Trabajo Práctico Final
Ciencia Participativa y Juegos

Programación orientada a Objetos II

Estudiantes:

Manuel Vladimir Dobrotinich — Email: Manueldobrotinich@gmail.com
Martin Alejandro Boidi — Email: Boidi.martinalejandro@gmail.com
Elian Camilo Alejandro Borda — Email: Eliancamiloalejandro@gmail.com

Comisión 2

Documentación del tp final

Todas las clases.

Proyecto

Desafío
Días Semana
Dificultad
Entre Fechas
IRestricciónTemporal
RestriccionMixta

DesafioDeUsuario
EstadoAceptado
EstadoFinalizado
IEstadoDeDesafio

ConjuncionDeFiltros
DisyuncionDeFiltros
Filtrable
FiltroDeExclusionCategorias
FiltroDeInclusionCategorias
FiltroSimple
FiltroTextoEnNombre
NegacionDeFiltro

Sistema
FormaDeRecomedacionFavorita
FormaDeRecomendacionPreferencia
IFormaDeRecomendacion

Usuario
Gusto
Preferencia

Muestra
Coordenada
Distancia
Muestra

Clases principales

Proyecto

Modela los proyectos de ciencia participativa. Cada proyecto conoce a sus **Usuarios** suscriptos y a los **Desafíos** que pueden ser completados por los usuarios para su desarrollo, así como las **muestras** que son recolectadas por los mismos para su completamiento. Cada proyecto regula que sus usuarios solo puedan aceptar **desafíos** que pertenezcan a sí mismos.

Usuario

Usuario modela a los usuarios que participan de los **proyectos**, y que llevan a cabo **desafíos** para colaborar con los mismos. Un usuario puede aceptar un **desafío** sólo si el mismo pertenece a alguno de los **proyectos** en los que participa, a los cuales se puede suscribir. Asimismo, el usuario puede calificar a los desafíos en los que participa y pedirle al **sistema** que le recomiende desafíos de acuerdo a determinadas **formas de recomendación**, las cuales están modeladas mediante un patrón de diseño **Strategy(**ver más adelante) y se basan en sus propias **preferencias.** Por último, el usuario puede recolectar **muestras** para completar con sus desafíos iniciados.

Sistema

Modela una clase cuyo propósito es recomendar desafíos delegando la implementación de la recomendación a otras clases. Las formas de recomendación que existen son de **FormaDeRecomendacionPreferencia** y **FormaDeRecomendacionFavorita.** Ambas implementan la interfaz **IFormaDeRecomendacion.**
Esta forma de diseño agiliza la extensión de posibles nuevas formas de recomendación.

Desafío

Modela una clase cuyo propósito es representar cada desafío del proyecto del cual es asignado.
Los desafíos para ser superados deben llegar a un límite de muestras pasadas por parámetro cuando este es creado, cada muestra debe estar dentro del **Área** y cumplir con la**_RestriccionTemporal** que requiere el desafío, las cuales también serán pasadas por parámetro.

Para poder cumplir con las precondiciones que nos pedía la Restricción temporal de cada desafío decidimos utilizar el Patrón de diseño **Composite.**

DesafioDeUsuario

Desafío de usuario modela cada desafío que realiza un **usuario** una vez que este último lo aprueba. Esto permite modificar el estado interno de un desafío para cada usuario sin sobrecargar a esta clase con la información de cada uno. En la medida que un usuario cumple o no con los requerimientos de un desafío de usuario el mismo puede modificar su comportamiento interno el de acuerdo a dos estados, **EstadoAceptado**_y **EstadoFinalizado.**

Decidimos usar el Patrón de diseño **State** para modelar los diferentes estados por los que pasa cada desafío del usuario para que este mismo pueda modificar su comportamiento en tiempo de ejecución.

EstadoAceptado
Nos indica cual el desafío del usuario está en curso, si el desafío del usuario se encuentra en este estado, el usuario puede reducir la cantidad de muestras que el desafío obliga para poder completarla.

