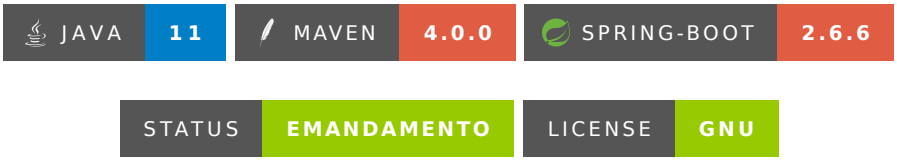
MoviesBattle - Desafio Lets's Code



Tópicos

Descrição do projeto

API REST para uma aplicação ao estilo card game, onde serão informados dois filmes e o jogador deve acertar aquele que possui melhor avaliação no IMDB.

Requisitos

Funcionais

✔️ O jogador deve fazer login para iniciar uma nova partida,

✔️ Dois usuários/jogadores já configurados,

❌ Cada rodada do jogo consiste em informar um par de filmes,

❌ O jogador deve tentar acertar qual filme possui maior pontuação,

❌ Ranking.

Não Funcionais

✔️ Armazene os dados em H2,

✔️ Aplicação iniciada usando webscraping,

❌ Testes unitários,

❌ Documentação da API com base no OpenAPI 3.0,

✔️ Solução de autenticação - JWT.

Pré-requisitos

⚠️ JDK 11

⚠️ Apache Maven

⚠️ Postman

Dependências

🔧 H2 Database

🔧 Spring Data JPA

🔧 Spring Web

🔧 Validation

🔧 Lombok

🔧 Security

🔧 JWT

🔧 Validation

🔧 jsoup

Como rodar a aplicação ▶️

clone o projeto

```
git clone https://github.com/nauam/MoviesBattle.git
cd MoviesBattle/api-movies-battle/
```

Construção

Para construir o projeto com o Maven, executar os comando abaixo:

```
mvn clean install
```

O comando irá baixar todas as dependências do projeto e criar um diretório target com os artefatos construídos, que incluem o arquivo jar do projeto. Além disso, serão executados os testes unitários, e se algum falhar, o Maven exibirá essa informação no console.

Execução

Para executar o projeto com o Maven Spring Boot Plugin, executar os comando abaixo:

```
mvn spring-boot:run
```

O comando irá rodar o projeto e subir na porta 8080

Cadastro:

```
localhost:8080/signup
```

▼ **Payload (Clique aqui)**
Cadastrar uma nova conta: [**POST**]

```
{
        "username" : "Maria de Nazaré Esteves Tedesco",
        "email" : "nazare.tedesco@outlook.com",
        "password": "10131949004"
}
```

Caso ocorra como esperado:

```
{
    "message": "Usuário cadastrado com sucesso!"
}
```

Caso ocorra algum erro de validação irá aparecer:

```
{
    "status": 400,
    "msg": "Erro de validação",
    "timeStamp": 1649641200705,
    "errors": [
        {
            "fieldName": "username",
            "message": "Username já existente"
        }
                .
```

```
                .
                .
    ]
}
```

Validação dos campos:

👦 username - tamanho entre 3 a 120 caracteres e não pode está em branco

🔑 password - tamanho entre 8 a 120 caracteres e não pode está em branco

📧 email será validado.

Login:

```
localhost:8080/signin
```

▼ **Payload (Clique aqui)**
Logar na conta: [**POST**]

```
{
        "username" : "Maria de Nazaré Esteves Tedesco",
        "password": "10131949004"
}
```

Caso ocorra como esperado:

```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJNYXJpYSBkZSBOYXphcsC
    "id": 4,
    "username": "Maria de Nazaré Esteves Tedesco",
    "email": "nazare.tedesco@outlook.com"
}
```

Caso ocorra algum erro de validação irá aparecer:

```
{
    "status": 400,
    "msg": "Erro de validação",
    "timeStamp": 1649641772269,
    "errors": [
        {
            "fieldName": "username",
            "message": "Username não existente"
        }
    ]
}
```

Listar filmes:

```
localhost:8080/imdb/list
```

▼ **Payload (Clique aqui)**
Ver os 250 melhores filmes rankeados do IMDB: [**GET**]

```
        KEY: Authorization,
        VALUE: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJNYXJpYSBkZSBOYXphcsOpIEVzdGV2ZXMg\
8HauuyyG1O99qOFXgfMCAmZeLw
```

Caso ocorra como esperado:

```
[
    {
        "id": 1,
        "ranking": "1",
        "rating": "9.2",
        "name": "Um Sonho de Liberdade (1994)"
    },
    {
        "id": 2,
        "ranking": "2",
        "rating": "9.2",
        "name": "O Poderoso Chefão (1972)"
    },
        .
        .
        .
]
```

Como concluir o projeto:

O projeto conta com a funcionalidade de encriptar a senha dos usuário para salvar no banco de dados, isso permite que haja uma segurança maior.

