Sistema de Streaming de Vídeos com Spring Data JPA

Este projeto implementa um sistema básico de streaming de vídeos utilizando Spring Boot e Spring Data JPA. Ele demonstra a modelagem de entidades, a criação de repositórios com query methods personalizados e a inserção de dados de exemplo, além de consultas específicas conforme solicitado.

Tecnologias Utilizadas

- **Spring Boot:** Framework para desenvolvimento rápido de aplicações Java.
- **Spring Data JPA:** Simplifica a implementação de repositórios baseados em JPA.
- **H2 Database:** Banco de dados em memória para desenvolvimento e testes.
- **Maven:** Ferramenta de automação de build.
- **Java 17:** Linguagem de programação.

Estrutura do Projeto

Plain Text

```
streaming-videos
├── src
│   ├── main
│   │   ├── java
│   │   │   └── com
│   │   │       └── example
│   │   │           └── streamingvideos
│   │   │               ├── StreamingVideosApplication.java
│   │   │               ├── controller
│   │   │               │   └── QueryController.java
│   │   │               ├── entity
│   │   │               │   ├── Avaliacao.java
│   │   │               │   ├── Categoria.java
│   │   │               │   ├── Usuario.java
│   │   │               │   ├── Video.java
│   │   │               │   └── Visualizacao.java
│   │   │               ├── repository
│   │   │               │   ├── AvaliacaoRepository.java
│   │   │               │   ├── CategoriaRepository.java
│   │   │               │   ├── UsuarioRepository.java
│   │   │               │   ├── VideoRepository.java
│   │   │               │   └── VisualizacaoRepository.java
```

```
│   │   │               └── service
│   │   │                   ├── DataInitializer.java
│   │   │                   └── QueryService.java
│   │   └── resources
│   │       └── application.properties
│   └── test
│       └── java
│           └── com
│               └── example
│                   └── streamingvideos
│                       └── StreamingVideosApplicationTests.java
├── pom.xml
└── diagram.puml
└── diagram.png
```

Entidades Modeladas

As seguintes entidades foram modeladas para representar o domínio do sistema de streaming:

- **Categoria:** Representa as categorias dos vídeos (e.g., Ação, Comédia).
  - `id` : Identificador único (chave primária).
  - `nome` : Nome da categoria.
- **Usuário:** Representa os usuários do sistema.
  - `id` : Identificador único (chave primária).
  - `nome` : Nome do usuário.
  - `email` : Endereço de e-mail do usuário.
- **Vídeo:** Representa os vídeos disponíveis para streaming.
  - `id` : Identificador único (chave primária).
  - `titulo` : Título do vídeo.
  - `descricao` : Descrição detalhada do vídeo.
  - `url` : URL do vídeo.
  - `dataPublicacao` : Data de publicação do vídeo.
  - `duracao` : Duração do vídeo em minutos.
  - `categoria` : Categoria à qual o vídeo pertence (relacionamento Many-to-One com Categoria).
- **Avaliação:** Representa a avaliação de um usuário para um vídeo.
  - `id` : Identificador único (chave primária).

    - `video`: Vídeo avaliado (relacionamento Many-to-One com Vídeo).
    - `usuario`: Usuário que fez a avaliação (relacionamento Many-to-One com Usuário).
    - `nota`: Nota atribuída ao vídeo (e.g., de 1 a 5).
    - `comentario`: Comentário do usuário sobre o vídeo.
    - `dataAvaliacao`: Data e hora da avaliação.
- **Visualização:** Registra quando um usuário assistiu a um vídeo.
    - `id`: Identificador único (chave primária).
    - `video`: Vídeo assistido (relacionamento Many-to-One com Vídeo).
    - `usuario`: Usuário que assistiu ao vídeo (relacionamento Many-to-One com Usuário).
    - `dataVisualizacao`: Data e hora da visualização.
    - `tempoAssistido`: Tempo assistido do vídeo em minutos.

Como Executar o Projeto

1. **Pré-requisitos:**
    - Java Development Kit (JDK) 17 ou superior.
    - Apache Maven.
2. **Compilar e Executar:**
3. **Acessar o Console H2 (opcional):**
    - **JDBC URL:** `jdbc:h2:mem:streamingdb`
    - **User Name:** `sa`
    - **Password:** (deixe em branco)

Endpoints da API (Consultas)

As consultas solicitadas foram expostas através de uma API RESTful no `QueryController`.

- **Buscar vídeos pelo título com ordenação:**
    - `GET /consultas/videos/titulo/{titulo}`
    - Exemplo: `http://localhost:8080/consultas/videos/titulo/Missão`
- **Todos os vídeos de uma categoria ordenado pelo título:**
    - `GET /consultas/videos/categoria/{nomeCategoria}`
    - Exemplo: `http://localhost:8080/consultas/videos/categoria/Ação`
- **Os top 10 vídeos mais bem avaliados:**

- `GET /consultas/videos/top10-avaliados`
- Exemplo: `http://localhost:8080/consultas/videos/top10-avaliados`

- **Os top 10 vídeos mais assistidos:**
  - `GET /consultas/videos/top10-assistidos`
  - Exemplo: `http://localhost:8080/consultas/videos/top10-assistidos`
- **O usuário que mais assistiu vídeos:**
  - `GET /consultas/usuarios/mais-assistiu`
  - Exemplo: `http://localhost:8080/consultas/usuarios/mais-assistiu`

Diagrama de Classes UML

O diagrama de classes UML reflete a estrutura das entidades e seus relacionamentos no projeto. Ele pode ser encontrado abaixo: