



One Card Game

Proyecto final de curso

Curso: DAW

Fecha: 2023-2024

Alumno: Luis Valle Curt

Contenido

INTRODUCCIÓN	3
OBJETIVO	3
JUSTIFICACIÓN	3
ANÁLISIS DE LO EXISTENTE.....	3
APLICACIONES EXISTENTES EN EL MERCADO COMPARABLES	4
REQUISITOS FU.....	4
ANÁLISIS Y DISEÑO:.....	5
DIAGRAMA DE LA ARQUITECTURA.....	5
DIAGRAMA DE CASOS DE USOS.....	7
Login/Register	7
Log Out.....	9
Crear Mazo.....	9
Crear Cartas	10
Borrar Cartas	11
Jugar Carta.....	11
Atacar Carta a Carta	12
Atacar Carta a Líder	13
Turnos de las partidas	14
DIAGRAMA DE CLASES	15
DISEÑO DE DATOS.	18
Resumen	18
CODIFICACIÓN.....	20
ENTORNO DE PROGRAMACIÓN.....	20
Lenguajes y Herramientas.....	21
Aspectos relevantes de la implementación.....	21
Manual de usuario.	22
Introducción	22
REQUISITOS E INSTALACIÓN.....	28
CONCLUSIONES	29
CONCLUSIONES SOBRE EL TRABAJO REALIZADO.....	29
Aprendizajes y Retos.....	30
Evaluación del Proyecto	30
Posibles ampliaciones y mejoras.....	31
BIBLIOGRAFÍA Y WEBGRAFÍA	33

INTRODUCCIÓN

Este documento detalla el proceso seguido de la realización del proyecto desde la fase de conceptualización hasta la implementación final, incluyendo el diseño de la base de datos, la arquitectura del sistema, el desarrollo de las funcionalidades principales, y las estrategias de prueba y despliegue.

A través de este proyecto, se espera proporcionar una herramienta útil y entretenida para los fans(yo incluido) de One Piece y los juegos de cartas, al mismo tiempo que se refuerzan mis habilidades en programación, diseño de sistemas, y manejo de bases de datos.

OBJETIVO

El objetivo de este proyecto es trasladar el juego de mesa a un entorno virtual y gratuito donde los usuarios puedan jugar y pasar un buen rato

JUSTIFICACIÓN

La elección de esta temática surge de mi afición a dicho juego de cartas y la falta de un portal web donde se puede jugar a dicho juego, además del reciente auge de popularidad de los juegos de cartas digitales. Con este proyecto busco llenar un nicho específico al usar los dos elementos de una plataforma interactiva y accesible

ANÁLISIS DE LO EXISTENTE.

Innovaciones para el mercado:

1. Uso de la IP de One Piece:
 - Popularidad y atractivo: One Piece es una de las series de manga y anime más populares a nivel mundial. Utilizar esta IP atrae automáticamente a una base de fans amplia y leal.
 - Elementos narrativos: Integrar elementos de la historia y personajes de One Piece en un juego de cartas puede crear una experiencia más inmersiva y atractiva para los jugadores.
 -
2. Accesibilidad en navegador:
 - Plataforma accesible: Al ser un juego de navegador, no requiere instalaciones complejas ni dispositivos específicos de alta gama. Esto facilita que un mayor número de usuarios puedan acceder y jugar desde casi cualquier dispositivo con acceso a internet.

- Juego casual: Los juegos de navegador suelen atraer a jugadores casuales que buscan una experiencia de juego rápida y accesible.
3. Diseño de mecánicas de juego:
- Mecánicas únicas: Si el juego introduce mecánicas de juego únicas o innovadoras que no se encuentran en otros juegos de cartas, esto podría marcar una diferencia significativa.

APLICACIONES EXISTENTES EN EL MERCADO COMPARABLES

1) Hearthstone:

- Popularidad y alcance: Uno de los juegos de cartas digitales más populares desarrollado por Blizzard Entertainment.
- Mecánicas de juego: Conocido por sus mecánicas de construcción de mazos, uso estratégico de cartas y modos de juego diversos.
- Innovación: Hearthstone ha sido pionero en la creación de un ecosistema competitivo y una experiencia de usuario pulida.

2) Yu-Gi-Oh! Duel Links:

- Uso de IP: Basado en la popular franquicia de Yu-Gi-Oh!, similar a cómo se basa en One Piece.
- Mecánicas y narrativa: Utiliza personajes y escenarios del anime para crear una experiencia de juego envolvente.
- Accesibilidad: Disponible en múltiples plataformas, incluyendo navegadores y dispositivos móviles.

3) Magic: The Gathering Arena:

- Complejidad y profundidad: Conocido por su complejidad estratégica y la profundidad de sus mecánicas de juego.
- Plataformas: Disponible en múltiples plataformas, incluyendo navegadores.
- Experiencia de usuario: Ofrece una interfaz pulida y una experiencia competitiva sólida.

El punto de mi proyecto es para los fans de la serie One piece, con un juego de cartas sencillo con una curva de aprendizaje sencilla que invita a los nuevos jugadores a entrar cuando quieran

REQUISITOS FU

- Usuarios, una vista para que el usuario pueda identificarse o registrarse en caso de no estar registrado
- Una vista inicial donde vea distintas opciones, crear una partida, elegir mazo y ajustes de cuenta
- Parte de administrador para poder añadir cartas al juego
- Una vista donde cada usuario pueda elegir que cartas tener en su mazo
- Las cartas tendrán vida y ataque además de un coste de mana, el mana ira incrementando en cada ronda hasta un máximo de 10
- El juego será por turnos en el primer turno el primer jugador elegido aleatoriamente entre los dos obtendrá 1 mana y el segundo obtendrá dos de mana, por cada turno cada jugador obtendrá dos de mana excepto en los primeros turnos o cuando ya se hayan obtenido 10
- La mano inicial del jugador será aleatoria y el jugador podrá robar 1 carta por turno excepto el primer turno

Opcionales:

- Me gustaría añadir que las cartas tuvieran habilidades
- Poder agregar otros usuarios para poder retarse entre ellos
- La vista de la partida donde se jugará de manera multijugador 1vs1
- Una vista con tu historial de partidas

Consta recalcar que algunos requisitos se han ido modificando a medida del desarrollo del proyecto debido a complicaciones

ANÁLISIS Y DISEÑO:

DIAGRAMA DE LA ARQUITECTURA.

Está diseñada utilizando una arquitectura de cliente-servidor, donde el front funciona por React y el back por ASP.NET Core. La comunicación entre el cliente y el servidor se hace mediante una API restful, que facilita un intercambio de datos en formato JSON.

- Frontend: Desarrollado con React, el frontend presenta una interfaz de usuario dinámica y responsiva, adaptada para una amplia gama de dispositivos.
- Backend: El backend está construido con ASP.NET Core, proporcionando un framework robusto y escalable, la autenticación de usuarios, y las operaciones de la base de datos. Se encarga de procesar las solicitudes del cliente, interactuar con la base de datos MySQL.

- Base de Datos: Uso MySQL como sistema de gestión de base de datos, que almacena todos los datos necesarios. El diseño sigue un modelo relacional.

DISPOSICIÓN DE BACKEND POR CAPAS:

- Capa de Acceso a Datos (CDatos): En esta capa se pone todo lo relacionado con la base de datos, además de modelos con los que rellenaremos la información de la base de datos.
 - Funcionalidad: Acceso directo a la base de datos, realización de consultas y operaciones CRUD (Create, Read, Update, Delete).
 - Componentes: Modelos de datos y repositorios.
- Capa de Negocio/Dominio (CNegocio): Aquí se convierte la información de la capa de datos, se transforman los modelos de la capa de datos en DTOs (Data Transfer Objects). Tiene interfaces que proporcionan las funciones por defecto a los repositorios.
 - Funcionalidad: Implementación de la lógica del negocio, transformación de datos, validación y aplicación de reglas de negocio.
 - Componentes: Servicios de negocio, DTOs, interfaces y lógica de dominio.
- Capa de API: Esta capa llama a los repositorios de la capa de dominio y usa los DTOs para rellenar la información de los controladores que a su vez usan los repositorios para aplicar las funciones específicas.
 - Funcionalidad: Exposición de endpoints de la API, manejo de solicitudes HTTP, envío y recepción de datos a través de la API.
 - Componentes: Controladores y modelos de solicitud/respuesta.

