Universidad Nacional de Quilmes

Trabajo Práctico Integrador

Programación Orientada a Objetos II - 2025s1

Integrantes:

- Díaz Marcos | ma.nahuel.d@gmail.com
- Mendoza Tomas | tm1453766@gmail.com
- Monteros Dario | monterosrubendario@gmail.com

Decisiones de diseño

Se definieron varias pautas para el desarrollo en equipo de este trabajo. En primera instancia se aclaró que todos las clases y métodos iban a estar en español y se documentaría la mayoría de clases con Javadoc.
En segunda instancia, todos trabajaríamos en una rama distinta del main del repositorio de Git para no mezclar o entorpecer a los demás. Para esto, todos crearían una interfaz del objeto que desarrollaría otro integrante de ser necesario durante su desarrollo y fusionaríamos cada rama.

Detalles de la implementación

Muestras:

- Atributos: Las muestras conocen el id de su publicador, la fotografía de la muestra, su ubicación, el historial de opiniones, la sistema donde cargarse, la fecha de su creación y de la última vez que se agregó una opinión a la misma y también posee un estado. Este último se explica en patrones de diseño, pero principalmente son quienes guardan las opiniones actuales y calculan el resultado actual de la muestra.
- Se representó como un String el atributo de fotografía de la clase Muestra porque representa las rutas de directorios. “C:/home/user/…”

Usuarios:

- Atributos: Los usuarios conocen su id, las muestras que publicó, a la base de datos donde se almacenan las muestras, las opiniones que realizó sobre las muestras y su rango.
- Funcionalidades: el usuario puede publicar muestras y opinar sobre muestras. La segunda funcionalidad, depende del rango del usuario y del nivel de muestra, si no cumple la condición, la muestra arroja una excepción de tipo: SinAccesoAlMuestraException.
- Rangos: al realizar mínimo 10 publicaciones y opinar sobre 20 muestras, el rango asignado del usuario es Experto. El rango experto permite opinar sobre muestras de nivel experto. En cambio, el rango básico, solo le permite opinar al usuario sobre muestras libres. Además, se incorpora el rango especialista, que posee las mismas características que el rango experto, con el plus de

que nunca pierde su rango, osea, no baja a básico aunque deje de cumplir las condiciones requeridas

Zona de cobertura:

- Atributos: Las zonas conocen su epicentro y un radio en kilómetro que determina el área geográfica que abarca. A Partir del epicentro, se puede determinar si con otra zona se solapan o si una ubicación cae dentro de la cobertura. Además, cada zona cuenta con un nombre que las identifica, las muestras que se ubican dentro de la zona y las organizaciones interesadas en recibir notificación de cuándo se ingresó una nueva muestra o si se verificó una muestra.
- Funcionalidades: A cada zona se le puede pedir su nombre, su epicentro y el radio que abarca. Estos últimos atributos se utilizan para determinar si se le debe cargar una muestra a la zona, y para definir si corresponde notificar que se ha verificado o registrado una muestra; con el fin de enviarle información a las organizaciones registradas. Por último, las organizaciones se pueden registrar o desregistrarse de una zona.
- Factory: La creación de zonas de cobertura se realiza a través de un factory. Este no solo encapsula la lógica de la creación de una zona, sino que también registra automáticamente las zonas generadas. Esta funcionalidad es utilizada por el sistema, para notificar a las zonas que corresponda si se registró o verificó una muestra.

Sistema:

- Atributos: El sistema cuenta con dos listas: una con el registro de todas las muestras cargadas, y la otra para registrar todas las zonas de cobertura.
- Funcionalidades: Dispone con dos métodos de agregación: uno para registrar nuevas muestras y otro para registrar zonas de cobertura. El registro de las muestras permite realizar búsquedas por medio de FiltroMuestra. La lista de zonas, por su parte, nos permite identificar a qué zonas pertenece una muestra, y notificar a las zonas de un registro o verificación de una muestra

Filtros:

- Implementación: Para que el sistema pueda filtrar por muestra, existen dos formas:
    - Filtro compuesto: permite combinar distintos tipos de filtros basicos utilizando logica booleana:

        - FiltroAnd: La muestra debe cumplir todos filtros
        - FiltroOr: La muestra debe cumplir al menos uno de los filtro

      Ambos filtros tienen en su implementación una list<IFiltroMuestra> y delega la verificacion a los filtros contenidos.
    - Filtro basico: Un filtro basico determina si una muestra cumple una condicion especifica. Los tipos de filtros basicos implementados son:
        - FiltroPorFechaDeCreacion,
        - FiltroUltimaFechaDeModificacion; para los dos filtros mencionados, se implemento tres tipos de estrategia para la verificación de una fecha,
        - FiltroPorNivelDeVerificacion; para este filtro se implemento dos estrategias que sirve para verificar si la muestra es votada o verificada, se utilizo este enfoque para delegar la responsabilidad, y
        - FiltroPorTipoDeInsecto.

