Trabajo Practico Final Objetos II – 2022

Integrantes:

- **Nicolas Carrizo – carrizonicolas97@gmail.com**
- **Luis Salas – sc.luis93@gmail.com**

Explicación del Trabajo Practico Integrador:

Nuestra aplicación fue realizada en Java en donde realizamos una serie de clases para el desarrollo Backend de nuestra aplicación web que su objetivo es poder controlar la plaga de Vinchucas en territorio argentino. Tenemos 5 clases principales en este desarrollo que son: Muestra, Usuario, Opinión, ZonaDeCobertura y Organización.

¿Cómo un Usuario puede dar una opinión?

En la clase Usuario tenemos el método **opinarMuestra** que recibe dos parámetros la muestra y la especie esta luego va hacia la clase Muestra que con el método **recibirOpinion** que recibe dos parámetros el usuario que va a dar la opinión y la especie en cuestión. Este método **recibirOpinion** lo primero que va a hacer es validar si la muestra ya no fue verificada anteriormente, si fue verificada ya no puede dar una opinión el usuario y lo segundo que hace es validar que este usuario no realizara una opinión con anterioridad, si este usuario ya opino no puede volver a opinar.

Una vez que se validara esto y cumpla con los requisitos el usuario va a opinar. Tenemos dos tipos de usuario representados por dos clases **UsuarioBasico** y **UsuarioExperto,** dependiendo que tipo de usuario es va dar su opinión. Esto fue representado con el método **opinar** que se encuentra tanto en **UsuarioBasico** como en **UsuarioExperto**. Si es **UsuarioBasico** lo primero que va a hacer es validar si no opino un experto con anterioridad. Luego lo que voy a hacer es verificar el rango de ese Usuario ya que puede cambiar a **UsuarioExperto** o seguir en el mismo rango.

Una vez realizado esto creo la nueva opinión, dentro de la clase **Opinion** en su constructor llamo al método **agregarOpinion** que se encuentra en la clase de Muestra de esta manera nos aseguramos que siempre que sea crea una nueva opinión esta termine dentro de una muestra. El método **agregarOpinion** lo que va a chequear primero es si la opinión que voy a agregar va a hacer un cambio de estado de la muestra tenemos dos tipos de estados representados por las clases **EstadoEnVotacion** y **EstadoVerificada.** Si la opinión es de un experto y la especie que decidió ya existe previamente en la lista y fue opinada por un experto también entonces la muestra cambiara de estado a **EstadoVerificada** y esa muestra quedara verificada. Si la opinión es de un **UsuarioBasico** el estado sigue igual en **EstadoEnVotacion.**

Datos particulares de implementación:

La clase Foto utilizada como atributo en Muestra, solo es representada por un link que nos llevaría a dicha foto, de esta manera un usuario final no debería subir su foto al sistema, sino que pasaría un link de referencia.

La clase Zona fue creada con la finalidad de que al momento de crear una muestra saber a qué zonas va a pertenecer esa muestra y así luego poder notificar su creación o verificación. Dentro del constructor de ZonaDeCobertura, cada vez que se instancie una nueva ZonaDeCobertura esta se guardara en la lista de Zonas.

La clase Muestra dentro de su constructor llama al método nuevaMuestra de BusquedaDeMuestra para que cada vez que se crea una nueva instancia de Muestra, esta se agregue a la lista de Muestras de BusquedaDeMuestra.

En la clase Ubicación para calcular la distancia entre dos ubicaciones se utilizó MATH.POW() y MATH.SQRT(). Adjunto video de donde se sacó la información de cómo se utiliza: **https://www.youtube.com/watch?v=NKVBGcfh1Jg&ab_channel=Profe.Casi**

Patrones de diseño:

Patron Observer:

Notificación a zonas de la creación/verificación de una nueva Muestra:

- Se encuentra el patrón Observer entre las clases: Muestra y ZonaDeCobertura. En donde Muestra seria nuestro Observable o (Subject/SujetoConcreto) y ZonaDeCobertura nuestro ConcreteObserver. En este observer no tenemos métodos attach ni detach ya que la ZonaDeCobertura no se agrega manualmente, sino que estas se agregan dependiendo de si la muestra está dentro de su rango o no, esto se puede ver en el constructor de Muestra. (Por esto mismo agregamos el objeto Zonas).

Notificación a organización de la creación/verificacion de una nueva Muestra:

- Se encuentra otro Observer entre las clases: ZonaDeCobertura y Organización. En donde ZonaDeCobertura es nuestro Observable o SujetoConcreto y Organización nuestro ConcreteObserver.

- ZonaDeCobertura tiene el método registrarA que recibe una Organización, este sería nuestro método attach.

- Organización tiene los métodos notificarCreacion y notificarVerificacion que son nuestros métodos notifys.

Patron State:

- El patrón State se puede encontrar entre las clases: Muestra que sería nuestra clase **Context** que contiene un atributo estadoDeMuestra que contiene el estado actual de la Muestra. Luego está la interfaz EstadoMuestra que es nuestra clase **State** que define una serie de métodos que son implementados por EstadoEnVotacion y EstadoVerificada que representan nuestros **ConcreteState** que pueden devolver la especie más opinada para el método resultadoActual que se encuentra en Muestra.

- Se encuentra otro patrón State entre las clases: Usuario que sería nuestra clase Context tenemos el atributo tipoDeRango que contiene el rango actual del usuario. Luego la interfaz UsuarioRango que sería nuestra clase State esta define los comportamientos que van a implementar las clases UsuarioBasico y UsuarioExperto estas representarían las clases ConcreteState.