Informações gerais

O projeto acompanha uma collection do postman com o nome "REQUESTS".
O arquivo REQUESTS contém todas as requests possíveis a partir da API Lightsaber, incluindo, também,
um método para interagir com cada endpoint da API Movies

As tabelas do banco de dados MySQL é gerado e atualizado automaticamente a partir da configuração spring.jpa.hibernate.ddl-auto=update,
necessitando, apenas, da execução do comando de create database:

CREATE DATABASE maythefourth;

Os métodos no controller começados por *"get"* se referem à interação com a API Movies, enquanto os métodos começados por *"find"*
se referem à pesquisa no banco de dados.

Os métodos *include()* e *edit()* interagem unicamente com o banco de dados MySQL, sem passar pela API Movies.

Antes de testar, certifique-se de que ambas as aplicações estejam sendo executadas em suas respectivas portas:

API Movies - 7029.
API Lightsaber - 8080

Endpoints

GET
http://localhost:8080/movies/filter/character?characterName=***{nomePersonagem}***
{nomePersonagem}: nome do personagem a ser pesquisado, pode conter espaços.
Pesquisa filmes em que o personagem fornecido aparece. Retorna a lista de filmes.

GET
http://localhost:8080/movies/filter/title?title=***{nomeFilme}***
{nomeTitle}: nome do filme a ser pesquisado, pode conter espaços.
Pesquisa filme pelo título. Retorna os dados do filme especificado e salva no banco de dados, caso o retorno da API não seja nulo e o dado não esteja registrado no banco de dados.

GET
http://localhost:8080/movies/filter/episode?episodeId=***{numeroEpisodio}***
{numeroEpisodio}: episódio do filme a ser pesquisado.
Pesquisa filme pelo episódio. Retorna os dados do filme especificado.

POST
http://localhost:8080/movies
Cadastra um filme no banco de dados.

```
{
   "movieName": "Filminho editado",
   "movieDescription": "Lorem Ipsum"
   “peopleId”: [
 {
    “peopleId”: 1
 },
 {
    “peopleId”: 2
 }]
}
```

movieName: nome do filme cadastrado;
movieDescription: descrição do filme;
peopleId: id do personagem a ser cadastrado na tablea que relaciona filmes e personagens, caso haja personagem para relacionar.

PUT
http://localhost:8080/movies
Atualiza o registro de um filme.

```
{
   "movieId":2,
   "movieName": "Filminho editado",
   "movieDescription": "Lorem Ipsum"
   “peopleId”: [
 {
    “peopleId”: 1
 },
 {
    “peopleId”: 2
 }]
}
```

movieId: id do filme a ser atualizado;
movieName: nome do filme cadastrado;
movieDescription: descrição do filme;
peopleId: id do personagem a ser cadastrado na tablea que relaciona filmes e personagens, caso haja personagem para relacionar.

GET

http://localhost:8080/movies/find/id?id=**{idDado}**
Pesquisa filme pela chave primária fornecida.
{idDado}: chave primária no banco de dados banco de dados.

GET
http://localhost:8080/movies/find/name?name=**{nomeFilme}**
Pesquisa filme no banco de dados por nome.
{nomeFilme}: nome do filme a ser pesquisado. Pode conter espaços.

GET
http://localhost:8080/movies/find/
Busca todos os filmes, não requer fornecimento de dados.

GET
http://localhost:8080/people/filter/name?name=**{nomePersonagem}**
Pesquisa na API Externa o personagem pelo nome passado.
{nomePersonagem}: nome do personagem a ser pesquisado, pode conter espaços.

POST
http://localhost:8080/people/
Cadastra um personagem e, caso fornecidos, os filmes aos quais ele está relacionado

```
{
   "peopleName": "Luke Skywalker"
     “movieId”: [
 {
   “movieId”: 1
 },
 {
   “movieId”: 2
 }]
}
```

peopleName: nome do personagem a ser cadastrado;
movieId: chave primária do filme a ser relacionado ao personagem, caso haja filme.

PUT
http://localhost:8080/people/

```
{
  “peopleId”:1
   "peopleName": "Luke Skywalker"
   “movieId”: [
 {
   “movieId”: 1
 },
```

```
  {
    “movieId”: 2
  }]
},
```

peopleId: chave primária do dado a ser atualizado no banco de dados
peopleName: nome do personagem a ser cadastrado;
movieId: chave primária do filme a ser relacionado ao personagem, caso haja filme.

GET
http://localhost:8080/find/name?name=**{nomePersonagem}**
Pesquisa personagem por nome no banco de dados.
{nomePersonagem}: nome do personagem a ser pesquisado no banco de dados;

GET
http://localhost:8080/find/id?peopleId=**{idPersonagem}**
Pesquisa um personagem no banco de dados pela chave primária
{idPersonagem}: chave primária do personagem a ser pesquisada no banco;