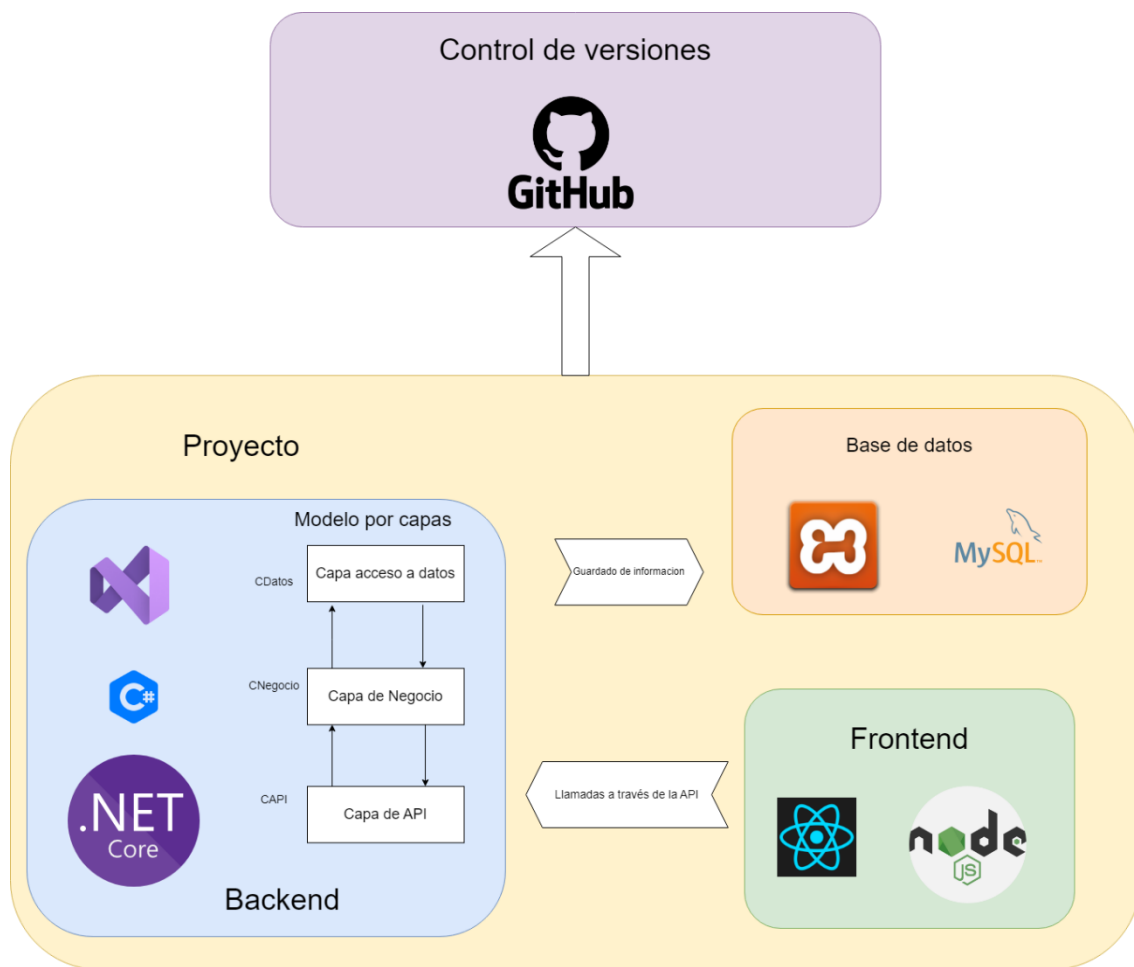
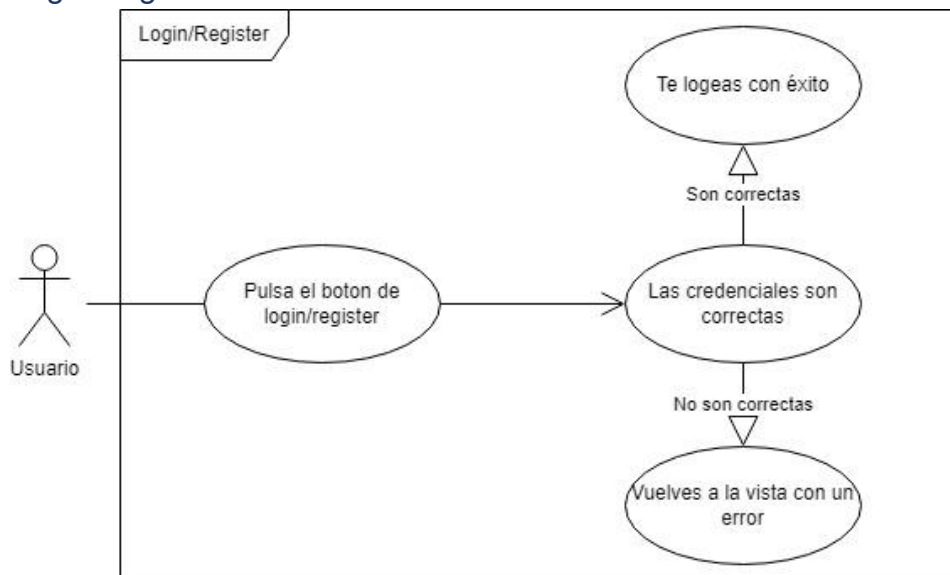


DIAGRAMA DE CASOS DE USOS

Login/Register



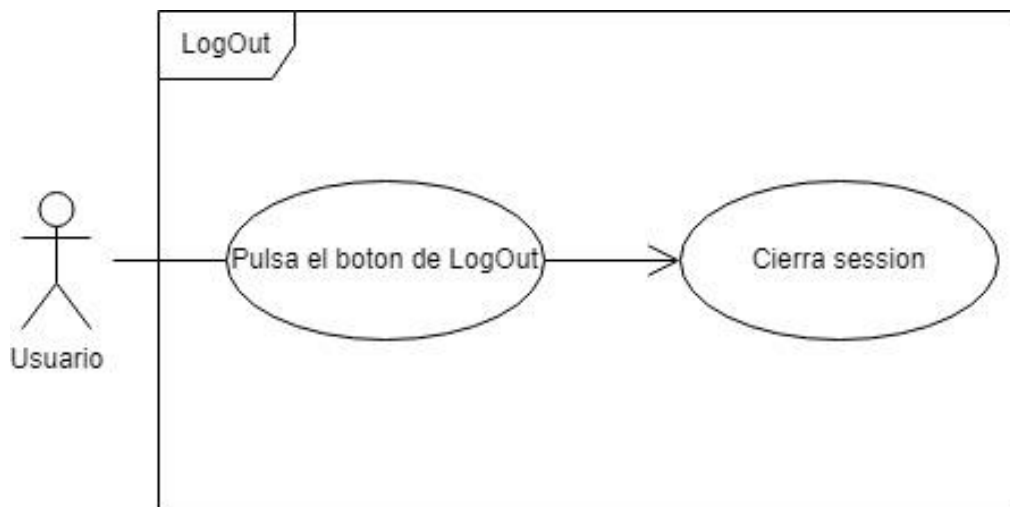
Login

- **Actor: Usuario**
- **Descripción:** El usuario introduce sus credenciales (nombre de usuario y contraseña) en la página de login y presiona el botón de iniciar sesión.
- **Flujo Principal:**
 1. El usuario introduce el nombre de usuario y la contraseña.
 2. El usuario presiona el botón "Iniciar sesión".
 3. El sistema verifica las credenciales.
 4. Si las credenciales son correctas, el usuario es redirigido a la página principal.
- **Flujos Alternativos:**
 - Si las credenciales son incorrectas, el sistema muestra un mensaje de error especificando el problema (por ejemplo, "Contraseña incorrecta" o "Usuario no encontrado").

Register

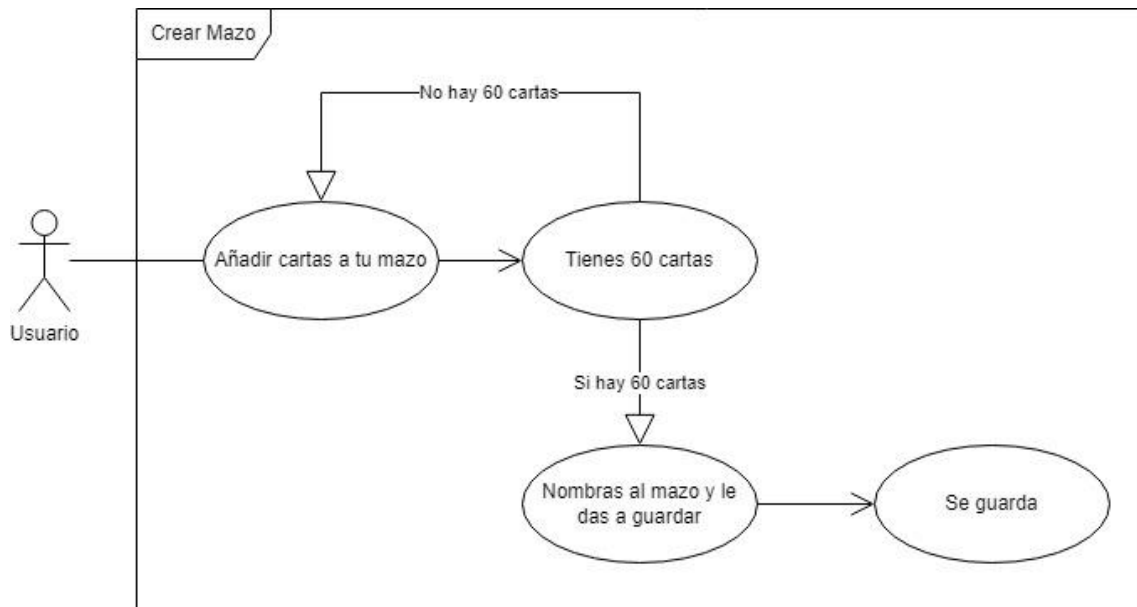
- **Actor: Usuario**
- **Descripción:** El usuario introduce los datos necesarios (nombre de usuario, contraseña, correo electrónico, etc.) en la página de registro y presiona el botón de registrar.
- **Flujo Principal:**
 1. El usuario introduce los datos requeridos.
 2. El usuario presiona el botón "Registrar".
 3. El sistema valida la información ingresada.
 4. Si los datos son correctos y el nombre de usuario no está en uso, el sistema crea una nueva cuenta y redirige al usuario a la página principal.
- **Flujos Alternativos:**
 - Si los datos son incorrectos o el nombre de usuario ya está en uso, el sistema muestra un mensaje de error especificando el problema (por ejemplo, "El nombre de usuario ya está en uso" o "Correo electrónico no válido").

Log Out



- **Actor: Usuario**
- **Descripción:** El usuario cierra su sesión actual.
- **Flujo Principal:**
 1. El usuario presiona el botón "Cerrar sesión".
 2. El sistema cierra la sesión del usuario y lo redirige a la página de inicio de sesión.

