Documentación de las llamadas a API

Parte 1: AWS Lambda

1. Generar PDF

- **Método:** POST
- **Endpoint:** https://nfu5ofywqc.execute-api.us-east-1.amazonaws.com/dev/pdf

Solicitud

- **Headers:**
  - Content-Type: application/json

- **Cuerpo (Body):**

```
{
  "groupName": "Nombre del grupo",
  "userName": "Nombre del usuario",
  "teamsInfo": ["Equipo 1", "Equipo 2", "Equipo 3"]
}
```

Respuesta Exitosa

- **Código de Estado:** 200 OK
- **Cuerpo:**

```
{
  "message": "PDF generado exitosamente",
  "url": "https://boletas-de-compra.s3.amazonaws.com/nombre-usuario-fecha.pdf"
}
```

Respuesta de Error

- **Código de Estado:** 500 Internal Server Error
- **Cuerpo:**

```
{
  "error": "Error al subir el PDF a S3",
  "details": "Detalles del error"
}
```

2. Obtener PDFs

- **Método:** GET
- **Endpoint:** https://nfu5ofywqc.execute-api.us-east-1.amazonaws.com/dev/pdfs

Solicitud

- **Query Parameters:**
  - userName: El nombre del usuario para el que se desean obtener los PDFs.

Respuesta Exitosa

- **Código de Estado:** 200 OK
- **Cuerpo:**

```
{
  "pdfs": [
    {
      "date": "2024-10-27T15:00:00.000Z",
      "url": "https://boletas-de-compra.s3.amazonaws.com/nombre-usuario-archivo.pd
    },
    ...
  ]
}
```

Respuesta de Error

- **Código de Estado:** 500 Internal Server Error
- **Cuerpo:**

```
{
  "error": "Error al listar PDFs: Detalles del error"
}
```

Funciones en AWS Lambda

- generatePdf

Esta función se encarga de generar un PDF con la información de la compra y subirlo al bucket de S3. Toma como parámetros el nombre del grupo, el nombre del usuario y la información de los equipos.

Parámetros

- groupName: Nombre del grupo.
- userName: Nombre del usuario.
- teamsInfo: Arreglo de nombres de equipos.

Proceso

1. Sanitiza el nombre del usuario reemplazando los espacios por guiones.
2. Genera un PDF utilizando la biblioteca PDFKit.
3. Sube el PDF al bucket de S3.
4. Devuelve la URL del PDF generado.

- getPdfs

Esta función recupera los PDFs de un usuario específico desde el bucket de S3.

Parámetros

- userName: Nombre del usuario para el cual se desean obtener los PDFs.

Proceso

1. Sanitiza el nombre del usuario reemplazando los espacios por guiones.
2. Lista los objetos en el bucket de S3 que coinciden con el prefijo del nombre del usuario.
3. Crea un array con la fecha y la URL de cada PDF.

4. Ordena los PDFs por fecha (de más reciente a más antiguo).
5. Devuelve la lista de PDFs.

Parte 2: API Django (backend en servidor)

Modelos utilizados

La API utiliza los siguientes modelos de Django:

- User: Representa a un usuario del sistema.
- Fixture: Representa un partido deportivo (fixture).
- Bonos: Representa un bono comprado por un usuario para un partido.
- Recommendation: Representa una recomendación de compra para un usuario.

Endpoints

Usuarios

- /users (GET): Permite obtener una lista de todos los usuarios.
- /users (POST): Permite crear un nuevo usuario.
- /users/<user_id> (GET): Permite obtener un usuario específico por su user_id.
- /users/<user_id> (PUT): Permite actualizar un usuario específico.
- /users/<user_id> (DELETE): Permite borrar un usuario específico.

Créditos (Wallet)

- /wallet/add (PATCH): Permite agregar dinero a la billetera (wallet) de un usuario.

Partidos (Fixtures)

- /fixtures (GET): Permite obtener una lista de partidos futuros (filtrables por equipo local, equipo visitante y fecha).
- /fixtures (POST): Permite crear un nuevo partido.
- /fixtures/<fixture_id> (GET): Permite obtener un partido específico por su fixture_id.

- /fixtures/<fixture_id> (PUT): Permite actualizar un partido específico.

Bonos

- /bonos (GET): Permite obtener una lista de todos los bonos. **(En desarrollo)**
- /bonos (POST): Permite comprar bonos para un partido.

Esta vista permite dos métodos de compra:

1. Pago con dinero de la billetera (wallet):

    - Se valida que el usuario tenga suficiente dinero en su billetera.

    - Se descuenta el costo total de los bonos de la billetera del usuario.

    - Se publica la solicitud de bono en el canal MQTT `fixtures/request`.

    - Se envía información al job_master para calcular recomendaciones.

2. Pago con Webpay Plus:

    - Se crea una transacción con Webpay Plus en modo integración.

    - Se devuelve un token y una URL para que el usuario realice el pago.

MQTT

- /mqtt/requests (POST): Permite almacenar solicitudes de compra de bonos desde el canal fixtures/requests de MQTT.
- /mqtt/validations (PUT): Permite procesar validaciones de solicitudes de bonos desde el canal fixtures/validation de MQTT.
    - Si la validación es positiva, se marca la compra como aprobada.
    - Si la validación es negativa, se devuelven los bonos al fixture y se marca la compra como rechazada.
- /mqtt/history (POST): Permite procesar el historial de resultados de partidos desde un cliente MQTT.
    - Busca las fixtures correspondientes a los partidos del historial.
    - Determina el resultado del partido según los goles anotados.
    - Busca los bonos relacionados con cada fixture.
    - Verifica si el resultado del bono coincide con el del partido.

    - Si el resultado coincide, se acredita el monto ganado al usuario.
    - De lo contrario, se marca el bono como perdido.

Recomendaciones de compra

- /store_recommendation (POST): Permite crear una recomendación de compra.