# Documentação do Projeto: Movie Catalog API

## 1. Visão Geral

Este projeto é uma **API RESTful** robusta e moderna, desenvolvida em **Spring Boot**, que simula um sistema de catálogo de filmes. Foi concebida para demonstrar a implementação de uma arquitetura de backend completa, com foco especial na criação e gestão de **múltiplos relacionamentos N×N (Muitos para Muitos)**, um desafio comum em sistemas de gerenciamento de dados complexos.

## 2. Tecnologias Utilizadas

O projeto utiliza o seguinte stack tecnológico, garantindo uma aplicação moderna e segura:

| Tecnologia | Função |
|---|---|
| Spring Boot 3.x | Framework principal para a API. |
| Spring Data JPA | Persistência de dados e mapeamento Objeto-Relacional. |
| Spring Security | Segurança da aplicação e autenticação via JWT. |
| JWT (JSON Web Token) | Mecanismo de autenticação stateless. |
| Flyway | Gerenciamento de database migrations. |
| H2 Database | Banco de dados em memória para desenvolvimento e testes. |
| Lombok | Redução de código boilerplate (getters, setters, builders). |
| SpringDoc OpenAPI | Geração automática de documentação Swagger. |

# 3. Domínio e Relacionamentos

O domínio escolhido, Catálogo de Filmes, foi selecionado por permitir a implementação natural e prática de dois relacionamentos N×N distintos, essenciais para a demonstração da arquitetura de dados.

## 3.1. Diagrama Entidade-Relacionamento (ER)

[O diagrama ER original deve ser inserido aqui. Como não posso gerar imagens, o espaço é reservado. O diagrama ilustra a estrutura do banco de dados, destacando as tabelas de junção utilizadas para modelar os relacionamentos N×N.]

## 3.2. Detalhe dos Relacionamentos N×N

| Entidades Envolvidas | Tabela de Junção | Detalhe |
|---|---|---|
| Filme e Gênero | `FilmeGenero` | Simples junção para associar múltiplos gêneros a um filme e vice-versa. |
| Filme e Ator | `FilmeAtor` | Junção com atributo extra (`papel`), permitindo armazenar o papel do ator naquele filme específico (ex: "Protagonista", "Coadjuvante"). |

# 4. Endpoints e Documentação

A API expõe endpoints para gerenciamento de usuários (autenticação) e operações CRUD (Create, Read, Update, Delete) para Gêneros, Atores e Filmes.

A documentação completa e interativa da API está disponível via Swagger UI no caminho `/swagger-ui.html` após a inicialização do projeto.

### 4.1. Autenticação

| Método | Caminho | Descrição |
|---|---|---|
| POST | `/auth/register` | Registra um novo usuário e retorna o JWT. |
| POST | `/auth/login` | Realiza o login e retorna o JWT. |

### 4.2. Endpoints Protegidos (CRUD)

Todos os endpoints abaixo exigem o JWT no cabeçalho `Authorization: Bearer <token>`.

| Entidade | Caminho | Métodos |
|---|---|---|
| Gêneros | `/api/generos` | `POST`, `GET`, `PUT/{id}`, `GET/{id}`, `DELETE/{id}` |
| Atores | `/api/atores` | `POST`, `GET`, `PUT/{id}`, `GET/{id}`, `DELETE/{id}` |
| Filmes | `/api/filmes` | `POST`, `GET`, `PUT/{id}`, `GET/{id}`, `DELETE/{id}` |

O endpoint de Filmes é o mais complexo, pois o `POST` e o `PUT` aceitam listas de IDs de Gêneros e Atores no DTO para gerenciar os relacionamentos N×N de forma transacional.

## 5. Como Iniciar o Projeto

1. **Requisitos:** Java 21+ e Maven.
2. **Compilação:** Navegue até o diretório `movie-catalog` e execute `mvn clean install`.
3. **Execução:** Execute o JAR gerado com `java -jar target/movie-catalog-0.0.1-SNAPSHOT.jar`.
4. **Acesso ao Swagger:** Abra seu navegador em `http://localhost:8080/swagger-ui.html`.