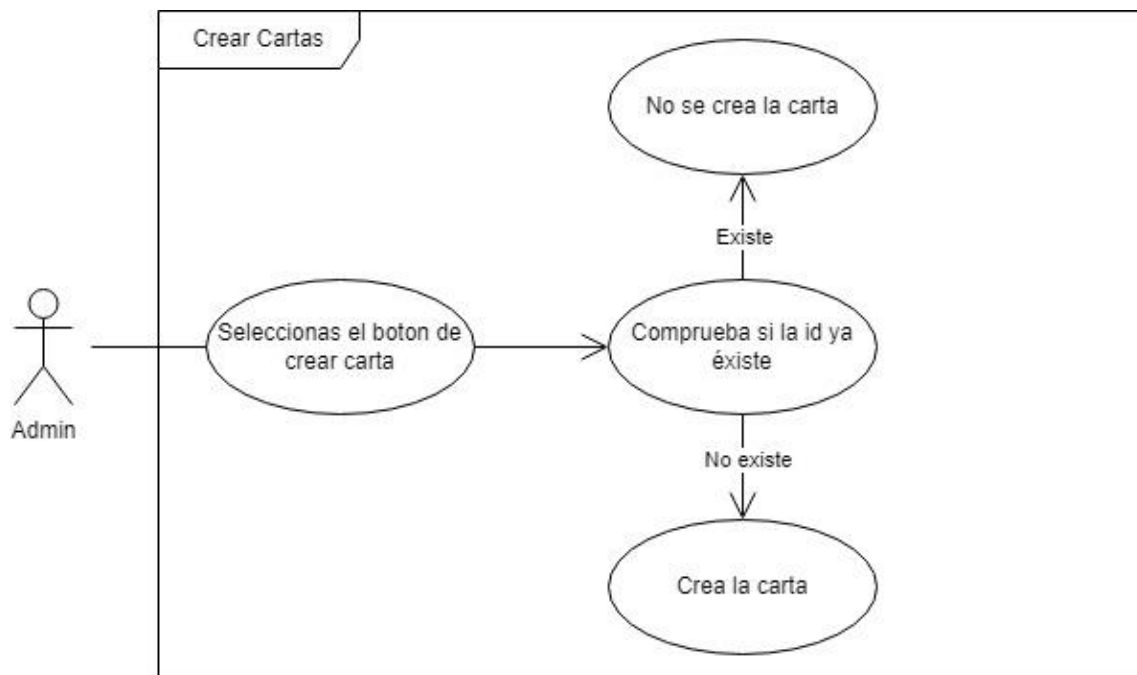
Crear Mazo



- **Actor:** Usuario
- **Descripción:** El usuario añade cartas a su mazo hasta completar 60 cartas, luego nombra el mazo y lo guarda.

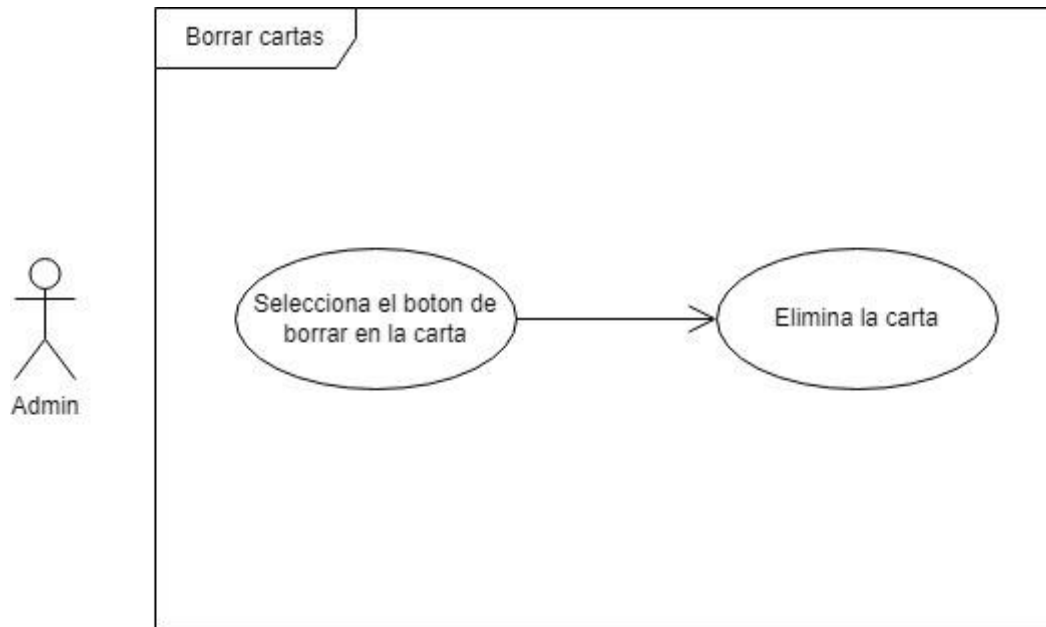
- **Flujo Principal:**
 1. El usuario selecciona la opción de "Crear Mazo".
 2. El usuario añade cartas a su mazo.
 3. El sistema verifica si el mazo contiene 60 cartas.
 4. Si el mazo contiene 60 cartas, el usuario le da un nombre al mazo y lo guarda.
 5. El sistema guarda el mazo.
- **Flujos Alternativos:**
 - Si el mazo no contiene 60 cartas, el usuario sigue añadiendo cartas hasta completar las 60.

Crear Cartas



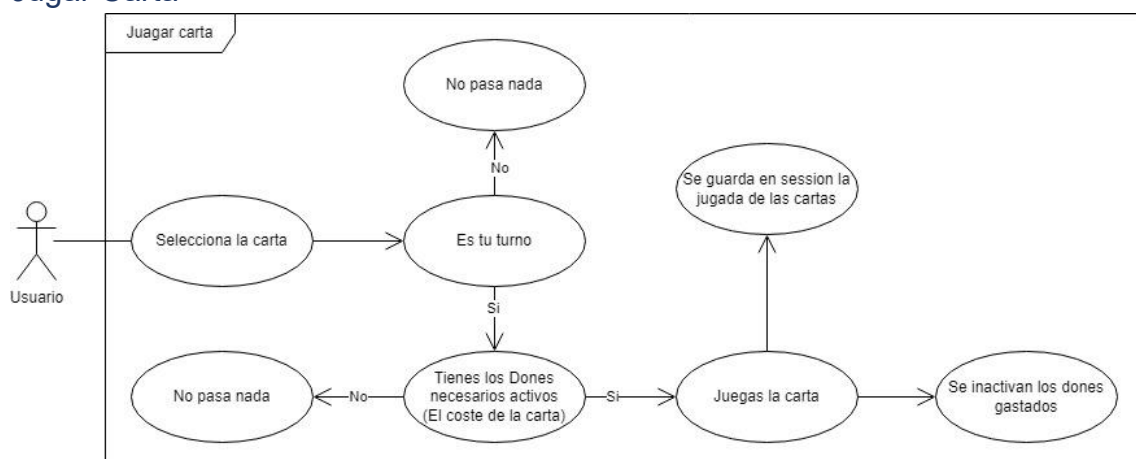
- **Actor:** Admin
- **Descripción:** El administrador selecciona la opción de crear carta, el sistema verifica si la ID de la carta ya existe. Si la ID existe, la carta no se crea y se muestra un mensaje de error. Si la ID no existe, la carta se crea exitosamente.
- **Flujo Principal:**
 1. El administrador selecciona la opción de "Crear Carta".
 2. El administrador introduce los detalles de la nueva carta.
 3. El sistema verifica si la ID de la carta ya existe.
 4. Si la ID no existe, el sistema crea la carta.
 5. La carta se guarda en la base de datos.
- **Flujos Alternativos:**
 - Si la ID de la carta ya existe, el sistema muestra un mensaje de error y no crea la carta.

Borrar Cartas



- **Actor:** Admin
- **Descripción:** El administrador selecciona la opción de borrar una carta y el sistema elimina la carta seleccionada.
- **Flujo Principal:**
 1. El administrador selecciona el botón de "Borrar" en la carta.
 2. El sistema elimina la carta de la base de datos.

Jugar Carta



- **Actor:** Usuario
- **Descripción:** El usuario selecciona una carta para jugar durante su turno. El sistema verifica si es el turno del usuario y si tiene los recursos necesarios para jugar la carta. Si cumple con los requisitos, el sistema guarda la jugada y actualiza los recursos.

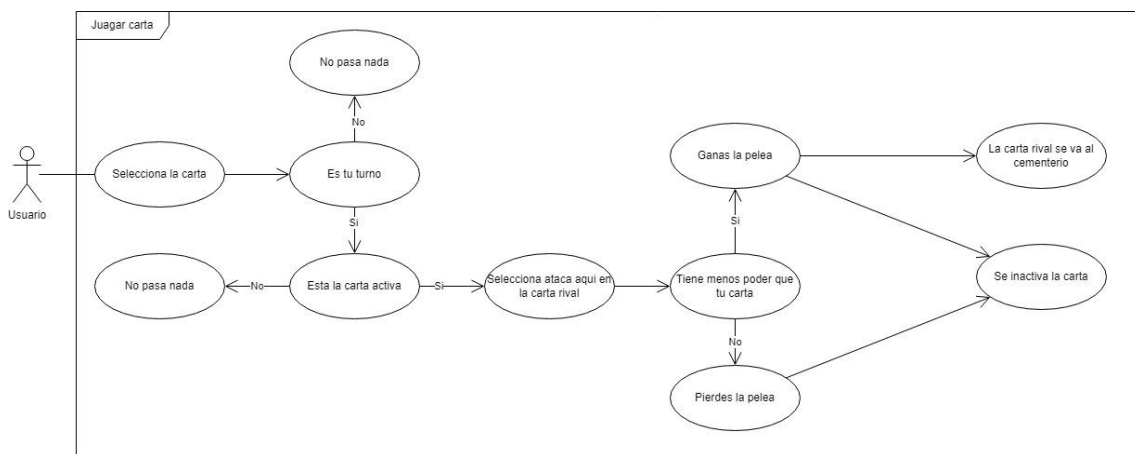
- **Flujo Principal:**

1. El usuario selecciona la carta que desea jugar.
2. El sistema verifica si es el turno del usuario.
3. Si es el turno del usuario, el sistema verifica si tiene los dones (recursos) necesarios para jugar la carta.
4. Si el usuario tiene los recursos necesarios, juega la carta.
5. El sistema guarda la jugada de la carta en la sesión.
6. El sistema inactiva los dones gastados.

- **Flujos Alternativos:**

- Si no es el turno del usuario, no pasa nada.
- Si el usuario no tiene los recursos necesarios, no pasa nada.

