Trabajo Práctico Final

Ciencia Participativa y Juegos

Alumnos:

Bragoni, Pablo. pablo.bragoni@gmail.com

Laborde, Walter. walter.laborde@hotmail.com

Mendez, Ornella. mendez.ornella.p@gmail.com

Profesores: Torres, Diego.

Cano, Diego.

Butti, Matias.

Año: 2° cuatrimestre 2022

Link al repositorio: https://github.com/PeMendez/Unq-Po2-TpFinal

Universidad Nacional de Quilmes
Departamento de Ciencia y Tecnología
Programación con Objetos II

A continuación se detallan los patrones de diseños elegidos para la correcta implementación del trabajo práctico.

- **Patrón State (Estados de los desafíos)**

Para los diferentes estados que pueden tomar los desafíos, decidimos utilizar el patrón State. Esta decisión nos permite definir el comportamiento específico que tiene que tener cada desafío en cada uno de sus estados y tener una transición explícita entre diferentes estados. Al utilizar este patrón nos aseguramos de no encapsular toda esa lógica en una sola clase y, en el caso de que sea necesario agregar nuevos estados se podrán agregar subclases concretas que dependan del State.

Según el libro de Gamma, los roles de las clases son:

```
DesafioUsuario         -> Context
EstadoDesafio          -> State
EstadoPendiente        -> ConcreteState
EstadoEnCurso          -> ConcreteState
EstadoCompleto         -> ConcreteState
```

- **Patrón Strategy (Recomendación de desafíos)**

Para que un usuario pueda elegir la forma en la que se le recomienden desafíos, decidimos utilizar el patrón Strategy. De este modo, el usuario va a poder seleccionar, al principio y a lo largo de la vida del proyecto, la manera en la que quiera esa recomendación. Internamente, se encapsulan los diferentes algoritmos para la recomendación uno en cada clase concreta, eliminando de esta forma los métodos condicionales difíciles de mantener y extender.

Según el libro de Gamma, los roles de las clases son:

```
Perfil               -> Context
RecomendadorDesafios -> Strategy
PreferenciaDeJuego   -> ConcreteStrategy
Favorito             -> ConcreteStrategy
```

- **Patrón Composite (Restricción temporal del desafío)**

Para la restricción temporal de los desafíos decidimos utilizar el patrón composite. Esto le permite al usuario poder seleccionar restricciones simples o combinadas desde su interfaz. En el backend, esta o estas condiciones que el usuario establezca para el desafío, serán tratadas de manera uniforme. Aplicando este patrón en la restricción temporal, nos aseguramos que de ser necesario agregar un nuevo tipo de restricción temporal, alcance con crear una nueva clase que herede el protocolo de la clase abstracta de la restricción temporal.

Según el libro de Gamma, los roles de las clases son:

```
Desafio           -> Client
TipoDeRestriccion -> Component
DiasHabiles       -> Leaf
FinDeSemana       -> Leaf
RangoDeFechas     -> Leaf
RestriccionMixta  -> Composite
```

- **Patrón Composite (Buscador de desafíos).**

Para el buscador de proyectos usamos el patrón composite. De esta manera, el usuario puede aplicar múltiples criterios de búsqueda. En el backend, estos criterios se irán anidando y será tratado como una única condición por la cual se filtraran los proyectos.
Con el uso de un composite, la búsqueda se vuelve más flexible y abierta a distintos y nuevos criterios.

Según el libro de Gamma, los roles de las clases son:

```
DesafioUsuario                  -> Client
CondicionDeBusqueda             -> Component
ExcluyeCategorias               -> Leaf
IncluyeCategorias               -> Leaf
IncluyeTextoEnElTitulo          -> Leaf
Negacion                        -> Composite
CompuestoBinario(abstract class)  -> Composite
AND                             -> Composite
OR                              -> Composite
```