Documentação da API

Índice

Introdução

Este é o guia oficial da API do nosso aplicativo. Aqui você encontrará todas as informações necessárias para utilizar nossa API e desenvolver sua própria integração.

Para usar a API, é necessário ter uma chave de acesso, que pode ser obtida

através de um cadastro em no site OMDB.

Autenticação

Descrição

O sistema de autenticação utiliza tokens JWT (JSON Web Tokens) para autenticar usuários em uma aplicação web. A funcionalidade é composta por duas partes: a função signup, responsável por criar um novo usuário na base de dados, criptografar a senha e gerar um token JWT para autenticar o usuário na aplicação; e a função AuthMiddleware, que é um middleware para proteger as rotas que requerem autenticação.

A função AuthMiddleware é responsável por verificar se o token JWT presente na requisição é válido e corresponde a um usuário existente na base de dados. Primeiramente, a função extrai o token da requisição HTTP, verificando se ele está no formato esperado e não está expirado. Em seguida, a função decodifica o token, obtendo o ID do usuário. Com o ID do usuário, a função faz uma consulta na base de dados, buscando o registro correspondente. Se o registro for encontrado, o middleware adiciona o usuário à requisição HTTP, e a função next() é chamada, permitindo que a requisição prossiga para a rota protegida. Caso contrário, a função retorna uma resposta HTTP com status 401 (Unauthorized) ou 500 (Internal Server Error), informando que o token é inválido ou houve um erro interno no servidor.

Middleware /auth

Este middleware é responsável por autenticar o usuário com base no token JWT fornecido no cabeçalho da requisição.

Exemplo de uso: `router.post('/review', userController.AuthMiddleware, reviewController.createReview);`

Resposta Sucesso O middleware não retorna uma resposta direta. Em vez disso, se a autenticação for bem-sucedida, ele adiciona o usuário autenticado ao objeto de requisição (req.user) e chama a próxima função no pipeline do Express.js.

Erros

Código: 401 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Unauthorized: o token JWT não é válido ou o usuário não existe. - Invalid token: o token JWT fornecido não é válido.

POST /user/signup

Essa função é responsável por criar um novo usuário no banco de dados, a partir dos dados fornecidos pelo corpo da requisição (req.body). Além disso, a função também verifica se o email informado já foi cadastrado anteriormente e, em caso afirmativo, retorna uma mensagem de erro.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

- username (obrigatório): nome de usuário do novo usuário.
- email (obrigatório): email do novo usuário.
- password (obrigatório): senha do novo usuário.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - user: objeto JSON contendo os dados do novo usuário cadastrado no banco de dados, incluindo o id gerado pelo banco de dados. - token: token de autenticação do novo usuário, gerado pela biblioteca jsonwebtoken.

Exemplo de resposta:


```
{
    "user": {
        "id": 9,
        "username": "test",
        "email": "test33@test.com"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6OSwidXNlcm5hbWUiOiJ0ZXN0IiwiaWF0I
}
```


Erros

Código: 400 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Email is required: email não foi fornecido no corpo da requisição. - Username is required: nome de usuário não foi fornecido no corpo da requisição. - Password is required: senha não foi fornecida no corpo da requisição. - Email already taken: o email fornecido já foi cadastrado anteriormente. Código: 500 - Conteúdo: objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

POST /user/profile

Essa função é responsável por criar um novo perfil de usuário no banco de dados, a partir dos dados fornecidos pelo corpo da requisição (req.body).

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

name (obrigatório): nome do usuário. familyName (obrigatório): sobrenome do usuário. bio (obrigatório): biografia do usuário.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: mensagem indicando que o perfil foi criado com sucesso. - body: objeto contendo o perfil do usuário que foi criado.