Atacar Carta a Carta



- **Actor:** Usuario

- **Descripción:** El usuario selecciona una carta activa para atacar a una carta rival. El sistema verifica si es el turno del usuario y si la carta está activa. Si cumple con los requisitos, el sistema compara el poder de las cartas y determina el resultado del ataque.

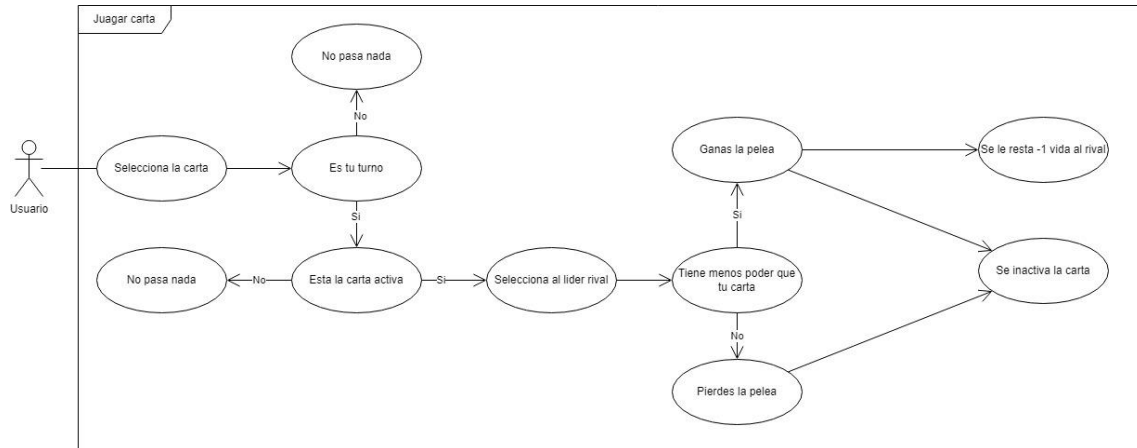
- **Flujo Principal:**

1. El usuario selecciona la carta que desea usar para atacar.
2. El sistema verifica si es el turno del usuario.
3. Si es el turno del usuario, el sistema verifica si la carta está activa.
4. Si la carta está activa, el usuario selecciona una carta rival para atacar.
5. El sistema compara el poder de las cartas.
6. Si la carta del usuario tiene más poder, gana la pelea:
 - La carta rival se va al cementerio.
 - La carta del usuario se inactiva.

- **Flujos Alternativos:**

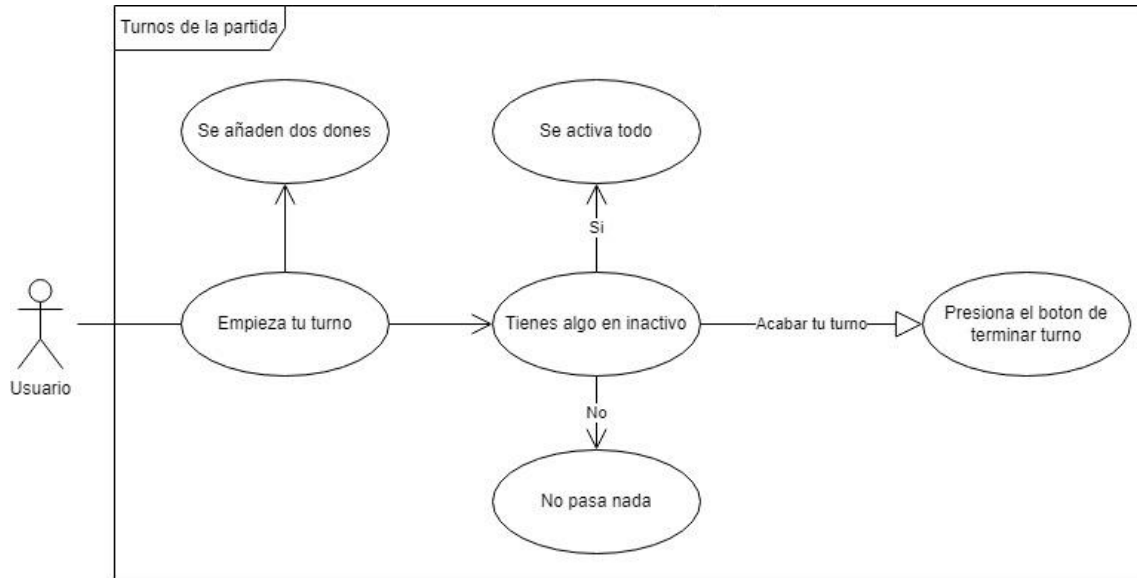
- Si no es el turno del usuario, no pasa nada.
- Si la carta no está activa, no pasa nada.
- Si la carta del usuario tiene menos poder, pierde la pelea y la carta del usuario se inactiva.

Atacar Carta a Líder



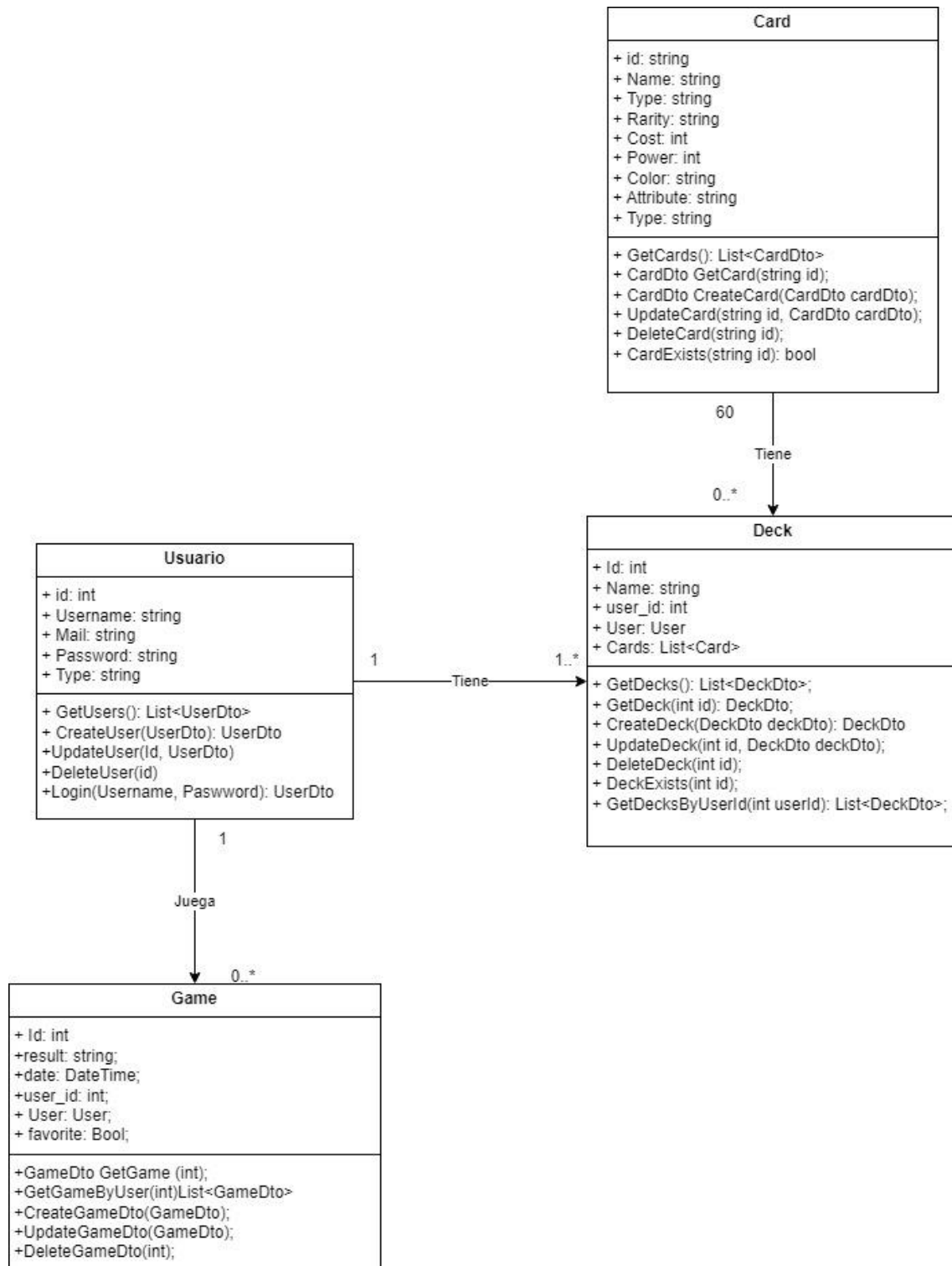
- **Actor:** Usuario
- **Descripción:** El usuario selecciona una carta activa para atacar al líder rival. El sistema verifica si es el turno del usuario y si la carta está activa. Si cumple con los requisitos, el sistema compara el poder de la carta del usuario con el poder del líder rival y determina el resultado del ataque.
- **Flujo Principal:**
 1. El usuario selecciona la carta que desea usar para atacar.
 2. El sistema verifica si es el turno del usuario.
 3. Si es el turno del usuario, el sistema verifica si la carta está activa.
 4. Si la carta está activa, el usuario selecciona al líder rival para atacar.
 5. El sistema compara el poder de la carta del usuario con el poder del líder rival.
 6. Si la carta del usuario tiene más poder, gana la pelea:
 - Se le resta 1 vida al rival.
 - La carta del usuario se inactiva.
- **Flujos Alternativos:**
 - Si no es el turno del usuario, no pasa nada.
 - Si la carta no está activa, no pasa nada.
 - Si la carta del usuario tiene menos poder, pierde la pelea y la carta del usuario se inactiva.

Turnos de las partidas



- **Actor:** Usuario
- **Descripción:** El usuario inicia su turno, se añaden dones, verifica si tiene algo en estado inactivo, y luego puede terminar su turno.
- **Flujo Principal:**
 1. El usuario empieza su turno.
 2. El sistema añade dos dones.
 3. El sistema verifica si el usuario tiene algo en estado inactivo.
 4. Si el usuario tiene algo en estado inactivo, el sistema lo activa.
 5. El usuario presiona el botón de "Terminar turno" para acabar su turno.
- **Flujos Alternativos:**
 - Si el usuario no tiene nada en estado inactivo, no pasa nada.

