Relatório do Projeto: Sistema de Gerenciamento de Jogadores de Futebol

Murilo dos Santos Cunha

23 de janeiro de 2025

1 Introdução

Este relatório descreve o desenvolvimento de uma API para gerenciamento de jogadores de futebol, utilizando FastAPI, SQLModel e PostgreSQL. O objetivo do projeto é aplicar conceitos de Mapeamento Objeto-Relacional (ORM) e fornecer funcionalidades CRUD completas, além de suporte à paginação, filtros e migrações de banco de dados.

2 Objetivos

O projeto visa:

- Implementar uma API RESTful para gerenciar jogadores de futebol.
- Utilizar SQLModel para mapeamento objeto-relacional.
- Realizar operações CRUD (Create, Read, Update, Delete) em um banco de dados PostgreSQL.
- Implementar funcionalidades como paginação, filtros e consultas complexas.
- Configurar migrações de banco de dados com Alembic.
- Adicionar logs para monitoramento das operações.

3 Desenvolvimento

3.1 Configuração do Ambiente

O projeto foi desenvolvido utilizando:

- **FastAPI**: Framework para construção da API.
- **SQLModel**: Biblioteca para ORM e interação com o banco de dados.

- **PostgreSQL**: Banco de dados relacional.
- **Alembic**: Ferramenta para migrações de banco de dados.

3.2 Estrutura do Projeto

O projeto foi organizado da seguinte forma:

- **main.py**: Arquivo principal da aplicação, contendo os endpoints da API.
- **models.py**: Definição das classes de modelo (Jogador e Estatisticas).
- **alembic/**: Configurações e migrações do banco de dados.
- **database/**: Criação de banco de dados ficticio.
- **utils/**: Configuração e inserção dos dados no banco.

3.3 Modelagem do Banco de Dados

Foram definidas duas entidades principais:

- **Jogador**: Representa um jogador de futebol, com atributos como nome, data de nascimento, posição e número da camisa.
- **Estatisticas**: Armazena as estatísticas de um jogador, como gols, assistências, defesas, etc.

3.4 Endpoints da API

A API oferece os seguintes endpoints:

- **Criar Jogador**: `POST /jogadores/`
- **Listar Jogadores**: `GET /jogadores/`
- **Contar Jogadores**: `GET /jogadores/contar`
- **Filtrar Jogadores por Posição**: `GET /jogadores/filtrar/{posicao}`
- **Filtrar Jogadores por Ano de Nascimento**: `GET /jogadores/filtrar/ano/{ano}`
- **Atualizar Jogador**: `PUT /jogadores/{jogador_id}`
- **Excluir Jogador**: `DELETE /jogadores/{jogador_id}`
- **Buscar Jogador por ID**: `GET /jogadores/{jogador_id}`
- **Calcular Melhor Escalação**: `GET /escalação/`

3.5 Exemplo de Código

Abaixo está um trecho do código principal da API:

```python
from fastapi import FastAPI, HTTPException
from sqlmodel import Session, select, create_engine, SQLModel
from typing import List
from pydantic import BaseModel
from models import Jogador, Estatisticas

app = FastAPI()

DATABASE_URL = "postgresql://postgres:1234@localhost/soccer"
engine = create_engine(DATABASE_URL)
SQLModel.metadata.create_all(engine)

class JogadorCreate(BaseModel):
    nome: str
    data_nascimento: str
    posicao: str
    numero_camisa: int

@app.post("/jogadores/", response_model=JogadorCreate)
def create_jogador(jogador: JogadorCreate):
    db_jogador = Jogador(**jogador.dict())
    with Session(engine) as session:
        session.add(db_jogador)
        session.commit()
        session.refresh(db_jogador)
    return db_jogador
```

4 Conclusão

O projeto foi desenvolvido com sucesso, atendendo a todos os requisitos propostos. A API permite o gerenciamento completo de jogadores de futebol, com funcionalidades avançadas como paginação, filtros e consultas complexas. Além disso, a integração com Alembic e a configuração de logs garantem a manutenibilidade e escalabilidade do sistema.

5 Referências

- Documentação do FastAPI: `https://fastapi.tiangolo.com/`
- Documentação do SQLModel: `https://sqlmodel.tiangolo.com/`
- Documentação do Alembic: `https://alembic.sqlalchemy.org/`