Maratona Tech

*Tecnologias da Informação*

*Backend – P2*

Uma imagem contendo Gráfico de pizza

Descrição gerada automaticamente

Um grupo de internautas cinéfilos se uniu para discutir sobre seu hobbie. Eles chegaram à conclusão de que falta um espaço para que eles divulguem as opiniões sobre os filmes que assistem e que estimule a interação entre eles.

Seu trabalho é desenvolver a API de uma plataforma que seja capaz de listar os diferentes filmes e seus artistas, além de armazenar avaliações de seus usuários.

Você terá o tempo de até 4 horas para concluir as seguintes tarefas:

AUTENTICAÇÃO

Regras de autorização e autenticação para todos os endpoints, exceto os que não necessitam de token

*Se o token (auth header) não for fornecido deve retornar:*

Status Code: 401 (Unauthorized) Response Body:

{

"message": "Unauthenticated user"

}

*Se o token (auth header) foi fornecido, mas é inválido, retorna...*

Status Code: 403 (Forbidden) Response Body:

{

"message": "Invalid token"

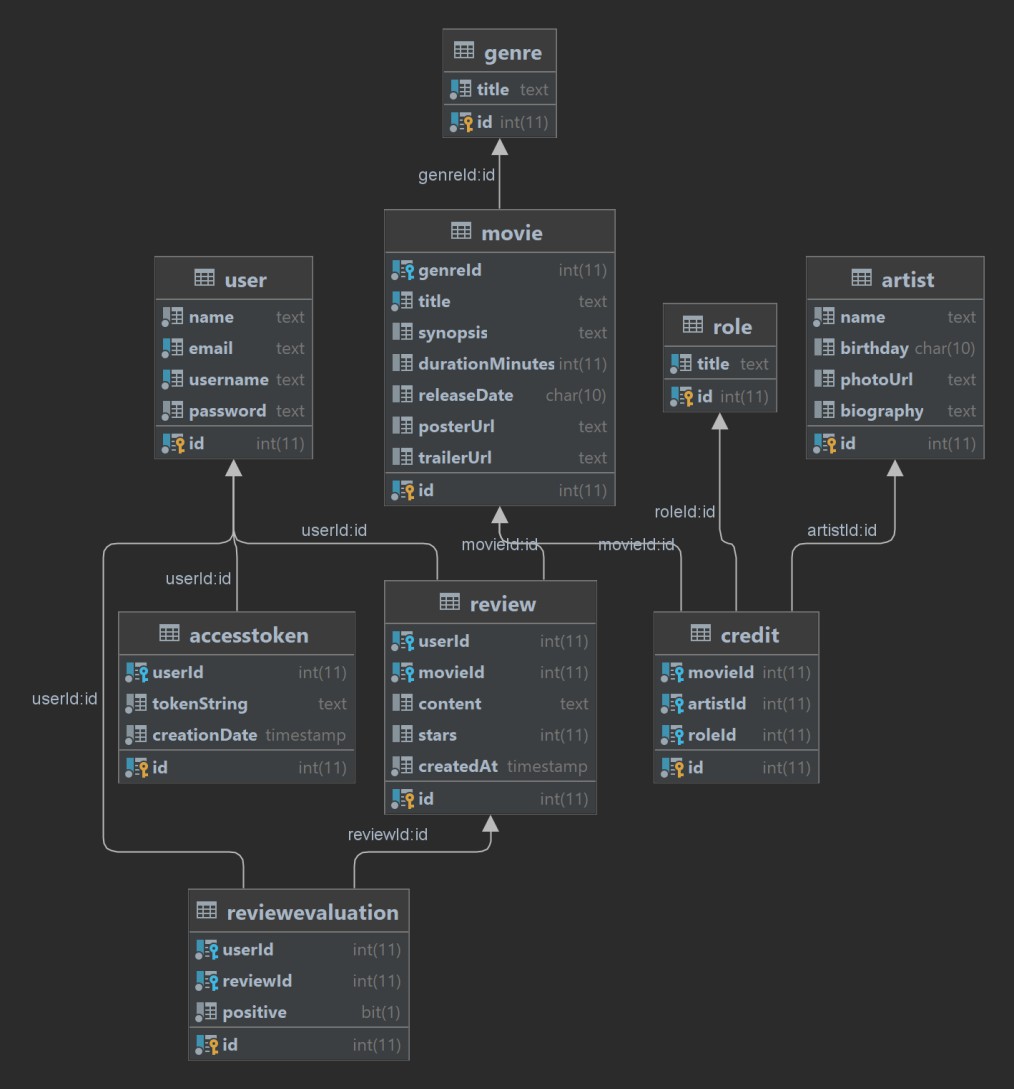
}

SEGURANÇA

* A senha deve ser criptografada no momento da criação do usuário. Caso o usuário já esteja salvo no banco com uma senha não criptografada, ela deve ser criptografada no próximo login deste usuário
* O token fornecido pelos endpoints “login” e ‘signup” é responsável por controlar o acesso de todos os endpoints exceto aqueles que não necessitam de token para serem acessados
* Todos os tokens devem expirar depois de uma hora. Depois disso, usar o token expirado irá resultar na sua remoção do banco de dados e o mesmo retorno de quando se usa um token inválido.

BANCO DE DADOS (ENTIDADES E MODELS)

O banco de dados deve ser criado, com base no diagram abaixo:

Os dados do banco de dados não serão fornecidos e nem populados. Você precisa criar dados fictícios para realizar os testes.

É necessário que as listagens feitas pelos endpoints sejam fiéis aos objetos JSON fornecidos na prova, seguindo a mesma ordem e apresentando as mesmas propriedades de acordo com as especificações definidas nos endpoints.