DIAGRAMA DE CLASES



Este diagrama de clases muestra la estructura de clases y sus relaciones en el sistema de este proyecto

1. Clase Card

- **Atributos:**
 - id: Identificador único de la carta (string).

- Name: Nombre de la carta (string).
- Type: Tipo de la carta (string).
- Rarity: Rareza de la carta (string).
- Cost: Coste de la carta en dones (int).
- Power: Poder de ataque de la carta (int).
- Color: Color asociado a la carta (string).
- Attribute: Atributo o habilidad de la carta (string).
- Type: Tipo de la carta (string, aparece dos veces, posiblemente un error).
- **Métodos:**
 - GetCards(): Devuelve una lista de objetos CardDto.
 - GetCard(id): Devuelve un objeto CardDto basado en el ID proporcionado.
 - CreateCard(cardDto): Crea una nueva carta a partir de un objeto CardDto.
 - UpdateCard(id, cardDto): Actualiza una carta existente basada en su ID y el objeto CardDto proporcionado.
 - DeleteCard(id): Elimina una carta basada en su ID.
 - CardExists(id): Verifica si una carta con el ID proporcionado existe.

2. Clase Deck

- **Atributos:**
 - Id: Identificador único del mazo (int).
 - Name: Nombre del mazo (string).
 - user_id: Identificador del usuario propietario del mazo (int).
 - User: Referencia al usuario propietario del mazo (User).
 - Cards: Lista de cartas incluidas en el mazo.
- **Métodos:**
 - GetDecks(): Devuelve una lista de objetos DeckDto.
 - GetDeck(id): Devuelve un objeto DeckDto basado en el ID proporcionado.
 - CreateDeck(deckDto): Crea un nuevo mazo a partir de un objeto DeckDto.
 - UpdateDeck(id, deckDto): Actualiza un mazo existente basado en su ID y el objeto DeckDto proporcionado.
 - DeleteDeck(id): Elimina un mazo basado en su ID.
 - DeckExists(id): Verifica si un mazo con el ID proporcionado existe.
 - GetDecksByUserId(userId): Devuelve una lista de mazos pertenecientes a un usuario específico.

3. Clase User

- **Atributos:**
 - id: Identificador único del usuario (int).
 - Username: Nombre de usuario (string).
 - Mail: Correo electrónico del usuario (string).
 - Password: Contraseña del usuario (string).

- Type: Tipo de usuario (por ejemplo, admin, jugador) (string).
- **Métodos:**
 - GetUsers(): Devuelve una lista de objetos UserDto.
 - CreateUser(userDto): Crea un nuevo usuario a partir de un objeto UserDto.
 - UpdateUser(id, userDto): Actualiza un usuario existente basado en su ID y el objeto UserDto proporcionado.
 - DeleteUser(id): Elimina un usuario basado en su ID.
 - Login(username, password): Autentica a un usuario basado en el nombre de usuario y la contraseña proporcionados.

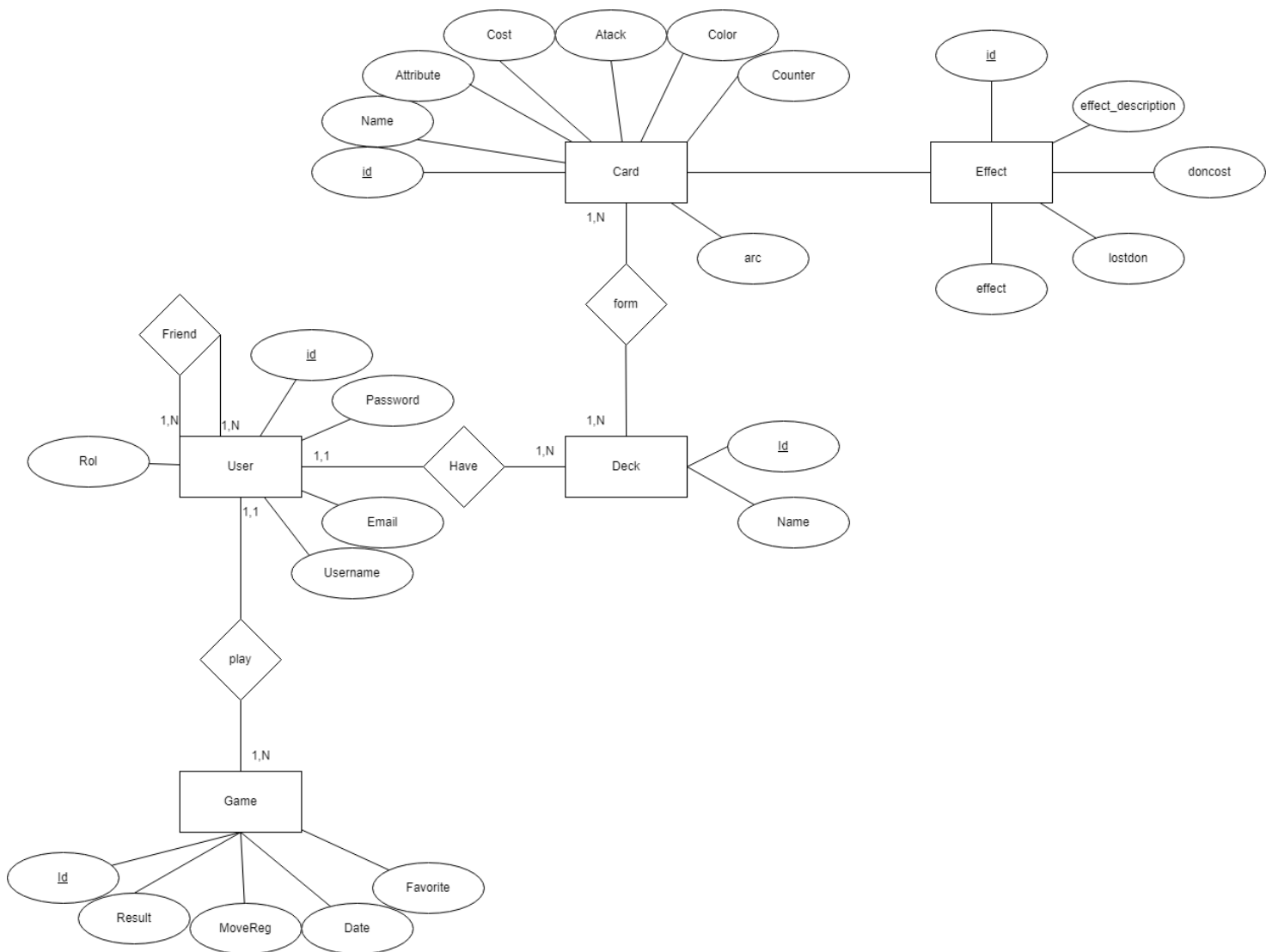
4. Clase Game

- **Atributos:**
 - Id: Identificador único del juego (int).
 - Result: Resultado del juego (string).
 - date: Fecha del juego (DateTime).
 - user_id: Identificador del usuario que jugó el juego (int).
 - User: Referencia al usuario que jugó (User).
 - favorite: Indica si el juego es favorito (bool).
- **Métodos:**
 - GetGame(id): Devuelve un objeto GameDto basado en el ID proporcionado.
 - GetGameByUser(userId): Devuelve una lista de objetos GameDto pertenecientes a un usuario específico.
 - CreateGame(gameDto): Crea un nuevo juego a partir de un objeto GameDto.
 - UpdateGame(id, gameDto): Actualiza un juego existente basado en su ID y el objeto GameDto proporcionado.
 - DeleteGame(id): Elimina un juego basado en su ID.

Relaciones entre Clases

- **User y Deck:** Un usuario puede tener múltiples mazos (1..*), mientras que un mazo pertenece a un único usuario (1).
- **Deck y Card:** Un mazo contiene múltiples cartas (0..*), y una carta puede estar en múltiples mazos.
- **User y Game:** Un usuario puede jugar múltiples juegos (0..*), y un juego pertenece a un único usuario (1).

DISEÑO DE DATOS.



Resumen

El modelo relacional del juego de cartas está compuesto por las siguientes tablas principales:

- **User**: Representa a los usuarios del sistema, con sus credenciales, roles y relaciones de amistad con otros usuarios.
- **Deck**: Representa los mazos creados por los usuarios, que contienen varias cartas.
- **Card**: Representa las cartas individuales que tienen varias características como nombre, atributo, coste, ataque, color, y contador.
- **Effect**: Describe los efectos de las cartas, incluyendo el costo y los dones perdidos.
- **Game**: Almacena información sobre las partidas jugadas, incluyendo el resultado, los movimientos registrados, la fecha y si es un juego favorito.
- **Friend**: Gestiona las relaciones de amistad entre los usuarios.
-

1. Tabla User

- **Atributos:**
 - id: INT, PK, AUTO_INCREMENT
 - username: VARCHAR(255)
 - email: VARCHAR(255)
 - password: VARCHAR(255)
 - role: VARCHAR(50)
- **Relaciones:**
 - Un usuario puede tener múltiples mazos.
 - Un usuario puede tener múltiples amigos.
 - Un usuario puede jugar múltiples juegos.

2. Tabla Deck

- **Atributos:**
 - id: INT, PK, AUTO_INCREMENT
 - name: VARCHAR(255)
 - user_id: INT, FK
- **Relaciones:**
 - Un mazo puede contener múltiples cartas (tabla intermedia DeckCard).
 - Un mazo puede tener múltiples efectos (tabla intermedia DeckEffect).

