Projeto Final: API Sistema de Locadora de Jogos

Visao Geral
Desenvolver uma API REST simples para gerenciar uma locadora de jogos, com autenticacao basica, controle de acesso e operacoes CRUD.
Usuarios comuns podem alugar/devolver jogos, enquanto administradores podem cadastrar e gerenciar o catalogo.

Objetivos de Aprendizagem
- Implementar rotas seguindo o padrao REST
- Criar middlewares de autenticacao e autorizacao
- Implementar autenticacao Basic Token
- Utilizar Prisma ORM + SQLite
- Diferenciar permissoes entre usuario comum e admin

Estrutura do Banco de Dados

Tabela: users
- id (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- username (TEXT, UNIQUE, NOT NULL)
- password (TEXT, NOT NULL)
- isAdmin (BOOLEAN, DEFAULT false)

Tabela: games
- id (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- title (TEXT, NOT NULL)
- platform (TEXT, NOT NULL)
- available (BOOLEAN, DEFAULT true)

Tabela: rentals
- id (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- userId (INTEGER, FOREIGN KEY -> users.id)
- gameId (INTEGER, FOREIGN KEY -> games.id)
- rentedAt (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- returned (BOOLEAN, DEFAULT false)

Sistema de Autenticacao
Formato do Token: Authorization: Basic <base64(username:password)>

Permissoes
Usuario Comum:
- Ver lista de jogos disponiveis
- Alugar um jogo
- Devolver jogo alugado

- Ver seus proprios alugueis

Administrador:
- Todas as acoes acima
- Cadastrar novo jogo
- Editar ou excluir jogos

Especificacao das Rotas
Rotas Publicas
POST /auth/register
- Cadastrar novo usuario
- Body: { username, password }
- Response: { message, userId }

Rotas de Jogos
GET /games
- Listar todos os jogos
- Auth: Usuario ou Admin
- Response: [{ id, title, platform, available }]

GET /games/:id
- Detalhes de um jogo
- Auth: Usuario ou Admin
- Response: { id, title, platform, available }

POST /games
- Criar novo jogo
- Auth: APENAS Admin
- Body: { title, platform }
- Response: { message, game }

PATCH /games/:id
- Atualizar jogo
- Auth: APENAS Admin
- Body: { title?, platform?, available? }
- Response: { message, game }

DELETE /games/:id
- Deletar jogo
- Auth: APENAS Admin
- Response: { message }

Rotas de Aluguel (rentals)
POST /rentals/:gameId
- Alugar jogo (marca como indisponivel)

- Auth: Usuario ou Admin
- Regras: nao pode alugar se o jogo nao estiver disponivel
- Response: { message }

POST /rentals/:id/return
- Devolver jogo (marca como disponivel)
- Auth: Usuario ou Admin
- Regras: apenas o dono do aluguel ou admin pode devolver
- Response: { message }

GET /rentals/me
- Listar meus alugueis
- Auth: Usuario ou Admin
- Response: [{ id, gameId, rentedAt, returned }]

Middlewares Obrigatorios
auth.js
- Decodifica o Basic Token (username:password)
- Busca o usuario no banco
- Retorna 401 se nao autenticado

admin.js
- Verifica se o usuario autenticado e admin
- Retorna 403 se nao for admin

Regras de Negocio
Usuarios
- Username unico
- Senha com no minimo 4 caracteres
- Primeiro usuario registrado e automaticamente admin

Jogos
- title e platform sao obrigatorios
- Apenas admin pode criar/editar/deletar

Alugueis
- Um jogo so pode ser alugado se estiver available = true
- Ao alugar, o jogo passa a available = false
- Ao devolver, available = true
- Apenas quem alugou ou um admin pode devolver

Dados Iniciais (SQL)
-- Usuarios
INSERT INTO users (username, password, isAdmin) VALUES
('admin', '1234', 1),

('player1', 'abcd', 0);

-- Jogos
INSERT INTO games (title, platform, available) VALUES
('The Legend of Zelda', 'Switch', 1),
('God of War', 'PS5', 1),
('Elden Ring', 'PC', 0),
('Mario Kart 8', 'Switch', 1);

Critérios de Avaliacao
Funcionalidades (50 pts)
- Autenticacao funcionando
- CRUD de jogos (admin)
- Sistema de aluguel/devolucao
- Controle de permissoes

Middlewares (30 pts)
- Autenticacao
- Autorizacao admin
- Tratamento basico de erros

Qualidade do Codigo (20 pts)
- Estrutura organizada
- Rotas REST corretas
- Codigo limpo e funcional

Estrutura do Projeto
projeto-locadora/
- src/
  - middlewares/
    - auth.js
    - admin.js
  - routes/
    - auth.js
    - games.js
    - rentals.js
  - server.js
- prisma/
  - schema.prisma
- package.json
- README.md

Dicas
1. Comece com /games (sem auth)
2. Depois implemente auth.js

3. Adicione /auth/register
4. Depois as rotas de aluguel
5. Por fim, middleware de admin

Fluxo de Teste Exemplo
1. POST /auth/register -> cria usuario novo
2. GET /games -> lista jogos
3. POST /rentals/1 -> aluga um jogo
4. POST /rentals/1/return -> devolve
5. POST /games -> tenta criar jogo (403 se nao admin)
6. POST /games -> cria jogo com token admin