Os endpoints estão definidos a seguir…

**POST** /api/v1/auth/signup

Cria um usuário e retorna um token. Não é necessário um token para acessar este endpoint

|  |  |
| --- | --- |
| **Campo** | **Descrição** |
| **email** | Email do usuário. Necessário e único. |
| **password** | Senha do usuário. Deve ter um mínimo de 6 caracteres. Necessária |
| **name** | Nome do usuário. Necessário. |
| **username** | Nome de usuário do usuário. Necessário e único. |

*Se qualquer campo estiver inválido (de acordo com a descrição), retornar...*

Status Code: 422 (Unprocessable Entity) Response Body:

{

"message": "Invalid properties", "errors": {

propriedade: [ erro,

...

],

propriedade: [ erro,

...

],

...

}

}

*Se a criação do usuário for bem-sucedida deve retornar:*

Status Code: 201 (Created) Response Body:

{

"token": tokenString

}

**DELETE** /api/v1/auth/signout

Deleta o token do usuário logado

*Se o token for deletado com sucesso deve retornar:*

Status Code: 204 (No Content)

**POST** /api/v1/login

Valida as credenciais do usuário e retorna um token. Não é necessário um token para acessar este endpoint

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Campo** | **Descrição** |  |
|  | **email** | Email do usuário. Necessário |  |
|  | **password** | Senha do usuário. Deve ter mais de 6 caractéres. Necessária |  |

*Se qualquer campo estiver inválido (de acordo com a descrição), retornar...*

Status Code: 422 (Unprocessable Entity) Response Body:

{

"message": "Invalid properties", "errors": {

propriedade: [ erro,

...

],

propriedade: [ erro,

...

],

...

}

}

*Se o email ou a senha não forem compatíveis deve retornar:*

Status Code: 422 (Unprocessable entity)

Response Body:

{

"message": "Invalid email or password"

}

*Se o email e a senha forem compatíveis deve retornar:*

Status Code: 200 (Success) Response Body:

{

"token": tokenString

}

**GET** /api/v1/movies

Lista os filmes de acordo com os filtros fornecidos nos parâmetros.