3. Tabla Card

- **Atributos:**
 - id: INT, PK, AUTO_INCREMENT
 - name: VARCHAR(255)
 - attribute: VARCHAR(50)
 - cost: INT
 - attack: INT
 - color: VARCHAR(50)
 - counter: INT

4. Tabla Effect

- **Atributos:**
 - id: INT, PK, AUTO_INCREMENT
 - effect_description: TEXT
 - don_cost: INT
 - lost_don: INT

5. Tabla Game

- **Atributos:**
 - id: INT, PK, AUTO_INCREMENT
 - result: VARCHAR(50)

- move_reg: TEXT
- date: DATETIME
- favorite: BOOLEAN
- user_id: INT, FK

6. Tabla Intermedia DeckCard

- **Atributos:**
 - deck_id: INT, PK, FK
 - card_id: INT, PK, FK
- **Descripción:** Relaciona mazos (Deck) con cartas (Card).

7. Tabla Intermedia CardEffect

- **Atributos:**
 - card_id: INT, PK, FK
 - effect_id: INT, PK, FK
- **Descripción:** Relaciona cartas (Card) con efectos (Effect).

8. Tabla Intermedia Friend

- **Atributos:**
 - user_id_1: INT, PK, FK
 - user_id_2: INT, PK, FK
- **Descripción:** Relaciona usuarios (User) que son amigos entre sí

CODIFICACIÓN

ENTORNO DE PROGRAMACIÓN

Herramientas Utilizadas

1. Visual Studio Community
 - Propósito: Utilizado principalmente para el desarrollo del backend con ASP.NET Core.
 - Características: Ofrece un entorno de desarrollo integrado (IDE) robusto con herramientas avanzadas para la depuración, pruebas y despliegue de aplicaciones.
2. Visual Studio Code
 - Propósito: Utilizado principalmente para el desarrollo del frontend con React y la edición de scripts de la base de datos.
 - Características: Es un editor de código fuente ligero y altamente extensible que soporta una amplia variedad de lenguajes y herramientas. Se utilizaron extensiones como ESLint para JavaScript y

Prettier para formatear el código, además de la integración con Git para el control de versiones.

3. XAMPP para MySQL

- **Propósito:** Utilizado para alojar y gestionar la base de datos MySQL localmente.
- **Características:** XAMPP es una distribución de Apache que incluye MySQL, PHP y Perl. Es fácil de instalar y proporciona una forma rápida de configurar un servidor local para desarrollo.
- **Configuración:** La base de datos del proyecto se gestiona y se ejecuta en el servidor MySQL incluido en XAMPP.

Lenguajes y Herramientas

1. C#

- **Uso:** Desarrollo del backend utilizando ASP.NET Core.
- **Propósito:** Manejar la lógica del servidor, la autenticación de usuarios y la interacción con la base de datos.

2. React

- **Uso:** Desarrollo del frontend.
- **Propósito:** Crear una interfaz de usuario dinámica y responsiva que interactúe con el backend mediante una API RESTful.

4. MySQL

- **Uso:** Sistema de gestión de bases de datos.
- **Propósito:** Almacenar y gestionar los datos de la aplicación.

5. Python

- **Uso:** Rellenar la base de datos con datos iniciales.
- **Propósito:** Scripting para la automatización de tareas relacionadas con la base de datos, como la inserción de datos.

Aspectos relevantes de la implementación

En esta sección se destacan las características y funcionalidades que hacen que este juego de cartas se diferencie de otras aplicaciones similares en el mercado.

1. Interfaz de Usuario Intuitiva y Responsiva

- **Diseño Moderno:** La interfaz de usuario ha sido diseñada utilizando React, lo que proporciona una experiencia de usuario fluida y moderna.
- **Responsividad:** La aplicación está optimizada para funcionar en una amplia gama de dispositivos, incluyendo computadoras de escritorio, tabletas y teléfonos móviles, lo que permite a los usuarios jugar en cualquier lugar y en cualquier momento.

2. Manejo de Mazos Avanzado

- **Facilidad de Uso:** La creación y gestión de mazos es sencilla e intuitiva, permitiendo a los usuarios añadir, editar y eliminar cartas con facilidad.
- **Validación de Mazos:** Implementación de reglas de validación para asegurar que los mazos cumplan con los requisitos del juego, como el límite de 60 cartas.

3. Sistema de Dones (Mana) Dinámico

- **Mecánica Innovadora:** Cada turno, los jugadores reciben dones adicionales hasta un máximo de 10, lo que añade una capa estratégica única al juego.
- **Balance de Juego:** El sistema de dones está diseñado para mantener un equilibrio competitivo, evitando desventajas significativas entre jugadores.

Personalización y Expansión

- **Personalización de Mazos:** Los usuarios pueden personalizar sus mazos con una variedad de cartas, permitiendo estrategias únicas y personalizadas.
- **Expansión Futura:** La arquitectura del sistema está diseñada para facilitar la adición de nuevas cartas, modos de juego y funcionalidades sin afectar la estabilidad del juego actual.

Manual de usuario.

Introducción

Descripción del Proyecto: Este proyecto es un juego de cartas desarrollado como trabajo de fin de curso de un ciclo superior. El juego consiste en partidas uno contra uno, donde cada jugador debe derrotar al líder del oponente utilizando cartas y estrategias. El objetivo es terminar con el líder del oponente antes de que el oponente lo haga contigo.

Audiencia Objetivo: Este manual está dirigido a los jugadores del juego de cartas, proporcionando instrucciones claras sobre cómo registrarse, iniciar sesión, crear mazos y jugar partidas.

Requisitos del Sistema

Hardware:

- Cualquier ordenador que permita abrir un navegador

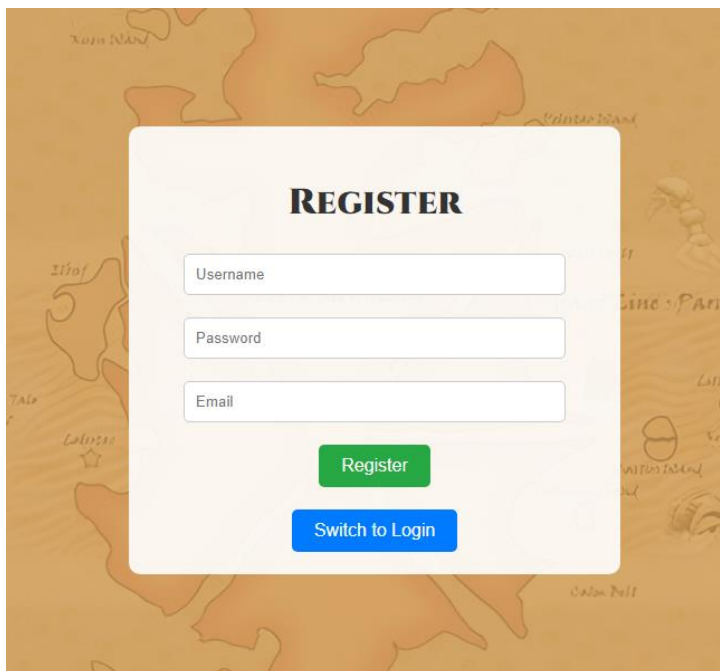
Software:

- Navegador Web: Google Chrome, Mozilla Firefox, Microsoft Edge...

Registro e Inicio de Sesión

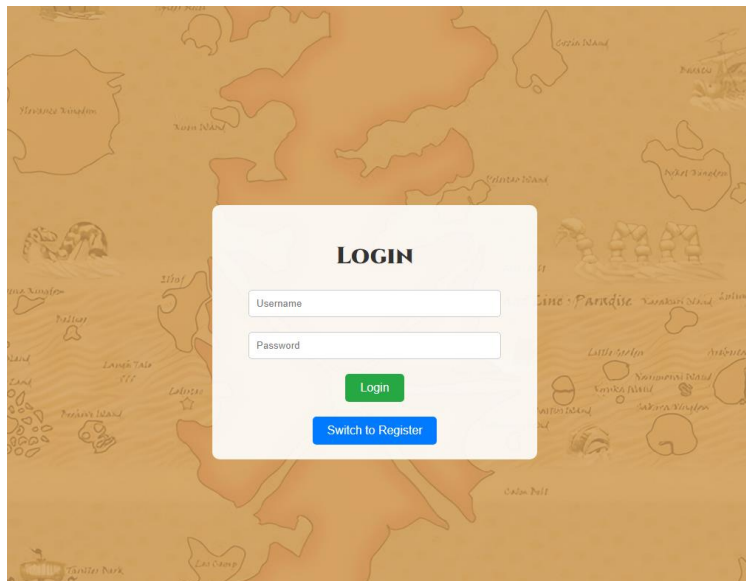
Registro de Usuario:

1. Navega a la página de registro.
2. Introduce tu nombre de usuario, correo electrónico y contraseña.
3. Presiona el botón "Registrar".
4. Recibirás un correo de confirmación. Sigue las instrucciones para activar tu cuenta.

A registration form titled "REGISTER" is centered on a light orange background with faint, stylized world map outlines. The form is a white rounded rectangle containing three input fields: "Username", "Password", and "Email". Below these fields are two buttons: a green "Register" button and a blue "Switch to Login" button.

Inicio de Sesión:

1. Navega a la página de inicio de sesión.
2. Introduce tu nombre de usuario y contraseña.
3. Presiona el botón "Iniciar sesión".

