Torneos de Yu-Gi-Oh

Chavely González Acosta C312
José Carlos Pendas Rodriguéz C311
Lázaro David Alba Ajete C311
Max Bengochea Moré C311

December 5, 2023

1 Introducción

2 Análisis y reformulación enriquecedora de los requerimientos funcionales e informacionales del sistema

2.1 Requerimientos funcionales:

1. **Registro de usuarios:** Los usuarios se registran en la aplicación proporcionando su información personal: nombre completo, municipio, provincia, teléfono (opcional) y dirección.
2. **Registro de Decks:** Los jugadores registraran sus decks (mazos de cartas) asociándolos con su perfil, incluyendo detalles como: nombre del deck, cantidad de cartas en el mazo principal, en el mazo alternativo y en el mazo extra, así como su arquetipo.
3. **Creación de torneos:** Los administradores pueden crear torneos especificando: nombre del torneo, fecha de inicio y dirección, un torneo puede ser creado por un solo administrador.
4. **Solicitud de inscripción en torneos:** Los jugadores solicitan inscribirse en los torneos disponibles, asegurándose de que no puedan inscribirse después de la fecha de inicio del torneo, y con esactamente un deck.
5. **Inscripción en torneos:** Los administradores reciben las solicitudes de inscripción de los jugadores a los torneos y se encargan de inscribirlos en los mismos.
6. **Gestión de Rondas y Partidas:** La aplicación administra las rondas de los torneos, asignando aleatoriamente los emparejamientos.
7. **Registro de los resultados de las partidas:** Los administradores registran los resultados de las partidas, asegurando que una partida de Yu-Gi-Oh! es de 3 a ganar 2.
8. **Consultas y Estadísticas**: La aplicación ofrece una interfaz donde los usuarios pueden realizar consultas y obtener estadísticas, así como exportar los resultados a otros formatos. Las consultas a modelarse son las siguientes:
    - Los n jugadores con más decks en su poder(de mayor a menor).
    - Los n jugadores más populares entre los jugadores(de mayor a menor).
    - La provincia/municipio donde es más popular un arquetipo.
    - El campeón de un torneo.
    - Los n jugadores con más victorias (ordenados de mayor a menor).
    - El arquetipo más utilizado en un torneo dado.
    - La cantidad de veces que los arquetipos han sido el arquetipo del campeón en un grupo de torneos(en un intervalo de tiempo).
    - La provincia/municipio con más campeones(en un intervalo de tiempo)
    - Dado un torneo y una ronda, cuáles son los arquetipos más representados(cantidad de jugadores usándolos)
    - Los n arquetipos más utilizados por al menos un jugador en el torneo(de mayor a menor)
9. **Seguridad**: La aplicación implementa autenticación y autorización para asegurarse de que solo los administradores pueden realizar acciones como crear torneos y registrar resultados de partidas. Además, se deben validar los datos de entrada para evitar errores o datos incorrectos en la base de datos.

2.2 Requerimientos informacionales del sistema:

Con el objetivo de validar las peticiones de los usuarios, así como dar respuesta a las consultas de los mismos, es necesario mantener diversas informaciones en una base de datos:

1. **Usuario:** De los usuarios se deben almacenar: un identificador para cada usuario, el nombre completo, el teléfono, la dirección, el municipio y la provincia.
2. **Deck:** De los decks se deben almacenar: un identificador para cada deck distinto, el nombre del deck, la cantidad de cartas en el mazo principal, en el mazo alternativo y en el mazo extra, el arquetipo y el usuario que lo creó.
3. **Torneo:** De los torneos se debe almacenar: un identificador para cada torneo, nombre del torneo, fecha de inicio, dirección y el administrador que lo creó.

4. **Rol:** Para cada uno de los roles que puede desempeñar un usuario en la aplicación se guarda un identificador y el nombre del rol.

5. **Partida:** De las partidas se debe almacenar: el identificador de cada usuario que participa(2 en total), el identificador del torneo al cual pertenece, la fecha exacta en la cual se realiza, el resultado de la partida(solo hay 4 resultados posibles: "2:1", "2:0", "1:2", "0:2") y la ronda del torneo en la cual se realiza.

6. **Solicitud de inscripción en torneos:** Los jugadores solicitan inscribirse en algún torneo disponible y estas solicitudes se almacenan como pendientes hasta que algún administrador las acepte e iscriba al jugador o las rechase. Para esto es necesario almacenar: el identificador del usuario, el identificador del torneo, el identificador del deck con el cual se desea inscribir, la fecha en la cual ralizó la solicitud y el estado de la solicitud. Los posibles estados son: pendiente, aceptado, rechazado.

7. **Registro de usuario:** Para todos los usuarios se debe almacenar el identificador del usuario relacionado con el identificador de cada rol que este desempeña.

3 Solucion Propuesta

Hemos optado por la Arquitectura Onion, presentada por Jeffrey Palermo, debido a su sólido historial de éxito en proyectos de software. Esta elección se basa en los siguientes motivos clave:

1. Principio de Inversión de Dependencias: La Arquitectura Onion se adhiere al Principio de Inversión de Dependencias, promoviendo la independencia entre capas y la flexibilidad del sistema.
2. Modularidad y Separación de Responsabilidades: Nos permite mantener una estructura modular y una clara separación de responsabilidades, esenciales para el desarrollo y mantenimiento eficaz.
3. Testabilidad: Facilita pruebas efectivas, garantizando la calidad y confiabilidad del código en nuestro proyecto.
4. Flexibilidad y Adaptabilidad: Nos permite adaptarnos a cambios futuros y nuevas tecnologías sin afectar significativamente la funcionalidad existente.
5. Colaboración Efectiva: Con múltiples ingenieros trabajando en el proyecto, la Arquitectura Onion establece contratos claros entre capas y componentes, lo que facilita la colaboración.

3.1 Capas Externas (Outer Layers)

YuGiOh.Infrastructure: En esta capa reside nuestra base de datos.
YuGiOh.API: Es la capa más cercana a la interfaz de usuario y proporciona una forma de comunicarse con el núcleo de la aplicación. Esta capa interactúa con la primera capa del "application core"(application services layer).

3.2 Capas Internas

YuGiOh.ApplicationServices (Transport Layer): En esta capa es donde reside la mayor parte de nuestra lógica de negocios. Lleva a cabo las operaciones para convertir A en B, entrada en salida. Específicamente en nuestro proyecto, esta capa se encargará de aspectos como la gestión de emparejamientos, el control de estadísticas, el manejo de algunas reglas del torneo, la gestión de resultados, el control de decks, entre otros.
YuGiOh.ApplicationCore: En esta capa definimos los crud realizables.
YuGiOh.Domain: Es la capa de representación de los objetos de datos de alto nivel que utilizamos. Aquí se definen y encapsulan los conceptos fundamentales y los elementos clave de los torneos, lo que permite una representación fiel y coherente de los datos en la aplicación. Algunos de los aspectos esenciales que se manejan en esta capa incluyen la representación de entidades o conceptos fundamentales de la lógica del proyecto, como jugadores, decks, mazos, partidas y torneos.

4 Resultados Obtenidos

Vacio pk hacen falta imágenes del front.

5 Concusiones y recomendaciones

Den muela amiguitos. Y recuerden k aki va lo de como se terminaria el proyecto. Hasta la proxima amiguitos.

6 Bibliografía

1. Agility The Good, The Hype and The Ugly, B Meyer, Springer
2. Ingeniería de software enfoque práctico 7ed Pressman