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Descrição** | **Padrão** |

|  |  |  |
| --- | --- | --- |
| **page** | Número da página | 1 |
| **pageSize** | Tamanho das páginas | 4 |
| **sortDir** | Direção da ordenação, pode ser “asc” (crescente) ou “desc” (decrescente) | desc |
| **sortBy** | Propriedade da ordenação, deve ser “title”, “releaseDate” | releaseDate |

Propriedades de ordenação descritas abaixo

|  |  |
| --- | --- |
| **Propriedade** | **Descrição** |
| **title** | Título do filme |
| **releaseDate** | Data de lançamento do filme |

*Se a requisição foi bem-sucedida deve retornar:*

Status Code: 200 (Success) Response Body:

[

{

"id": id do filme (integer), "title": título do filme (string),

"duration": duração do filme formatada como: (horas)h(minutos)m

(string),

"releaseDate": data de lançamento do filme(string), "posterUrl": link para o endpoint do poster do filme (string),

"singlePageUrl": link para o endpoint de listagem individual do filme (string)

},

…

]

**GET** /api/v1/movies/<id>

Lista um único filme de acordo com seu id

|  |  |
| --- | --- |
| **Parâmetro** | **Descrição** |
| **id** | Id do filme. Necessário |

*Se um id de filme inválido/inexistente for fornecido, retornar...*

Status Code: 404 (Not Found) Response Body:

{

"message": "Invalid movie id",

}

*Se a requisição for bem-sucedida, retornar...*

Status Code: 200 (Success) Response Body:

{

"title": título do filme (string), "synopsis": sinopse do filme(string),

"duration": duração do filme formatada como: (horas)h(minutos)m (string), "releaseDate": data de lançamento do filme(string),

"posterUrl": link para o endpoint do poster do filme (string), "trailerUrl": link para o endpoint do trailer do filme (string), "credits": [

{

"artistId": id do artista (integer), "name": nome do artista (string),

"role": título da função relacionada ao crédito (string), "photoUrl": link para o endpoint do poster do artista (string), "singlePageUrl": link para o endpoint de listagem individual do

artista (string)

},

…

]

}

**GET** /api/v1/artists

Lista os artistas de acordo com os filtros fornecidos nos parâmetros

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Descrição** | **Padrão** |
| **page** | Número da página | 1 |
| **pageSize** | Tamanho das páginas | 10 |
| **sortDir** | Direção da ordenação. Pode ser “asc” (crescente) ou “desc” (decrescente) | asc |

Os artistas devem utilizar o nome como parâmetro para ordenação

*Se a requisição for bem-sucedida deve retornar:*

Status Code: 200 (Success) Response Body:

[

{

"id": id do artista (integer), "name": nome do artista (string),

"photoUrl": link para o endpoint do poster do artista (string), "singlePageUrl": link para o endpoint de listagem individual do

artista (string)

},

…

]

**GET** /api/v1/artists/<id>

Lista um único filme de acordo com seu id

|  |  |
| --- | --- |
| **Parâmetro** | **Descrição** |
| **id** | Id do artista. Necessário |

*Se o id for inválido deve retornar:* Status Code: 404 (Not Found) Response Body:

{

"message": "Invalid artist id"

}

*Se a requisição for bem-sucedida deve retornar:*

Status Code: 200 (Success) Response Body:

{

"name": nome do artista (string),

"birthday": data de nascimento do artista(string), "biography": biografia do artista (string),

"photoUrl": link para o endpoint do poster do artista (string), "movies": [

{

(string),

"id": id do filme (integer), "title": típtulo do filme (string),

"duration": duração do filme formatada como: (horas)h(minutos)m

"releaseDate": data de lançamento do filme (string), "posterUrl": link para o endpoint do poster do filme (string), "singlePageUrl": link para o endpoint de listagem individual do

filme (string)

},

…

]

}

**POST** /api/v1/reviews/<movieId>

Cria uma resenha relacionada ao id do filme fornecido e ao usuário logado. Se já existe uma resenha relacionada a estes, atualize a resenha já existente.

|  |  |
| --- | --- |
| **Parâmetro** | **Descrição** |
| **movieId** | O id do filme associado a resenha.  Necessário |

|  |  |
| --- | --- |
| **Campo** | **Descrição** |
| **stars** | Número de estrelas da resenha. Deve ser um valor entre 1 e 10. Necessário |
| **content** | Texto de conteúdo da resenha. |

*Se qualquer campo for inválido (de acordo com a descrição), retorna...*

Status Code: 422 (Unprocessable Entity) Response Body:

{

"message": "Invalid properties", "errors": {

propriedade: [ erro,

...