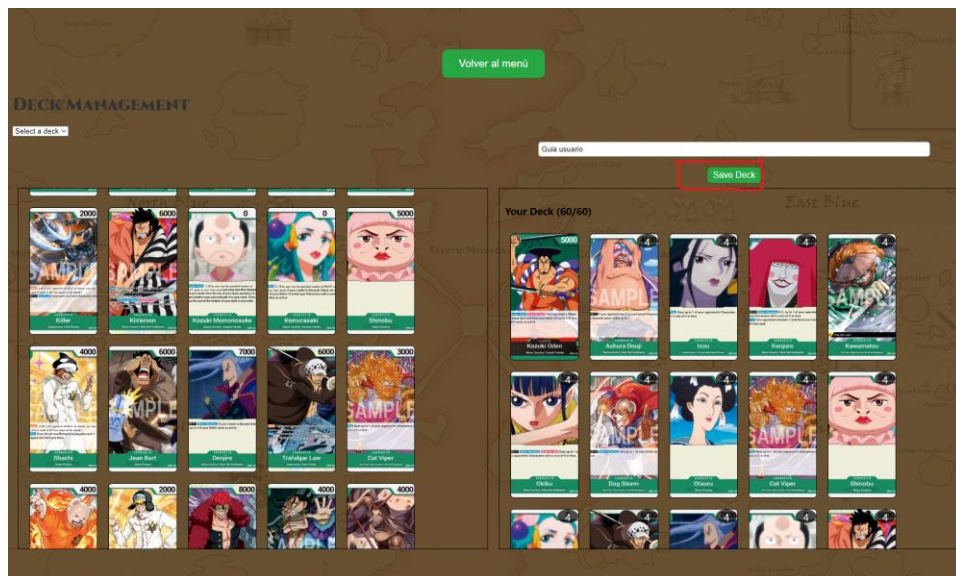


Creación y Gestión de Mazos



Crear un Mazo:

1. Después de iniciar sesión, ve a la sección "Crear Mazo".
2. Selecciona las cartas que quieres añadir a tu mazo desde la lista de cartas disponibles.
3. Asegúrate de tener un total de 60 cartas en tu mazo.
4. Nombra tu mazo en el campo correspondiente.
5. Presiona el botón "Guardar" para guardar tu mazo.



Jugando una Partida

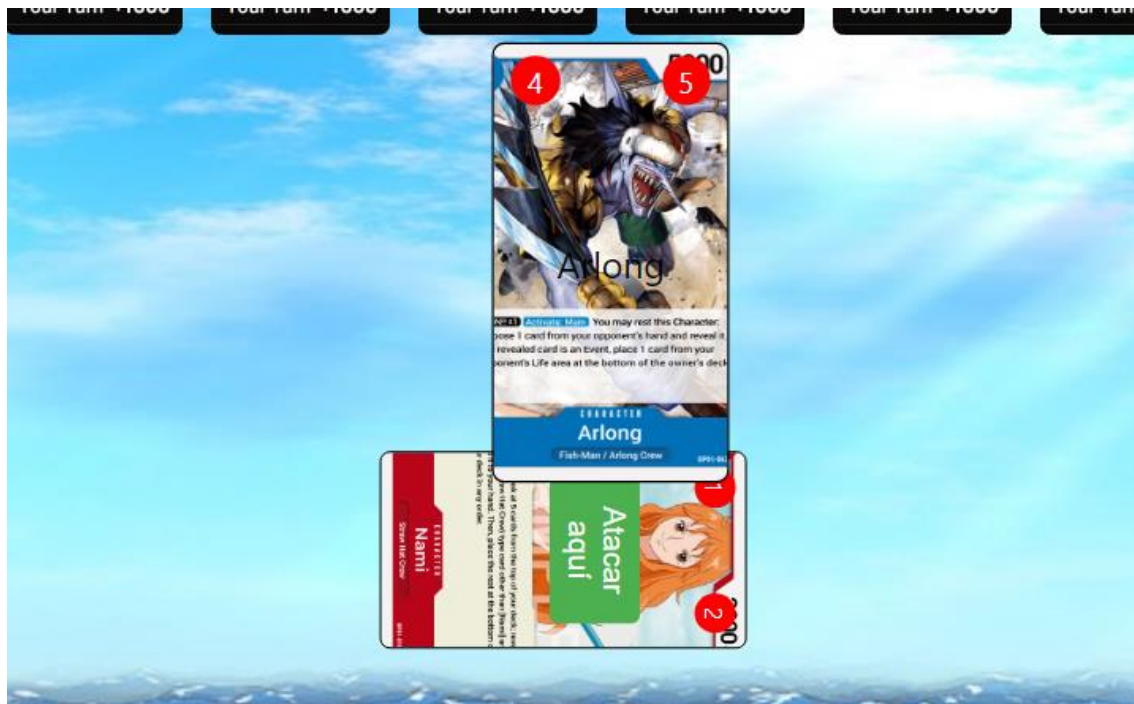
Iniciar una Partida:

1. Selecciona la opción "Jugar" en el menú principal.
2. Elige el mazo que deseas usar para la partida.
3. El segundo jugador tiene que elegir mazo y presionáis el botón



Desarrollo del Juego:

1. **Inicio de la Partida:**
 - Ambos jugadores reciben 5 cartas en mano y 1 carta de líder.
 - El primer jugador recibe 1 don (mana/coste).
2. **Turnos:**
 - Cada turno, los jugadores roban dos dones (hasta un máximo de 10).
 - Los jugadores pueden jugar cartas de su mano utilizando los dones disponibles.
 - Las cartas pueden ser utilizadas para atacar al líder del oponente o defenderse de ataques.
 - Las cartas jugadas pueden tener habilidades especiales que afectan el juego.
 - Para terminar el turno, presiona el botón "Terminar Turno".
3. **Atacar:**
 - Selecciona una carta activa para atacar.
 - Elige el objetivo del ataque (carta rival o líder rival).
 - Si la carta atacante tiene más poder, el objetivo recibe daño o es eliminado.



4. Fin de la Partida:

- La partida termina cuando el líder de uno de los jugadores es derrotado.
- El jugador que derrota al líder rival gana la partida.

Audiencia Objetivo: Esta guía está dirigida a los administradores del juego, proporcionando instrucciones claras sobre cómo gestionar las cartas, incluyendo la creación, actualización y eliminación de cartas.

Acceso a Funciones de Administración

Inicio de Sesión como Administrador:

1. Navega a la página de inicio de sesión.
2. Introduce tu nombre de usuario y contraseña.
3. Presiona el botón "Iniciar sesión".
4. Asegúrate de que tu cuenta tiene privilegios de administrador.

Gestión de Cartas

Añadir Nueva Carta:

1. Ve a la sección "Administrar Cartas" en el menú principal.
2. Presiona el botón "Añadir Carta".
3. Rellena los detalles de la nueva carta:
 - **Nombre:** El nombre de la carta.

- **Atributo:** El atributo o tipo de la carta.
 - **Coste:** El coste en dones (mana/coste) necesario para jugar la carta.
 - **Ataque:** El valor de ataque de la carta.
 - **Color:** El color asociado con la carta.
 - **Contador:** Valor adicional si la carta tiene alguna habilidad especial.
 - **Descripción del Efecto:** Una descripción del efecto de la carta si tiene alguno.
 - **Costo del Efecto:** El coste en dones para activar el efecto de la carta.
 - **Dones Perdidos:** La cantidad de dones que se pierden al activar el efecto.
4. Presiona el botón "Guardar" para añadir la carta al juego.

Actualizar Carta Existente:

1. En la sección "Administrar Cartas", busca la carta que deseas actualizar.
2. Presiona el botón "Editar" junto a la carta seleccionada.
3. Realiza los cambios necesarios en los detalles de la carta.
4. Presiona el botón "Guardar" para actualizar la carta.

Eliminar Carta:

1. En la sección "Administrar Cartas", busca la carta que deseas eliminar.
2. Presiona el botón "Eliminar" junto a la carta seleccionada.
3. Confirma la eliminación de la carta cuando se te solicite.

Procedimiento para la Gestión de Cartas

1. **Acceso a la Gestión de Cartas:**
 - Solo los usuarios con privilegios de administrador pueden acceder a la sección de gestión de cartas.
 - Navega a "Administrar Cartas" desde el menú principal después de iniciar sesión como administrador.
2. **Añadir Carta:**
 - Rellena todos los campos obligatorios y asegúrate de que los valores sean correctos.
 - Verifica que el ID de la carta no exista ya en el sistema para evitar duplicados.
3. **Actualizar Carta:**
 - Realiza los cambios necesarios, como actualizar atributos o costes.
 - Verifica que los cambios sean consistentes con las reglas del juego.
4. **Eliminar Carta:**
 - Asegúrate de que eliminar una carta no afectará negativamente a las partidas en curso.
 - Considera desactivar la carta en lugar de eliminarla si existe la posibilidad de restaurarla en el futuro.

REQUISITOS E INSTALACIÓN.

Hardware:

- **CPU:** Un procesador de al menos 2 núcleos es suficiente para empezar. A medida que el tráfico y la cantidad de cartas aumenten, podría ser necesario utilizar procesadores con más núcleos.
- **Memoria RAM:** Mínimo 4 GB de RAM para un rendimiento básico. Se recomienda aumentar la RAM si se espera un tráfico elevado o se añaden más cartas y funcionalidades.
- **Almacenamiento:** Cualquier disco mayor a 100 GB será suficiente inicialmente. La aplicación actual pesa menos, aunque podría ampliarse en el futuro a medida que se añadan más cartas y datos.
- **Ancho de Banda:** Depende del volumen de tráfico esperado. Para un tráfico medio-bajo, la conexión estándar de banda ancha debería ser suficiente.

Software:

- **Sistema Operativo:** Linux (distribuciones populares incluyen Ubuntu, CentOS o Debian). También puedes utilizar Windows si lo prefieres.
- **Servidor Web:** Apache o Nginx.
- **Base de Datos:** Instalación de MySQL o MariaDB (una bifurcación de MySQL).
- **Node.js:** Necesario para ejecutar el entorno de desarrollo de React.
- **Ejecutable de WebAPI:** Un ejecutable de mi WebAPI, que será proporcionado por mí, o puedes crearlo a partir de los archivos que se encuentran en GitHub.

Instalación

Preparativos:

1. Descarga e instala XAMPP desde apachefriends.org para gestionar MySQL.
2. Descarga e instala Visual Studio Community desde visualstudio.microsoft.com.
3. Descarga e instala Visual Studio Code desde code.visualstudio.com.
4. Descarga e instala Node.js desde nodejs.org para gestionar las dependencias de React.

Despliegue de la Base de Datos:

Una vez instalado Xampp tendremos que ejecutar el servicio de Apache y MSQl

Lo primero que haremos será crear la base de datos por lo que nos dirigimos a <http://localhost/phpmyadmin> teniendo iniciado el servicio de MySQL.

