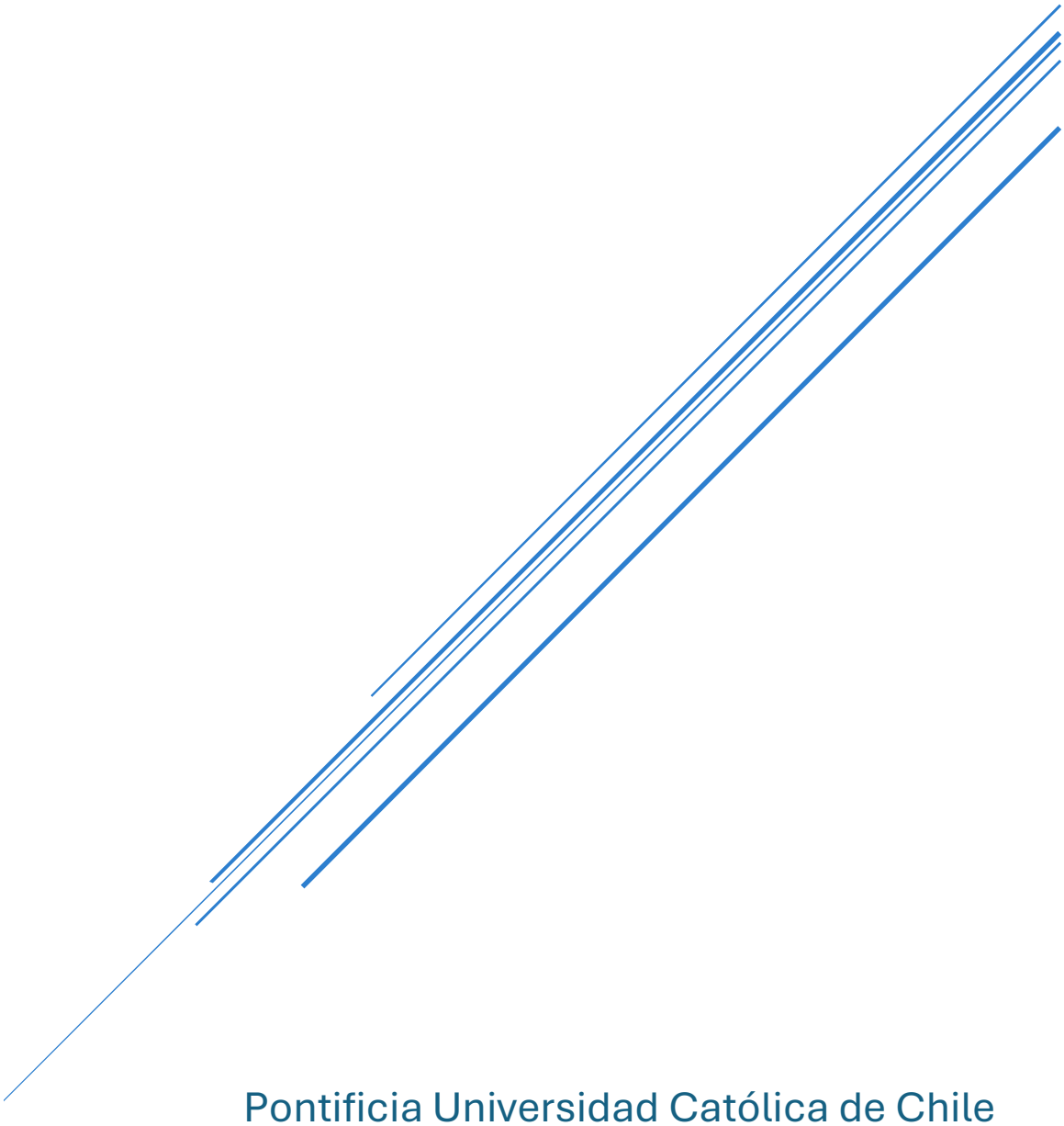
ENTREGA 4 – PROYECTO GRUPAL

Equipo de desarrollo de DuckZone



Pontificia Universidad Católica de Chile
IIC2513 - Tecnologías y aplicaciones web

Índice:

1.1. Seteo de la base de datos en local

En esta entrega es necesario para configurar la base de datos, primero, dropearla. Esto es debido a cambios importantes dentro del esquema. Para hacer esto, usamos el comando:

```
czsanchez@Mimi-laptop:~/WEB/back-DuckZone$ dropdb duckzonedb development
```

Luego, hay que volver a crearla. Para crear la base de datos, primero necesitamos crear el archivo que definirá el enviroment. En este, se necesita crear en vscode, dentro de la carpeta principal del repositorio un archivo .env que contenga la siguiente información:

```
BD_USERNAME=grupodz
DB_PASSWORD=patitos
DB_NAME=duckzonedb
DB_HOST=localhost
JWT_SECRET=jwt_secret
```