Exemplo de resposta:


```
{
    "message": "Perfil criado com sucesso",
    "body": {
        "profile": {
            "id": 3,
            "userid": 14,
            "name": "Ian",
            "familyname": "Oliveira",
            "icon": null,
            "bio": "Teste"
        }
    }
}
```


Erros

Código: 400 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Um erro aconteceu enquanto o perfil de usuario era criado: ocorreu um erro ao tentar criar o perfil do usuário no banco de dados. Isso pode ser causado por uma variedade de razões, como um erro de conexão com o banco de dados ou um erro na consulta SQL. Código: 500 - Conteúdo: objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

GET /user/profile

Essa função é responsável por obter o perfil do usuário atualmente autenticado.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

Nenhum parâmetro é necessário para essa rota.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: mensagem indicando que o perfil foi atualizado com sucesso. - body: objeto contendo o perfil do usuário que foi atualizado.

Exemplo de resposta:


```
{
    "message": "Perfil atualizado com sucesso",
    "profile": {
        "id": 3,
        "userid": 14,
        "name": "Ian",
        "familyname": "Silva",
        "icon": null,
        "bio": "Teste 3"
    }
}
```


Erros

Código: 500 - Conteúdo: objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

PUT /user/profile

Essa função é responsável por atualizar o perfil do usuário atualmente autenticado.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

- name (obrigatório): nome do usuário.
- familyName (obrigatório): sobrenome do usuário.
- bio (obrigatório): biografia do usuário.

Resposta Sucesso - Código: 200 - Conteúdo: objeto JSON com os seguintes campos: - message: mensagem indicando que o perfil foi encontrado com sucesso. - body: objeto contendo o perfil do usuário.

Exemplo de resposta:


```
{
    "message": "Perfil encontrado com sucesso!",
    "body": {
        "profile": {
            "id": 3,
```

```
            "userid": 14,
            "name": "Ian",
            "familyname": "Oliveira",
            "icon": null,
            "bio": "Teste"
        }
    }
}
```


Erros

Código: 500 - Conteúdo: objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

PATCH /user/profile

Essa função é responsável por atualizar parcialmente o perfil do usuário atualmente autenticado.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

- name (opcional): nome do usuário.
- familyName (opcional): sobrenome do usuário.
- bio (opcional): biografia do usuário.

Resposta Sucesso - Código: 200 - Conteúdo: objeto JSON com os seguintes campos: - message: mensagem indicando que o perfil foi encontrado com sucesso. - body: objeto contendo o perfil do usuário.

Exemplo de resposta:


```
{
    "message": "Perfil atualizado com sucesso!",
    "review": {
        "id": 3,
        "userid": 14,
        "name": "Ian",
        "familyname": "Silva",
        "icon": null,
        "bio": "teste 4"
    }
}
```

Erros

Código: 500 - Conteúdo: objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

Reviews

Descrição

O CRUD implementado é um sistema básico que permite que os usuários gerenciem avaliações de filmes. As funções incluem criar uma nova avaliação, selecionar todas as avaliações existentes, selecionar uma avaliação específica por seu ID, selecionar avaliações específicas por filme, selecionar avaliação especificas por usuario, apagar uma avaliação, atualizar uma avaliação por completo e atualizar parcialmente uma avaliação.

Endpoints

POST /api/review

Cria uma nova avaliação no banco de dados.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

- title (obrigatório): O titulo do filme.
- rating (obrigatório): A classificação do filme em uma escala de 0 a 5.
- comment (obrigatório): O comentário sobre o filme.
- ispublic (opcional): Um valor booleano que indica se a avaliação deve ser pública. O padrão é false.
- specialRating (opcional): A classificação do filme de acordo com o seu genero.

Exemplo de uso:

```
{
  "title": "La La Land",
  "rating": 3,
  "comment": "Um bom filme!",
  "isPublic": true,
  "specialRating": 4
}
```

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Review criada com sucesso! - review: objeto json contendo todos dados da nova review.

Exemplo de resposta:


```
{
    "message": "Review criada com sucesso!",
    "body": {
        "review": {
            "id": 702,
            "userid": 15,
            "movieid": 4,
            "rating": 3,
            "review": "Um bom filme!",
            "ispublic": true,
            "created_at": "2023-09-18T15:01:48.278Z",
            "specialrating": 4
        },
        "title": "La La Land",
        "Nivel de Diversão": 4
    }
}
```


Erros

Código: 400 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Title is required: titulo não foi fornecido no corpo da requisição. - Rating is required: nota não foi fornecido no corpo da requisição. - Review is required: avaliação não foi fornecida no corpo da requisição. - O campo 'rating' deve ser um número entre 0 e 5: usuario digitou um rating invalido. Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

GET /api/allReviews

Essa função é responsável por retornar todas as avaliações do usuario autenticado existentes no banco de dados.

