realizar según la actualización que llegue de las muestras que se registran en las zonas a las que esté registrada

Buscador

Este requerimiento lo pensamos como un patrón State que se combina con un patrón Composite adaptado.

La clase Buscador (**Contexto**), colabora por medio de la interfaz BuscarPor (**Strategy**) con las distintas maneras posibles de realizar una búsqueda (**Concret Strategy**).

A su vez pensamos en utilizar un template method para las búsquedas por fecha que colabora con OperadorRealacional.

Por otro lado, tambien implementando BuscarPor, creamos una clase abstracta llamada OperadorLogico cuyas subclases son And y Or. En este caso OperadorLogico cumple el rol de **Composite**, BuscarPor el rol de **Component** y el resto de estrategias de búsqueda son las **Leaf.**

Todo esto es con el fin de que sea fácilmente extensible en caso de incorporar nuevas maneras de búsqueda en el requerimiento original.