A funcionalidade de autenticação foi feita usando o JWT (JSON Web Token) que é um método RCT 7519 padrão da indústria para realizar autenticação entre duas partes por meio de um token assinado que autentica uma requisição web. Esse token é um código em Base64 que armazena objetos JSON com os dados que permitem a autenticação da requisição. Para acessar qualquer endpoint é necessário está autenticado, exceto para os endpoints de cadastro `/signup` e o de login `/signin` .

Cada requisição feita, há um padrão de validação do payload que deve ser seguida, por exemplo, para o signup o campo username deve conter entre 3 a 120 caracteres, não pode está em branco e tem que ser único, o campo de password deve conter entre 8 a 120 caracteres e não pode está em branco e por fim o campo email é validado e deve conter as três partes principais: a parte local, um símbolo @ e um domínio. Caso haja algum campo invalido, aparecerá uma mensagem de explicando o erro de validação.

Quando o projeto é rodado, dois jogadores são criados o Player1 `{"username": Player1, "password": luvadepedreiro}` e o Player2 `{"username": Player2, "password": jaacaboujessica}` .

A base de dados é formada pelos 250 melhores filmes rankeados do IMDB e é baseada do site oficial do IMDb, onde é usado o WebScrapping para respar os dados da pagina e salvar no BD (banco de dados). As informações que são pegas no site são referente ao ranking, nota, nome e ano de lançamento. Para gerar esse banco de dados é necessário está autenticado e acessar o endpoint `/imdb/list` , essa funcionalidade ainda tem que ser melhorada para gerar o BD ao rodar o projeto.

Devido ao tempo para codificar a aplicação, as regras de negócio do jogo não foram implementadas, mas essas regras seriam:

Com o usuário devidamente autenticado, o jogo seria inicializado pelo endpoint `game/quiz/start` com @GetMapping e teria como resposta o numero da partida, posição em tela do filme 1, nome do filme 1, posição em tela do filme 2, nome do filme 2, numero de acertos, o numero de erros, ranking e se a partida está finalizada a resposta seria assim:

```
{
    "quiz": 1,
    "choice": [{
        "position": "LEFT",
        "name": "O Poderoso Chefão (1972)"
    },{
        "position": "RIGHT",
        "name": "Um Sonho de Liberdade (1994)"
    }],
    "score": 0,
    "failure": 0,
    "ranking": 5,
    "finish": false
}
```

Para validar a resposta o endpoint seria `game/answer/{$quiz}?choice={$position}` com @PutMapping e teria como resposta o numero da nova partida, se a resposta estava certo ou errada, numero de acertos, o numero de erros, ranking e se a partida está finalizada a resposta seria assim:

```
{
    "quiz": 1,
    "answer": "correct",
    "score": 1,
    "failure": 0,
    "ranking": 3,
    "finish": false
}
```

Caso erre 3 vezes a resposta seria assim:

```
{
    "quiz": 3,
    "answer": "wrong",
    "score": 0,
    "failure": 3,
    "ranking": 5,
    "finish": true
}
```

Para continuar o jogo o endpoint seria `game/quiz/next` com @GetMapping e teria como resposta igual ao endpoint `game/quiz/start` :

```
{
    "quiz": 2,
    "choice": [{
        "position": "LEFT",
        "name": "O Poderoso Chefão (1972)"
    },{
        "position": "RIGHT",
        "name": "Batman: O Cavaleiro das Trevas"
    }],
    "score": 1,
    "failure": 0,
    "ranking": 3,
    "finish": false
}
```

O jogo pode ser finalizado pelo usuário a partir do endpoint `game/quiz/finish` com @GetMapping e teria como resposta o numero de acertos, o numero de erros, ranking e se a partida está finalizada, a resposta seria assim:

```
{
    "score": 1,
    "failure": 0,
    "ranking": 3,
    "finish": true
}
```

A listagem do ranking pode ser visualizado pelo usuário a partir do endpoint `game/ranking` com @GetMapping e teria como resposta o numero de acertos, o numero de erros, ranking e se a partida está finalizada, a resposta seria assim:

```
{
    "1": {
        "username": "player2"
    },
    "2": {
        "username": "player1"
    },
```

```
    "3": {
        "username": "player1"
    },
    .
    .
    .
    "5": {
        "username": "player10"
    }
}
```

Devido ao tempo, não foi implementado os testes unitários na aplicação, o teste unitário consiste em verificar o comportamento das menores unidades na aplicação.

Conclusão

Essa aplicação me fez/fará colocar bastante teória em prátrica e também colocar bastante coisas que eu vejo no meu dia a dia como desenvolvedor em um projeto totalmente novo.

Licença