Una vez realizado esto, necesitamos crear la base de datos. Para ello, dentro de la terminal, se ejecuta el comando:

```
czsanchez@Mimi-laptop:~/WEB/back-DuckZone$ createdb duckzonedb_development
```

Una vez hecho esto, necesitamos setear el usuario que tenemos en el .env. Para esto, ejecutamos en terminal:

```
czsanchez@Mimi-laptop:~/WEB/back-DuckZone$ sudo -u postgres psql
```

Esto debería abrirnos la consola de postgresql con permisos. En ella, escribimos los siguientes comandos, que crearán el user y le darán permisos sobre la base de datos:

```
postgres=# CREATE ROLE grupodz WITH LOGIN PASSWORD 'patitos';
postgres=# \c duckzonedb development
postgres=# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO grupodz;
```

Una vez hecho esto, salimos de la consola de postgres con \q. Con eso quedan los settings iniciales de la base de datos. Ahora, corremos las migraciones y los seeders con las siguientes líneas en la terminal:

```
czsanchez@Mimi-laptop:~/WEB/back-DuckZone$ yarn sequelize-cli db:migrate
```

```
czsanchez@Mimi-laptop:~/WEB/back-DuckZone$ yarn sequelize-cli db:seed:all
```

Con eso, la base de datos queda creada con sus respectivas tablas, asociaciones, entre otros.

1.2. Visualizar deploy de la aplicación

Para poder acceder a la aplicación, se debe ingresar al siguiente link: https://duckzone.netlify.app

2. Documentación de endpoints

Analizamos los endpoints de cada modelo:

/players

- **POST /players.create**: Crea un nuevo jugador.
    - **Request Body**: Datos del jugador.
    - **Response**: El jugador creado o un error si el jugador ya existe.
- **GET /**: Obtiene la lista de todos los jugadores.
    - **Response**: Lista de jugadores.
- **GET /:id**: Obtiene la información de un jugador específico.
    - **Params**: id - ID del jugador.
    - **Response**: Información del jugador o error si no se encuentra.
- **GET /:id/cards**: Obtiene la mano del jugador.
    - **Params**: id - ID del jugador.
    - **Response**: Mano del jugador o error si no se encuentra.
- **PATCH /:id/refill_hand**: Rellena las cartas de la mano del jugador
    - **Params**: id - ID del jugador.
    - **Response**: Información del jugador o error si no se encuentra.
- **PATCH /:id/empty_hand**: Elimina las cartas de la mano del jugador.
    - **Params**: id - ID del jugador.
    - **Response**: Información del jugador o error si no se encuentra.
- **PATCH /:id/remove_card/:cardNumber**: Elimina las cartas de la mano del jugador.
    - **Params**: id - ID del jugador, cardNumber - número de la carta
    - **Response**: Información del jugador o error si no se encuentra.
- **PATCH /:id/play_duck/:cardNumber** Juega una carta de la mano del jugador y actualiza el mana

    - **Params**: id - ID del jugador , cardNumber – número de la carta
    - **Response**: Información de la carta jugada o error si no se encuentra.
- **PATCH /:id/play_spell/:cardNumber** Juega una carta hechizo de la mano del jugador.
    - **Params**: id - ID del jugador, cardNumber – número de la carta
    - **Response**: Información de la carta hechizo jugada o error si no se encuentra.
- **PATCH /:id/update_mana** Actualiza el maná del jugador.
    - **Params**: id - ID del jugador
    - **Response**: La entidad player actualizada
- **DELETE /** Elimina los jugadores
    - **Params**: ninguno (ruta general)
    - **Response**: Información del status de la request.
- **PATCH /updateuser** Actualiza a la instancia de jugador.
    - **Params**: id - ID del jugador
    - **Response**: Instancia de jugador actualizada.

/cards

- **GET /**: Obtiene la lista de todas las cartas.
    - **Response**: Lista de cartas.
- **GET /:id**: Obtiene la información de una carta específica.
    - **Params**: id - ID de la carta.
    - **Response**: Información de la carta o error si no se encuentra.

/games

- **POST / games.create**: Crea un nuevo juego.
    - **Request Body**: Datos del juego.

    - **Response**: El juego creado o un error si ocurre un problema.
- **PATCH/**save_card1/:id Guarda la primera carta jugada en la partida especificada, del player 1
    - **Request Body**: ID – id de la Carta
    - **Response**: La partida creada o un error si ocurre un problema.
- **PATCH/**save_card2/:id Guarda la segunda carta jugada en la partida especificada, del player 2
    - **Request Body**: ID – id de la Carta
    - **Response**: La partida creada o un error si ocurre un problema.
- **PATCH/**save_spell/:id : Guarda una carta hechizo en la partida especificada
    - **Request Body**: ID – id de la Carta
    - **Response**: El juego creado o un error si ocurre un problema.
- **PATCH/**resolve/:id Resuelve el combate/jugada entre dos cartas en la partida especificada
    - **Params:** Id del jugador
    - **Response**: El juego creado o un error si ocurre un problema.
- **PATCH/**update_player_two/:id : Actualiza el segundo jugador para unirlo a la partida
    - **Request Body**: id- ID del juego.
    - **Response**: La información del jugador o un error si ocurre un problema.
- **PATCH/**update_player_count/:id : Actualiza el conteo de jugadores en la partida
    - **Request Body**: id- ID del juego.
    - **Response**: El juego creado o un error si ocurre un problema.
- **PATCH/**start/:id : Marca la partida especificada
    - **Request Body**: id- ID del juego.
    - **Response**: El juego creado o un error si ocurre un problema.

