PassaBola

Aplicação backend em FastAPI para gerenciar campeonatos, partidas e participação de jogadoras/es. Inclui autenticação via JWT, controle de acesso por função (user/admin), e endpoints para criação/listagem de campeonatos, inscrição, agendamento de jogos e consulta das partidas do usuário.

Sumário

Tecnologias

- Python 3.13
- FastAPI
- SQLAlchemy ORM
- SQLite (desenvolvimento)
- JWT (JSON Web Token)
- CORS Middleware

Arquitetura e Estrutura

Pastas principais:

```
PassaBola/
  core/            # Config, segurança (JWT), dependências (current_user/admin)
  crud/            # Camada de acesso a dados por recurso
  db/              # Engine, Session, Base e utilidades de banco
  middleware/      # Middlewares (ex.: autenticação)
  models/          # Modelos SQLAlchemy
  routers/         # Rotas FastAPI (auth, championship, match/games)
  schemas/         # Schemas Pydantic (entrada/saída)
  main.py          # Aplicação FastAPI
  .env             # Variáveis de ambiente (desenv)
  pyproject.toml   # Dependências/projeto
```

`README.md`

Modelo de domínio (simplificado): - User: conta do usuário (com role admin opcional). - Championship: campeonato; possui campo `is_closed` (aberto quando false). - Match: partida; `status` em [`scheduled`, `in_progress`, `finished`]. - UserChampionship: associação usuário-campeonato. - UserMatch: associação usuário-partida.

Primeiros Passos

1) Instale Python 3.13.
2) Instale uv
3) Rode uv sync
4) Rode uvicorn main:app ## Configuração de Ambiente

Crie um arquivo `.env` na raiz (já há um exemplo no projeto). Variáveis usadas:

```
DATABASE_URL=sqlite:///./database.db
API_PREFIX=/api              # opcional, se aplicável
DEBUG=True
ALLOWED_ORIGINS=https://localhost:3000,https://localhost:5173,http://localhost:5173
```

Executando o Projeto

Inicie a API (hot-reload):

```
bash
uvicorn main:app --reload
```

Acesse: - Documentação Swagger: http://localhost:8000/docs - Redoc: http://localhost:8000/redoc - Health check: GET http://localhost:8000/health

Na inicialização, as tabelas são criadas automaticamente.

Autenticação e Autorização

- Autenticação via Bearer token (JWT).
- Todas as rotas, exceto `/auth/*`, são protegidas e exigem token.
- Endpoints administrativos também exigem que o usuário seja admin.

Fluxo: 1) Signup: `POST /auth/signup` 2) Login: `POST /auth/login` => retorna `access_token` 3) Enviar header Authorization nos pedidos protegidos: `Authorization: Bearer <access_token>`

API

Principais endpoints (resumo):

Auth (público) - POST /auth/signup - POST /auth/login - GET /auth/me

Utilidades - GET /health (protegido - requer token)

Championships (protegidos) - GET /championships - Query: `status` (open|closed|ongoing|completed), `q` (busca pelo nome), `page`, `page_size` - Retorna: lista paginada com `participants_count` - GET /championships/{championship_id} - Detalhe de um campeonato (+ contagem de participantes) - POST /championships/{championship_id}/join - Usuário atual ingressa se `is_closed == false`

Admin (protegidos e admin) - POST /championships - Cria campeonato (status aberto; `is_closed=false`) - POST /championships/{championship_id}/close_signups - Fecha inscrições (`is_closed=true`) e pode gerar partidas iniciais (agendadas) - PATCH /games/{game_id}/schedule - Agenda/atualiza data e local de uma partida (status -> scheduled) - PATCH /games/{game_id}/score - Registra placar e marca como concluída (status -> finished). Observação: se não houver colunas de placar no modelo, a API aceita valores e retorna no payload, mas não persiste.

Jogos (protegidos) - GET /championships/{championship_id}/games - Lista partidas de um campeonato (paginado) - GET /games/{game_id} - Detalhe de uma partida - GET /me/games - Query: `status` (upcoming|completed) - Lista partidas do usuário (com base em `UserMatch`), paginado

Exemplos de requisição (curl)

Login:

```
curl -X POST http://localhost:8000/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"user@example.com","password":"secret"}'
```

Listar campeonatos:

```
curl -H "Authorization: Bearer <TOKEN>" http://localhost:8000/championships?page=1&page_size
```

Entrar em campeonato:

```
curl -X POST -H "Authorization: Bearer <TOKEN>" http://localhost:8000/championships/1/join
```

Agendar jogo (admin):

```
curl -X PATCH http://localhost:8000/games/10/schedule \
  -H "Authorization: Bearer <ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{"date":"2025-09-20T15:00:00Z","location":"Estádio A"}'
```

Exportação de Dados

A API oferece funcionalidade de exportação de dados em formato CSV.

Export (protegido)

- GET /export/match/{match_id}/players/csv
  - Exporta informações de todos os jogadores de uma partida específica
  - Retorna arquivo CSV para download
  - Campos incluídos: Nome, Email, Telefone, Documento, Posição, Data de Nascimento, Time, Status (Confirmado/Pendente)
  - Requer autenticação via Bearer token
  - Retorna 404 se a partida não for encontrada
  - Nome do arquivo: `match_{match_id}_players.csv`

Exemplo de uso

Exportar jogadores de uma partida: “‘bash curl -H “Authorization: Bearer ” http://localhost:8000/export/match/1/players/csv –output match_1_players.csv

Padrões e Convenções

- Respostas JSON.
- Datas em ISO 8601 (UTC).
- Paginação padrão: `page` (1) e `page_size` (20, máx 100).
- Erros:
  - 400: erro de estado/validação de domínio
  - 401: não autenticado
  - 403: sem permissão (admin)
  - 404: não encontrado
  - 409: conflito (ex.: já inscrito)
  - 422: erro de validação (FastAPI)