EstadoFinalizado
Nos indica cuando el desafío del usuario finaliza, es decir recoge todas las muestras que necesitaba para completar el mismo. Este estado evita que el usuario pueda alterar el desafío cada vez que el mismo recolecte una muestra. En este estado, la cantidad de muestras a recolectar dado por el desafío se encuentra en cero.

Filtros
Los **filtros de búsqueda** implementan la interfaz **Filtrable** y permiten buscar dentro de un grupo de proyectos aquellos que cumplan con un determinado criterio de búsqueda. Este último es implementado por cada uno de los **filtros** modelados. Esta implementación es modelada por medio del patrón de diseño **Composite** que nos permite unir distintos filtros mediante filtros recursivos.

Implementación de patrones de diseño.

Formas de recomendación
Para implementar las formas de recomendación decidimos utilizar el patrón **strategy**. El mismo es conveniente ya que permite al usuario modificar la forma de recomendación que desea intercambiando **Estrategias** de

recomendación. Dentro del esquema del patrón, la clase **Sistema** cumple el rol de Contexto, el cual brinda al usuario una interfaz que le permite modificar la estrategia de recomendación y realizar la búsqueda de acuerdo a la misma. Por otro lado, la interfaz **IFormaDeRecomendacion** es la que contiene el protocolo de algoritmo que debe implementar cada **Estrategia** de manera independiente. Este último rol es cumplido por las clases **FormaDeRecomendacionPreferencia** y **FormaDeRecomendacionFavorita,** cada una cumpliendo con las restricciones solicitadas. Este patrón permite que fácilmente se puedan incorporar nuevas **estrategias** de búsqueda, sin modificar la implementación vigente y solo agregando nuevas clases que implementen el protocolo dado.

Restricciones Temporales

Para las restricciones temporales decidimos usar el patrón de diseño composite,
cada restricción cumplen con el rol de las hojas (**EntreSemana** , **DiasDeSemana** y **FinDesemana**) , estas clases implementan la interfaz **IRestriccionTemporal** la cual es el componente y tiene dos métodos fundamentales para cumplir con la restricción temporal del desafío, por un lado el mensaje validar el cual chequea que la fecha de la muestra cumpla con la fecha de la restricción que corresponde el desafío. Por otro lado la interfaz implementa el método estaAbierta el cual nos indica si la fecha del desafío que nos dan está cerrada definitivamente o si puede volver a estar vigente.
La **RestriccionMixta** , cumple con el rol del composite ya que esta clase nos permite combinar las distintas hojas y validar si la fecha de la muestra pasada por parámetro cumple con las distintas restricciones que podemos convinar , como por ejemplo si nos dan una restricción la cual nos pide que sean todos los fines de semana del mes de Diciembre.

Filtros de búsqueda

Para implementar los filtros de búsqueda utilizamos también el patrón de diseño **Composite**. El mismo nos permite fácilmente realizar búsquedas ya que incorpora la posibilidad de componer búsquedas con distintos filtros mediante conectores lógicos. Para implementar el patrón, la interfaz Filtrable cumple el rol de **Component,** indicando el protocolo que cada clase de la jerarquía debe implementar para realizar las búsquedas. La clase abstracta **FiltroSimple** incorpora la estructura que deben cumplir las **hojas** del **Composite,** ya que todas deben filtrar proyectos de acuerdo a un criterio independiente. Por otro lado, las clases **ConjuncionDeFiltros, DisyuncionDeFiltros y NegacionDeFiltros** cumplen el rol de **Composite,** ya que incorporan componentes para ejecutar el protocolo de manera recursiva, permitiendo unir a sus componentes de acuerdo al comportamiento de las compuertas lógicas. Encontramos innecesario en este caso implementar un comportamiento que permitiera agregar nuevos filtros a los compuestos, ya

que entendemos que las formas lógicas de unir filtros solo actúan siempre a partir de dos filtros, pudiendo ser cada uno de los mismos también compuestos para realizar búsquedas aún más complejas, pero siempre cumpliendo con este mismo criterio básico.