- **GET /**: Obtiene todas las partidas
    - **Response**: Listado de todas las partidas o error si no se encuentra.
- **GET /:id**: Obtiene la información de una partida específica.
    - **Params**: id - ID del juego.
    - **Response**: Información del juego o error si no se encuentra.
- **DELETE /**: Elimina todas las partidas y jugadores asociados.
    - **Response**: Información de las partidas eliminadas.
- **DELETE /:id**: Elimina a una partida en específico y jugadores asociados.
    - **Params**: id - ID del juego.
    - **Response**: Información de la partida eliminada
- **PATCH /start/:id**: Comienza una partida
    - **Params**: id - ID del juego.
    - **Response**: Información de la partida eliminada
- **PATCH /updateplayedcards/:id** Actualiza las cartas jugadas para los jugadores asociados.
    - **Params**: id - ID del juego.
    - **Response**: Información de la partida eliminada

/reports

- **POST /**: Crea un reporte
    - **Request Body**: Datos del reporte.
    - **Response**: Lista de reportes.
- **PATCH /:id/check**: Actualiza el estado de un reporte
    - **Request Body**: ID del reporte
    - **Response**: Reporte actualizado.

- **GET /:id**: Obtiene la información de un reporte específico.
    - **Params**: id - ID del reporte.
    - **Response**: Información del reporte o error si no se encuentra.
- **GET /**: Obtiene la información de todo reporte.
    - **Response**: Lista de todos los reportes.
- **GET /user/report-counts**: Cuenta cuántos reports existen por cada user
    - **Response**: Lista de usuarios, ids, status y la cantidad de reportes.

/auth

- **POST /login**: Autentica a un usuario.
    - **Request Body**: Credenciales del usuario.
    - **Response**: Token JWT o error si las credenciales son incorrectas.
- **POST /signup**: Registra un nuevo usuario.
    - **Request Body**: Datos del usuario.
    - **Response**: El usuario registrado o error si ya existe.

/users

- **GET /**: Obtiene la lista de todos los usuarios.
    - **Response**: Lista de usuarios.
- **GET /:id**: Obtiene la información de un usuario específico.
    - **Params**: id - ID del usuario.
    - **Response**: Información del usuario o error si no se encuentra.
- **GET /current**: Obtiene la información del usuario con sesión iniciada.
    - **Response**: Información del usuario o error si no se encuentra.
- **PATCH /:id/ban**: Banea a un usuario específico.
    - **Params**: id - ID del usuario a banear.
    - **Response**: Información del usuario baneado o error si no se encuentra.

- **PATCH /:id/unban**: Activa la cuenta de un usuario específico.
    - **Params**: id - ID del usuario a activar.
    - **Response**: Información del usuario activado o error si no se encuentra.
- **GET /api/leaderboard**: Muestra cuales son los mejores jugadores con relación a partidas ganadas y perdidas.
    - **Response**: Información del leaderboard con los jugadores que se encuentran entre los 10 primeros.

/profile

- **PUT /editprofile**: Edita el perfil del usuario autenticado.
    - **Request Body**: Datos a editar del usuario y contraseña actual
    - **Response**: Perfil actualizado o error

3. ¿Cómo inicio los servers en local?

Para un correcto testeo de esta entrega es necesario que ambos frontend y backend estén corriendo con tal de que se pueda establecer una conexión exitosa. Para esto, en la terminal se deben ejecutar los siguientes comandos para la ejecución del frontend:

```
czsanchez@Mimi-laptop:~/WEB/front-DuckZone$ cd frontdz/
czsanchez@Mimi-laptop:~/WEB/front-DuckZone/frontdz$ yarn dev
```

En el caso del backend, basta solamente con ejecutar:

```
czsanchez@Mimi-laptop:~/WEB/back-DuckZone$ yarn dev
```

PD: Seguimos el seteo paso a paso de las capsulas. Si llegase a ser necesario modificar el directorio para el setup de render, lo haremos. Temporalmente, la conexión de esta manera sirve.

4. Mockup actualizado

Debido a que el diseño no sufrió muchas modificaciones por los comentarios del ayudante, lo dejamos como estaba inicialmente. De igual manera, el mockup se puede encontrar en la carpeta de documentos de una entrega pasada.

De todas maneras, dejamos una breve comparación mockup vs html realizado respectivamente a continuación: