TP Final Objetos 2
2022

Gabriel Moreyra gabrielmoreyra97@gmail.com
Lucio Jara luciojara8@gmail.com
Nicolas Hubczyk nico.hubczyk@gmail.com

CienciaParticipativa:

La clase CienciaParticipativa funciona como "punto de entrada al sistema" ya que a través de esa clase es que podemos dar de alta los proyectos y los usuarios del sistema.

Usuario:

Los usuarios se suscriben a proyectos y reciben Desafios de estos. Los Desafios en lugar de guardarlos de por sí, tiene como colaborador interno un DesafioUsuario que contiene al Desafio y todos los datos que conciernen a la relación entre el Usuario y dicho Desafio.

DesafioUsuario:

DesafioUsuario es una clase contenida por un Usuario, contiene un Desafio y los elementos que relacionan al Usuario con el Desafio. Cuando un Desafio es agregado a un Usuario se guarda como DesafioNoAceptado, una vez aceptado pasa a DesafioActivo y al terminarse se convierte en un DesafioCompletado. Estos 3 estados son una implementación del patrón **State**.

-Patrón State:

DesafioUsuario es el Context en el patrón, mientras que EstadoDesafio es el State. A medida que DesafioUsuario cambia de estado, cambia su variable local de EstadoDesafio por un diferente Concrete State: DesafioNoAceptado, DesafioActivo, DesafioCompletado.

Cuando DesafioUsuario quiere realizar una funcionalidad, llama a la implementación de sus States. Por ejemplo cuando se pregunta si un desafio fue completado por el usuario se llama a la función fueCompletado() en DeafioUsuario, luego este llama a fueCompletado() de su estado y devuelve ese boolean.

Recomendación:

Las recomendaciones son, básicamente, estrategias de búsqueda de Desafíos en Proyectos para un Usuario.
Los Usuarios poseen una Recomendación, pues son quienes eligen cuál implementar. Cuando un Proyecto quiere recomendar Desafíos pide a los Usuarios su Recomendación e implementa la búsqueda. Esta solución es una implementación del patrón **Strategy**.

-Patrón Strategy:

El Usuario y Proyecto son el Context, mientras que la clase abstracta Recomendación es el Strategy. Cada clase concreta hija de Recomendación: RecomendacionFavoritos y RecomendacionPreferencia; son los Concrete Strategy.

Consideramos que tanto Usuario como Proyecto como Context debido a que si bien, el usuario elige y almacena una referencia a la Recomendacion, el Proyecto tiene referencias indirectamente a través de los usuarios suscritos y es quien implementa la función de Recomendacion.

Filtros:

Los distintos filtros pueden discernir si un proyecto cumple o no ciertas condiciones, dando un booleano como resultado. Y pueden filtrar de una lista los proyectos que cumplen llamando a la función anterior.
Son destacables los filtros And, Or y Not, que poseen otros filtros como colaboradores internos y operan sobre estos. El hecho de que tengan como colaboradores otros objetos tipados de una clase padre hace que el patrón sea **Composite**.

-Patrón Composite:

Los filtros And tienen 2 colaboradores. Para que un proyecto cumpla el filtro tiene que cumplir ambos filtros colaboradores. En el filtro Or tiene que cumplir al menos uno. Y en el filtro Not no tiene que cumplirlo.

Aclaración:
En el enunciado se pide que se puedan incluir y excluir ciertas categorías como ítems distintos. En lugar de hacer una diferenciación en el diseño, se optó por dejar un simple FiltroCategoría y si se quiere que excluya se puede dejar como colaborador interno de un filtro Not.

Puntuable:

La interfaz Puntuable fue creada debido a la similitud de los campos de PreferenciasUsuario y Desafio. Útil para la extensibilidad del código gracias al polimorfismo.

Restricciones Temporales:

Con permiso del profesor nos tomamos la libertad de, en lugar de permitir entre día de semana/ fin de semana, permitir que sea unos días personalizados. Creamos la clase RestriccionTemporalDiaSemana y contiene una lista de los días de la semana permitidos, de Lunes a Domingo, de tipo DayOfWeek.

Aclaraciones tras correcciones:

La relación contieneTag de FiltroCategoria a Categorías fue cambiado en el UML a categoriaAFiltrar. Es la categoría con la que se filtra, si tiene la categoría pasa el filtro.

Recomendaciones ahora es una interfaz en lugar de una clase abstracta, debido a que tras la corrección quedó con solo un método abstracto en vez de métodos concretos y abstractos.

La clase EstadoDeDesafio la dejamos como clase abstracta debido a que posee colaborador interno.