Proyecto Final: Desarrollo e Implementación de API RESTful "FutPlay"

Asignatura: Programación Orientada a Objetos
Estudiante: Néyfer Perdomo

---

1. Introducción

El presente documento detalla el proceso de análisis, diseño, desarrollo e implementación de la API RESTful "FutPlay". [cite_start]Este proyecto se enmarca en la asignatura de Programación Orientada a Objetos y tiene como objetivo principal aplicar conceptos avanzados de desarrollo de software, incluyendo la creación de una API con FastAPI, la gestión de una base de datos NoSQL con MongoDB y la implementación de un sistema robusto de autenticación y autorización de usuarios mediante Firebase Authentication[cite: 24130, 24140].

[cite_start]La API FutPlay está diseñada para gestionar la información de torneos de fútbol, abarcando entidades como equipos, jugadores, partidos y eventos, siguiendo una arquitectura de microservicios y aplicando las mejores prácticas de la industria[cite: 24130]. [cite_start]El desarrollo se guió por las especificaciones del instructor, la retroalimentación recibida sobre el modelo de datos y el análisis de un proyecto de referencia (`dulceria.api`) para adoptar un estilo de codificación consistente y una arquitectura modular[cite: 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257].

---

2. Objetivos de Aprendizaje

[cite_start]De acuerdo con los requerimientos del proyecto, al finalizar esta implementación se lograron los siguientes objetivos de aprendizaje[cite: 24140, 24141]:

* [cite_start]**Implementar autenticación y autorización con JWT** para proteger las rutas y gestionar el acceso basado en roles[cite: 24141].
* [cite_start]**Diseñar y desarrollar APIs RESTful** siguiendo los principios y estándares del mercado utilizando FastAPI[cite: 24141].
* [cite_start]**Utilizar MongoDB y sus pipelines de agregación** para realizar consultas complejas y eficientes que involucren múltiples colecciones[cite: 24141].
* [cite_start]**Aplicar patrones de arquitectura de software** como el Modelo-Vista-Controlador (MVC) para una correcta separación de responsabilidades[cite: 24141].
* [cite_start]**Validar datos de entrada y salida** de manera rigurosa utilizando Pydantic para garantizar la integridad de la información[cite: 24141].
* [cite_start]**Implementar relaciones entre colecciones NoSQL** para modelar de forma efectiva las interacciones entre las distintas entidades del sistema[cite: 24141].
* [cite_start]**Usar decoradores de Python** para la implementación de capas de seguridad y lógica transversal[cite: 24141].
* [cite_start]**Manejar errores y validaciones de negocio** de forma centralizada para ofrecer una experiencia consistente al cliente de la API[cite: 24141].

---

3. Arquitectura y Estructura del Proyecto

[cite_start]Para asegurar la escalabilidad, mantenibilidad y coherencia del código, se adoptó una estructura de proyecto inspirada en el repositorio de referencia `dulceria.api`, la cual sigue el principio de separación de capas[cite: 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239,

1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257].

[cite_start]La estructura de carpetas final del proyecto `FutPlay` fue la siguiente[cite: 24158]:

```
my_api/
 ├─── app/
 |    ├─── main.py             # Punto de entrada de la aplicación FastAPI
 |    ├─── config.py           # Configuración y variables de entorno
 |    ├─── database.py         # Módulo de conexión a MongoDB
 |    ├─── auth/               # Lógica de autenticación con Firebase y JWT
 |    ├─── controllers/        # Lógica de negocio (separada de las rutas)
 |    ├─── models/             # Modelos de datos y esquemas Pydantic
 |    ├─── routes/             # Definición de los endpoints de la API
 |    ├─── pipelines/          # Pipelines de agregación de MongoDB
 |    └─── utils/              # Funciones utilitarias (seguridad, errores)
 ├─── requirements.txt        # Dependencias del proyecto
 ├─── .env                    # Archivo de variables de entorno (local)
 └─── README.md               # Documentación del proyecto
```

---

4. Tecnologías Utilizadas

[cite_start]El desarrollo del proyecto se basó en un stack tecnológico moderno y eficiente, alineado con los requerimientos de la asignación[cite: 24143]:

* **FastAPI:** Framework web de alto rendimiento para la construcción de APIs con Python.
* **MongoDB:** Base de datos NoSQL orientada a documentos, utilizada por su flexibilidad y escalabilidad.
* **Motor (Driver Asíncrono):** Para la comunicación no bloqueante con MongoDB.
* **Pydantic:** Para la validación de datos, la configuración y la definición de esquemas.
* **Firebase Authentication:** Servicio de Google para gestionar la autenticación de usuarios de forma segura.
* **JSON Web Tokens (JWT):** Para la creación de tokens de sesión que permiten la autorización de endpoints.
* **Uvicorn:** Servidor ASGI (Asynchronous Server Gateway Interface) para ejecutar la aplicación FastAPI.