Una vez ahí crearemos una base de datos nueva y vacía con el nombre “onecardgame”

Cuando se haya creado importaremos el archivo sql que se adjunta en las carpetas del proyecto, este creará todas las entidades y les dará contenido

Despliegue de la WebAPI:

1. **Obtener el ejecutable de la WebAPI:**
 - Puedes obtener el ejecutable directamente de mí o compilarlo desde el repositorio GitHub.
2. **Configurar el servidor web:**
 - Configura Apache o Nginx para servir el ejecutable de la WebAPI.
 - Asegúrate de que el servidor web pueda acceder a la base de datos y que todas las configuraciones de red estén correctamente establecidas.

Instalación del Frontend:

1. **Clonar el repositorio del frontend:**

```
git clone https://github.com/HumanoHDR/TFGCardGame
```

2. **Abrir el proyecto en Visual Studio Code:**
 - Abre Visual Studio Code.
 - Navega hasta el directorio del repositorio clonado.
3. **Instalar las dependencias de Node.js:**
 - Abre una terminal en Visual Studio Code y ejecuta:

```
npm install
```

4. **Iniciar el servidor de desarrollo:**
 - En la misma terminal, ejecuta:

```
npm start
```

Usuario por defecto para comprobar la aplicación

Username: admin
Password: admin

CONCLUSIONES

CONCLUSIONES SOBRE EL TRABAJO REALIZADO

Este proyecto ha sido un desafío significativo que me ha permitido profundizar en tecnologías clave como React y ASP.NET. A pesar de que el tiempo

proporcionado de 40 horas no fue suficiente para abordar todas las complejidades del proyecto, el resultado obtenido, aunque mejorable, me ha dejado satisfecho con mi trabajo.

Aprendizajes y Retos

Comprensión de React y ASP.NET:

Este proyecto me ha ayudado a comprender mejor cómo utilizar React para desarrollar interfaces de usuario dinámicas y responsivas. La gestión del estado y la integración con APIs RESTful mediante la librería Axios fueron áreas clave de aprendizaje.

En el backend, tuve que aprender a crear una API en C# utilizando ASP.NET Core. Implementar la arquitectura por capas fue un reto, pero me permitió estructurar el código de una manera modular y mantenible.

Desarrollo de la API en C#:

Aprender a desarrollar una API en C# fue una de las mayores curvas de aprendizaje del proyecto. La creación de controladores, servicios y repositorios en una arquitectura por capas me proporcionó una comprensión más profunda de cómo estructurar aplicaciones web complejas.

Uso de Axios en React:

La integración de la API con el frontend utilizando Axios en React fue una experiencia valiosa. Me permitió comprender cómo manejar solicitudes HTTP, gestionar el estado de la aplicación y actualizar la interfaz de usuario de manera eficiente.

Evaluación del Proyecto

Tiempo Limitado:

El tiempo proporcionado de 40 horas fue insuficiente para completar todas las funcionalidades previstas. Sin embargo, logré implementar las características esenciales del juego de cartas, lo cual me dejó una sensación de logro.

Resultado Final

Aunque el proyecto es mejorable en varios aspectos, estoy satisfecho con el trabajo realizado. Pude crear un juego funcional con una interfaz de usuario atractiva y un backend robusto.

Mejora de Habilidades

Este proyecto no solo me permitió poner a prueba mis habilidades actuales, sino que también me ayudó a mejorarlas significativamente. La experiencia adquirida en el desarrollo de aplicaciones web con React y ASP.NET, junto con la gestión de APIs y el uso de librerías como Axios, será invaluable para futuros proyectos.

Conclusión

En conclusión, este proyecto ha sido una excelente oportunidad para probar y mejorar mis habilidades en desarrollo web. Me ha permitido trabajar con tecnologías modernas, resolver problemas complejos y entender mejor las buenas prácticas de desarrollo. A pesar de las limitaciones de tiempo, el esfuerzo invertido ha resultado en un producto funcional del cual me siento orgulloso. Estoy seguro de que los conocimientos y experiencias adquiridos durante este proceso serán de gran utilidad en mi carrera profesional.

Posibles ampliaciones y mejoras

Aunque el proyecto actual ha alcanzado un estado funcional satisfactorio, existen numerosas áreas en las que se pueden realizar ampliaciones y mejoras para enriquecer la experiencia de usuario y aumentar la complejidad y disfrute del juego. A continuación se detallan algunas de estas posibles ampliaciones y mejoras:

1. Multijugador Online

Descripción: Implementar la capacidad de jugar partidas en línea contra otros jugadores en tiempo real.

Desafíos: Necesidad de un servidor de juegos en tiempo real, gestión de la sincronización de estados entre clientes, manejo de conexiones y posibles desconexiones.

2. Inteligencia Artificial (IA)

Descripción: Desarrollar una IA contra la cual los jugadores puedan jugar cuando no haya otros jugadores disponibles.

Desafíos: Crear algoritmos de decisión que simulen un jugador humano, balancear la dificultad de la IA para mantener el juego interesante.

3. Modo Torneo y Tablas de Clasificación

Descripción: Añadir un modo de torneo donde los jugadores compitan en un bracket eliminatorio y tablas de clasificación para mostrar a los mejores jugadores.

Desafíos: Implementar la lógica del torneo, actualización y mantenimiento de las tablas de clasificación en tiempo real, manejo de emparejamientos y resultados.

4. Tutorial y Ayuda en el Juego

Descripción: Incluir un botón o sección de tutorial que explique los principios básicos y las mecánicas del juego.

Desafíos: Crear contenido educativo claro y conciso, integrar tutoriales interactivos que guíen a los jugadores a través de los fundamentos del juego.

5. Habilidades de Cartas

Descripción: Añadir habilidades especiales a las cartas para aumentar la profundidad estratégica del juego.

Desafíos: Definir y equilibrar las habilidades de las cartas, implementar la lógica de las habilidades y su interacción con otras mecánicas del juego.

6. Funcionalidad Completa de los Dones

Descripción: Asegurarse de que los dones (mana) cumplan con todas sus funcionalidades y afecten el juego de manera completa.

Desafíos: Implementar todas las interacciones posibles de los dones, asegurarse de que las reglas de los dones estén bien integradas y balanceadas.

7. Guardado de Partidas en la Base de Datos

Descripción: Implementar la capacidad de guardar el estado de las partidas en la base de datos para que puedan ser retomadas más tarde.

Desafíos: Diseñar la estructura de la base de datos para almacenar el estado del juego, implementar la lógica para cargar y guardar partidas, manejar la persistencia de datos.

8. Mejorar la Responsividad del Juego

Descripción: Cambiar las imágenes de fondo a iframes y hacer que todo el juego sea más responsive.

Desafíos: Rediseñar la interfaz de usuario para adaptarse a diferentes tamaños de pantalla, asegurar que los iframes se comporten correctamente en todos los dispositivos y resoluciones.

Conclusión

Estas ampliaciones y mejoras no solo enriquecerán la experiencia del usuario, sino que también llevarán el juego a un nivel más profesional y competitivo. Implementar estas características puede requerir una inversión considerable de tiempo y recursos, pero los beneficios en términos de funcionalidad, jugabilidad y satisfacción del usuario final justificarán el esfuerzo. Este proyecto ha sido una excelente base, y estas mejoras potenciales pueden ayudar a transformar el juego en una plataforma completa y atractiva para los jugadores.

BIBLIOGRAFÍA Y WEBGRAFÍA

Bibliografía:

- Jorda, T. (s.f.). Apuntes proporcionados por Toni Jorda. Recuperado de https://aules.edu.gva.es/fp/pluginfile.php/7058796/mod_resource/content/1/Apuntes%20REACT%20parte%201.pdf
 - **Descripción:** Apuntes detallados sobre la implementación de aplicaciones en React, incluyendo ejemplos prácticos y explicaciones teóricas.
- Jorda, T. (s.f.). Programar paso a paso una aplicación TODO con REACT. Recuperado de <https://aules.edu.gva.es/fp/mod/assign/view.php?id=5586401>
 - **Descripción:** Tutorial paso a paso para crear una aplicación de lista de tareas con React, abordando temas como la gestión del estado y el uso de componentes.

Webgrafía:

- Smith, T. (2022). ASP NET Web API Tutorial. Recuperado de <https://www.youtube.com/playlist?list=PL82C6-O4XrHdiS10BLh23x71ve9mQCln0> (Accedido el 1 de mayo de 2024)
 - **Descripción:** Serie de videos tutoriales sobre la creación de APIs web con ASP.NET Core, cubriendo conceptos desde básicos hasta avanzados.
- Node.js. (s.f.). Recuperado de <https://nodejs.org/en> (Accedido el 19 de abril de 2024)
 - **Descripción:** Sitio oficial de Node.js, proporcionando documentación y recursos para el desarrollo de aplicaciones utilizando JavaScript en el servidor.
- ChatGPT. (s.f.). ChatGPT. Recuperado de <https://chatgpt.com/> (Accedido el 30 de mayo de 2024)
 - **Descripción:** Herramienta de inteligencia artificial utilizada para asistencia en la generación de contenido y resolución de dudas durante el desarrollo del proyecto sobre todo enfocadas en css y documentacion.

- Draw.io - Free Flowchart Maker and Diagrams Online. (s.f.). Recuperado de <https://app.diagrams.net/#> (Accedido el 28 de mayo de 2024)
 - **Descripción:** Herramienta en línea para la creación de diagramas de flujo y otros gráficos, utilizada para diagramas de arquitectura y casos de uso.
- React. (s.f.). Recuperado de <https://es.react.dev> (Accedido el 10 de abril de 2024)
 - **Descripción:** Sitio oficial de React, ofreciendo documentación y guías para el desarrollo de interfaces de usuario con React.
- Axios. (s.f.). Recuperado de <https://axios-http.com> (Accedido el 10 de mayo de 2024)
 - **Descripción:** Librería para realizar solicitudes HTTP desde el navegador y Node.js, utilizada para la comunicación entre el frontend y el backend en el proyecto.
- One Piece Card Game. (s.f.). Recuperado de <https://en.onepiece-cardgame.com> (Accedido el 1 de abril de 2024)
 - **Descripción:** Sitio oficial del juego de cartas One Piece, utilizado como referencia temática y de mecánicas para el desarrollo del proyecto