],

propriedade: [ erro,

...

],

...

}

}

*Se o id do filme é inválido, retorna …*

Status Code: 400 (Bad Request) Response Body:

{

"message": "Invalid movie id"

}

*Se uma resenha associada ao filme e ao usuário (logado) já existe E a requisição é válida, retorna…*

Status Code: 200 (Success) Response Body:

{

"message": "Review has been successfully updated"

}

*Se uma resenha associada ao filme e ao usuário (logado) não existe E a requisição é válida, retorna …*

Status Code: 201 (Created) Response Body:

{

"message": "Review has been successfully created"

}

**GET** /api/v1/reviews/<movieId>

Lista as resenhas relacionadas a um filme de acordo com os filtros fornecidos nos parâmetros...

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Descrição** | **Padrão** |
| **movieId** | Id do filme relacionado as resenhas. Necessário |  |
| **page** | Número da página | 1 |
| **pageSize** | Tamanho das páginas | 10 |

|  |  |  |
| --- | --- | --- |
| **sortDir** | Direção da ordenação. Pode ser “asc” (crescente) ou “desc” (decrescente) | desc |
| **sortBy** | Propriedade da ordenação.  Deve ser “stars” ou “createdAt” | stars |

Propriedades de ordenação descritas abaixo

|  |  |
| --- | --- |
| **Propriedade** | **Descrição** |
| **stars** | Número de estrelas da resenha |
| **createdAt** | Data de criação da resenha |

*Se o id do filme for inválido, retorna...*

Status Code: 400 (Bad Request) Response Body:

{

"message": "Invalid movie id"

}

*Se a requisição for bem-sucedida, retorna...*

Status Code: 200 (Success) Response Body:

[

{

"id": id da resenha (integer),

"username": nome de usuário do publicador da resenha (string), "content": conteúdo da resenha(string),

"stars": número de estrelas da resenha (integer), "createdAt": data de criação da resenha (string),

"myEvaluation": **avaliação do usuário logado para essa resenha. Pode**

**ser positiva (true), negativa (false) ou nenhuma das duas (null)**

},

…

]

**DELETE** /api/v1/reviews/<movieId>

Deleta a resenha relacionada ao filme definido pelo id e ao usuário logado.

|  |  |
| --- | --- |
| **Parâmetro** | **Descrição** |
| **movieId** | Id associado a resenha. Necessário |

*Se o id do filme for inválido, retorna...*

Status Code: 404 (Not Found) Response Body:

{

"message": "Invalid movie id"

}

*Se o usuário logado não tem uma resenha associada ao filme, retorna...*

Status Code: 404 (Not Found) Response Body:

{

"message": "You haven’t published a review to this movie"

}

*Se a resenha for deletada com sucesso, retorna...*

Status Code: 204 (No Content)

**GET** /api/v1/genres

Lista os gêneros de filme com base nos filtros fornecidos nos parâmetros

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Descrição** | **Padrão** |
| **page** | Número da página | 1 |
| **pageSize** | Tamanho das páginas | 10 |
| **sortDir** | Direção de ordenação, deve ser “asc” (crescente) ou “desc” (decrescente) | desc |

Todos os gêneros devem ser ordenados pelo título

*Se a requisição for bem-sucedida, retorna...*

Status Code: 200 (Success) Response Body:

[

{

},

...

]

"id": id do gênero (integer), "title": titulo do gênero (string)

**Instruções ao competidor**

Ao final do processo deve ser enviado um

* arquivo “README.txt” indicando como rodar a aplicação
* Código fonte com o backend
* Script sql para importar dados e tabelas do banco de dados
* Os arquivos devem ser salvos dentro da pasta “Backend – P1”