Análisis y diseño de aplicaciones - TFU 5

Profesores: José Abadie, Sebastián Feirres

Equipo 4

Índice

Introducción

Para el final de la unidad 5, se pidió crear un backend y una API para la aplicación modelada en unidades anteriores de los Juegos Olímpicos. Para esto, es necesario que el código cumpla con los patrones SOLID, así como también con otros patrones de diseño estructurales, creacionales y de comportamiento.
En la aplicación, fue necesario realizar algunos cambios para que satisfaga estas condiciones, modificando algunos de los diagramas creados para ajustarlos a estos requisitos.

Repository

Para la creación del backend y la API se utilizó el patrón repository, el cual separa la capa de acceso a datos de la capa de lógica de negocios. A través del repository, se puede organizar mejor el código, así como también facilitar la creación de nuevos endpoints.

Adapter

Nuestra aplicación tiene 2 roles centrales, un usuario cualquiera o un juez, el cual debe identificarse. Para este proceso de inicio de sesión, el mismo debe ingresar su información de autenticación. Por lo que utilizar el patrón Adapter para el componente de inicio de

sesión permite integrar múltiples métodos de autenticación sin modificar la lógica central de la aplicación.

Este enfoque promueve la flexibilidad y la extensibilidad al aislar la lógica de autenticación en componentes separados. Asegura que agregar nuevos mecanismos de autenticación se pueda hacer sin modificar lo ya existente.

Observer

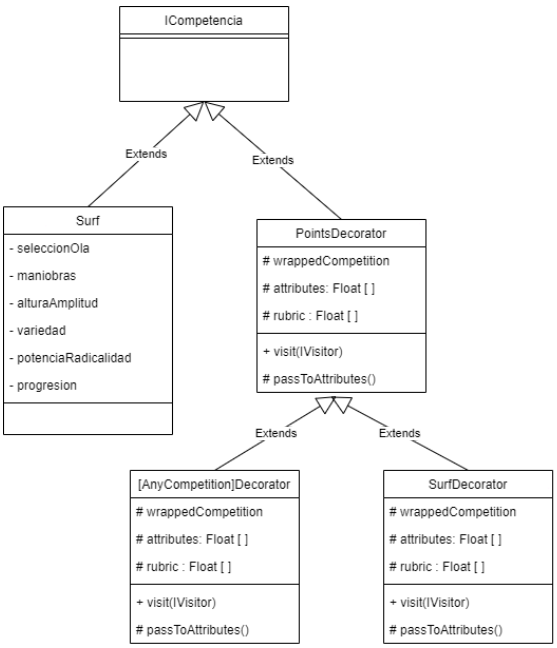
Como nuestra aplicación tiene como funcionalidad destacada la puntuación en vivo, decidimos que la misma debía ser modelada. Por ello, para la puntuación en vivo que debe realizar el juez, se decidió aplicar el patrón Observer. Esto nos permite que los competidores se puedan suscribir a los resultados que irá subiendo el juez, por lo que al subir la puntuación de cada competidor, el competidor será notificado.

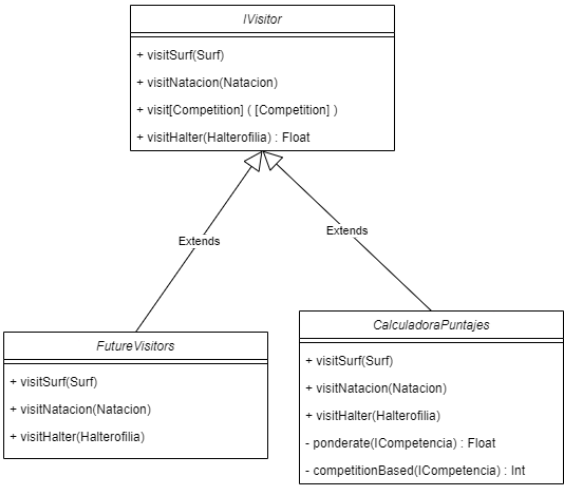
Visitor

El patrón visitor fue aplicado al cálculo de puntajes, debido a que cada disciplina tiene su formato de puntuación. En la implementación, se creó una interfaz que contiene los métodos para calcular los puntajes en cada disciplina, que serán utilizados posteriormente a la hora de obtener los puntajes en la aplicación.

Decorator

Se utilizó el patrón decorator en conjunto con el visitor, para agregar información al momento de devolver los puntajes de las disciplinas. Cuando se realiza una visita con el visitor, el decorador agrega la rúbrica utilizada, así como también los puntajes individuales dentro de cada categoría de puntuación.
El siguiente diagrama de clases ilustra ambos patrones:





State

Las olimpiadas son eventos donde ocurren muchas competencias en simultáneo de diferentes disciplinas y/o categorías, es por ello que el patrón State nos pareció esencial para poder controlar el despliegue y comportamiento de las distintas competencias según su estado. De esta manera, tendremos competencias en vivo, terminadas y competencias que van a ocurrir en un futuro. Estos estados son utilizados no sólo para desplegar los eventos del día en la landing page, sino también para el proceso de realizar puntuaciones de los jueces, ya que es necesario saber el estado de cada competencia para saber si la misma debe ser puntuada.

Facade

Se utilizó el patrón facade para unificar varios contextos dentro del mismo controlador, sin necesidad de que el controlador mantenga referencias a cada contexto. De esta manera, se maneja un único objeto dentro del controlador, lo que hace más fácil de mantener este endpoint.

Testing

Para corroborar que la API y el backend funcionan correctamente, se utilizó Postman para probar los endpoints individualmente. A través de esta herramienta, se puede comprobar

que los controladores funcionan correctamente, así como también los repositorios que se conectan a la base de datos.

Cambios a futuro

En vista de que el equipo no contaba con tiempo suficiente para implementar todos los patrones, se decidió documentar posibles aplicaciones de patrones.

Builder

La aplicación asume que los competidores pueden participar en varias disciplinas, por lo que sería necesario reflejar cada disciplina dentro de la entidad que modela a los competidores. Para esto, el patrón builder es ideal, siendo capaz de instanciar cada competidor con los datos necesarios, y no se precisa modificar el objeto una vez creado para agregar nuevas disciplinas.

Singleton

Los jueces pueden calificar en vivo a los participantes a medida que el evento transcurre. Para asegurarse que no hayan errores en las calificaciones, se puede utilizar un singleton para que haya una única lista de competidores que se despliegue al juez, evitando que hayan puntajes repetidos para un mismo competidor en la misma disciplina.

Adapter

En el evento de que se permita descargar información de la aplicación en distintos formatos, el patrón adapter es ideal para convertir los datos de la base de datos en el formato correcto para descargar. Además, se pueden agregar formatos nuevos para descargar sin alterar el código existente.

Prototype

En conjunto con la funcionalidad anterior, en vez de obtener la información de la base de datos cada vez que se quiere descargar, se puede obtener una vez y generar copias para cada formato, optimizando el funcionamiento de la aplicación.

Link al proyecto

.https://github.com/ManuelM512/UT5_TFU_G4_Olympics_BE