---

5. Desarrollo de la API

5.1. Modelo de Datos

El modelo de datos fue refinado a partir de la retroalimentación del instructor para mejorar la normalización y la flexibilidad del sistema. [cite_start]Las entidades principales son[cite: 24135, 24136]:

* **Usuarios:** Gestiona la información básica del usuario, con `uid_firebase` como clave principal.
* **Tipos de Torneo:** Colección "maestra" para categorizar los torneos (ej: Liga, Copa).

  * **Torneos:** Entidad principal que contiene la información de cada torneo y se relaciona con un `tipo_id`.
  * **Equipos:** Almacena los equipos, asociados a un torneo.
  * **Entrenadores:** Colección separada para gestionar a los entrenadores y su historial.
  * **Jugadores:** Contiene los datos de los jugadores, con un historial de equipos para manejar relaciones muchos a muchos.
  * **Partidos:** Registra los partidos, vinculados a un torneo, equipos y lugar.
  * **Lugares:** Colección para almacenar estadios o canchas.
  * **Tipos de Evento:** Colección "maestra" para definir eventos (Gol, Asistencia, Tarjeta Amarilla, etc.).
  * **Eventos:** Registra los sucesos ocurridos en un partido, vinculando un jugador, partido y tipo de evento.

5.2. Implementación de Endpoints

[cite_start]Se implementó un total de 16 endpoints para cumplir con todos los requisitos de la evaluación[cite: 24147, 24148]:

1. Sistema de Autenticación (2 Endpoints):

  * `POST /users`: Registro de un nuevo usuario en Firebase y almacenamiento en MongoDB.
  * `POST /login`: Autenticación de usuario con Firebase y generación de un token JWT.

2. CRUD Modelo Maestro/Tipo - `tipos_torneo` (5 Endpoints):

  * `GET /tipos-torneo`: Listar todos los tipos de torneo.
  * `POST /tipos-torneo`: Crear un nuevo tipo de torneo (protegido).
  * `GET /tipos-torneo/{id}`: Obtener un tipo de torneo específico.
  * `PUT /tipos-torneo/{id}`: Actualizar un tipo de torneo (protegido).
  * `DELETE /tipos-torneo/{id}`: Eliminar un tipo de torneo, con validación para no permitir el borrado si está en uso (protegido).

3. CRUD Modelo Principal - `torneos` (5 Endpoints):

  * `GET /torneos`: Listar todos los torneos.
  * `POST /torneos`: Crear un nuevo torneo (protegido).
  * `GET /torneos/{id}`: Obtener los detalles de un torneo.
  * `PUT /torneos/{id}`: Actualizar un torneo (protegido).
  * `DELETE /torneos/{id}`: Eliminar un torneo (protegido).

4. Endpoints con Pipelines Obligatorios (3 Endpoints):

  * **Pipeline 1 (`$lookup`):** Un endpoint que combina información de las colecciones de `partidos`, `equipos` y `torneos` para mostrar un resumen detallado de los partidos con los nombres de los equipos en lugar de sus IDs.
  * **Pipeline 2 (`$group`):** Un endpoint que calcula estadísticas, como el total de eventos (goles, asistencias) agrupados por jugador, utilizando operadores como `$group` y `$sum`.
  * **Pipeline 3 (Validación Compleja):** Se implementó una validación dentro de un endpoint `POST` para impedir el registro de un partido si un equipo ya tiene otro partido programado en la misma fecha, utilizando un pipeline para consultar los datos existentes antes de la inserción.

5. Endpoint con QueryString (1 Endpoint):

  * `GET /jugadores/buscar`: Un endpoint público (sin token) que permite filtrar jugadores por parámetros opcionales en la URL, como `?posicion=delantero&edad=22`, para demostrar el manejo de Query Parameters en FastAPI.

---

6. Conclusiones

El desarrollo del proyecto FutPlay permitió la aplicación práctica e integral de los conocimientos teóricos adquiridos en la asignatura. Se logró construir una API RESTful funcional, segura y bien estructurada, cumpliendo con todos los requerimientos técnicos y de negocio solicitados.

El uso de FastAPI facilitó un desarrollo rápido y moderno, mientras que MongoDB proporcionó la flexibilidad necesaria para un modelo de datos deportivo. La implementación de autenticación con Firebase y la protección de rutas mediante JWT aseguró un sistema robusto. Finalmente, el proyecto refuerza la importancia de una arquitectura limpia y la separación de responsabilidades para crear software mantenible y escalable.