    Para los filtros de fecha (FechaDeModificacion, UltimaFechaDeModificacion), se aplicó el patrón Strategy, mediante el cual se encapsula las distintas formas de comparar fechas.
    Para el filtro de nivel de verificación, también se aplicó el patrón Strategy, definiendo estrategias que evalúan si una muestra es verificada o simplemente votada.

Patrones de diseño

Como patrones se modelaron:
State respecto a las Muestras, donde el contexto serían las mismas muestras y el estado es un atributo que cada muestra posee. Esta misma responde a 3 estados concretos: MuestraLibre, MuestraExperto y MuestraVerificada. Cuando una muestra se crea su estado es MuestraLibre, tras recibir la opinión de un experto muta a MuestraExperto, y si la misma recibe 2 opiniones de expertos, cambia a MuestraVerificada.
Se modeló de esta manera ya que los usuarios tienen distintos comportamientos dependiendo de si la muestra está verificada o del rango de los mismos usuarios. Además hace más fácil la transición entre estados, porque en las muestras en las que opinaron usuarios básicos, esas opiniones se pierden en el conteo cuando el estado cambia (No se pierde su registro, sólo su conteo para el resultado actual).

Otro state que se modeló es el de los usuarios, siendo que estos conocen un rango que designa qué tipo de opinión van a dar en una muestra. El rango tiene tres estados: UsuarioBasico, UsuarioExperto o UsuarioEspecialista. Este mismo cambia según el usuario (Contexto) dependiendo de cuántas muestras hayan publicado y cuántas opiniones dieron en otras muestras.

Factory respecto a las zonas de cobertura, donde se utiliza la clase ZonaDeCoberturaFactory para crear y controlar las instancias de ZonaDeCoberturas y pasar la misma a una sistema para que las conozca a todas.

Observer:
se definió una de cadena de responsabilidad, para la verificación o carga de una muestra:
Las muestras funcionan como sujetos/observados por el sistema que conocen. Estas, al recibir la notificación de que están verificadas, realizan su propia operación. La misma consiste en filtrar las zonas de cobertura que conoce (que funcionan de observadores) por aquellas a las que pertenece la muestra que se verificó, y de esa lista avisar que la muestra fue verificada.
Las zonas de cobertura, a su vez, notifica a las organizaciones que están interesadas en ellas sobre la verificación y también avisan en qué zona fue.
Por último, las organizaciones que reciben dicha notificación delegan a la funcionalidad externa que tienen asignada, avisando sobre la muestra, la zona y la organización donde se notificó el evento.
Strategy: Mencionado anteriormente en la explicación de filtro, existen dos Strategy para los filtros basicos:

- EstrategiaComparacionPorFecha: Esta clase define que las estrategias deben tener una fecha por la cual se realiza la comparacion y un metodo abstracto para comparar con una fecha recibida.
- EstrategiaVerificacion: Esta clase define que las estrategias que hereden deban definir de que manera se verifica una muestra.
- Estrategias concretas de comparación por fecha: Se definió tres formas de comparar fechas, antes, después o igual a una fecha dada.
- Estrategias concretas de verificacion: Se definio que una estrategia indica si la muestra dada es verificada y que la otra indica si la muestra dada es votada.

Composite: Mencionado anteriormente en la explicación de filtros, además de los FiltroBasico también se implementó un tipo de filtro llamado FiltroCompuesto. Este filtro se basa en el patrón Composite, el cual permite tratar objetos individuales (FiltroBasico) y composiciones de objetos (FiltroCompuesto). Los filtros compuestos contienen una lista de objetos que implementan la interfaz IFiltroMuestra, y delegan la verificación a esos elementos.
El cliente que utiliza este composite es el Sistema, el cual puede aplicar un filtro sin necesidad de conocer si se trata de un filtro simple o una combinación de varios.