Desafío - FutScript

En este desafío validaremos nuestros conocimientos de Express para la creación de una API REST conectada a una base de datos PostgreSQL y Testing Unitario con Jest. Para lograrlo, necesitarás completar el desarrollo de un servidor que encontrarás en el ***Apoyo Desafío - FutScript***.

Lee todo el documento antes de comenzar el desarrollo **grupal**, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Descripción

FutScript es un proyecto iniciado por 5 aficionados de la programación que tiene como objetivo crear un sistema de administración para las distintas escuelas de fútbol de la ciudad.

En este desafío deberás continuar el desarrollo de una aplicación Express basada en la temática planteada, agregándole lo necesario para cumplir los requerimientos.

El desarrollo de esta API REST debe continuar agregando:

- Una ruta **POST /login** que espere recibir en el payload las credenciales de un usuario administrador
    - El único usuario administrador es el siguiente:

```
{
    "username": "admin",
    "password": "1234"
}
```

    En caso de recibir correctamente estas credenciales se debe responder la consulta con **JWT**

- En las 2 rutas **POST** de la API REST para registrar nuevos equipos y jugadores se debe agregar un middleware que obtenga y valide un token ubicado en las cabeceras de la consulta

- En las 2 Rutas **GET**:
    - **/equipos**

      Se debe retornar los registros de los equipos siguiendo la siguiente estructura de ejemplo:

```
[
  {
    "id": 7,
    "name": "Real Madrid"
  },
  {
    "id": 8,
    "name": "Barcelona"
  },
]
```

    - **/equipos/:teamID/jugadores**

      Se debe retornar la siguiente estructura de ejemplo:

```
[
  {
    "name": "Karim Benzema",
    "posicion": "delantero"
  },
  {
    "name": "Luka Modrić",
    "posicion": "centrocampista"
  }
]
```

      Para lograrlo deberás realizar una consulta SQL con INNER JOIN

Las tablas en la base de datos son las siguientes:

Posiciones	
PK	id
	name

Jugadores	
PK	id
FK	id_equipos
FK	position
	name

Equipos	
PK	id
	name

Dentro del material de apoyo encontrarás un archivo *script.sql* que contiene las consultas para crear estas entidades y sus relaciones.

Tests

Deberás crear los siguientes supertest:

- Se obtiene un **Array** y un status code 200 como respuesta de la ruta **GET /equipos**
- Al enviar las credenciales correctas en la ruta **POST /login** se obtiene un **Object**
- Al enviar credenciales incorrectas en la ruta **POST /login** se obtiene un status code **400**
- Al enviar un nuevo jugador en la ruta **POST /equipos/:teamID/jugadores** junto a un token válido en las cabeceras se obtiene un status code 201.

Requerimientos

1. Utilizar JWT para la autorización de usuarios (**2 Puntos**)
2. Utilizar el paquete pg para la comunicación y gestión de una base de datos en una aplicación Node Js (**2 Puntos**)
3. Realizar consultas SQL para la obtención e inserción de datos desde Node (**2 Puntos**)
4. Utilizar correctamente los códigos de estado HTTP en las situaciones según correspondan (**1 Puntos**)
5. Realizar tests unitarios usando el paquete supertest (**3 Puntos**)

 ¡Mucho éxito!

Consideraciones y recomendaciones

- Lee detalladamente el código del servidor y este documento para comprender a detalle qué debes hacer
- Entre los archivos del material de apoyo encontrarás un JSON que contiene una colección de Requests con las diferentes rutas del servidor. Impórtalo en la pestaña **Collections** de la extensión **Thunder Client** para revisar cómodamente tus avances.
- Utiliza la variable *secretKey* del archivo *utils* para firmar y verificar los tokens
- Recuerda que si deseas trabajar con la sintaxis de módulos, debes agregar al package.json del material de apoyo para este desafío, el `'type: module'`.