Parâmetros Nenhum parâmetro é necessário para essa rota.

Resposta Sucesso - Código: 200 - Conteúdo: objeto JSON com os seguintes campos: - reviews: array contendo todos objetos JSON com as informações das avaliações. - message: Review(s) encontrada(s) com sucesso!

Exemplo de resposta:


```
{
    "reviews": [
```

```
    {
        "id": 702,
        "userid": 15,
        "movieid": 4,
        "title": "La La Land",
        "rating": 3,
        "review": "Um bom filme!",
        "ispublic": true,
        "created_at": "2023-09-18T15:01:48.278Z"
    },
    {
        "id": 704,
        "userid": 15,
        "movieid": 21,
        "title": "The Nun",
        "rating": 2,
        "review": "muito fraco!",
        "ispublic": true,
        "created_at": "2023-09-18T15:05:43.901Z"
    }
    ],
    "message": "Review(s) encontrada(s) com sucesso!"
}
```


Erros

Código: 400 - O usuario não possui reviews

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

GET /api/allReviews/user

Essa função é responsável por retornar todas as avaliações de um determinado usuario.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:user: nome do usuario que deseja pesquisar as avaliações.

Resposta Sucesso - Código: 200 - Conteúdo: objeto JSON com os seguintes campos: - reviews: array contendo todos objetos JSON com as informações das avaliações. - message: Review(s) encontrada(s) com sucesso!

Exemplo de resposta:

```
[
    {
        "username": "testandoweb2",
        "title": "The Nun",
        "rating": 1,
        "id": 658,
        "specialrating": null,
        "review": "1",
        "created_at": "2023-09-13T00:51:09.666Z"
    },
    {
        "username": "testandoweb2",
        "title": "The Nun",
        "rating": 1,
        "id": 659,
        "specialrating": 1,
        "review": "Obra Prima!",
        "created_at": "2023-09-13T00:51:17.991Z"
    }
]
```


Erros

Código: 400 - O usuario não possui reviews

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

GET /api/allReviews/movies

Essa função é responsável por retornar todas as avaliações de um determinado filme.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:title: nome do filme que deseja pesquisar as avaliações.

Resposta Sucesso - Código: 200 - Conteúdo: objeto JSON com os seguintes campos: - reviews: array contendo todos objetos JSON com as informações das avaliações. - message: Review(s) encontrada(s) com sucesso!

Exemplo de resposta:


```
[
    {
        "username": "testandoweb4",
```

```
        "title": "Pearl",
        "rating": 5,
        "specialrating": 4,
        "review": "Assustador!",
        "created_at": "2023-09-19T14:26:26.814Z"
    },
    {
        "username": "testandoweb3",
        "title": "Pearl",
        "rating": 5,
        "specialrating": 5,
        "review": "Horripilante!",
        "created_at": "2023-09-19T14:26:48.369Z"
    }
]
```


Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

GET /api/review/:id

Essa função é responsável por retornar uma avaliação específica do banco de dados, com base no id fornecido.

Parâmetros

O seguinte parâmetro deve ser enviado na URL da requisição:

:id (obrigatório): id da avaliação que se deseja buscar.

Resposta Sucesso

Código: 200 Conteúdo: objeto JSON com os seguintes campos: review: objeto json contendo todas informações da avaliação. message: Review encontrada com sucesso! Exemplo de resposta:


```
{
    "message": "Review encontrada com sucesso!",
    "body": {
        "review": {
            "username": "testandoWeb",
            "title": "Parasite",
            "rating": 5,
            "specialrating": 5,
```

```
            "review": "5",
            "created_at": "2023-09-09T02:52:49.535Z"
        }
    }
}
```

Erros

Código: 400 - objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Id is required: o id não foi fornecido na URL da requisição. - Invalid id: o id fornecido na URL da requisição não é válido.

Código: 404 - objeto JSON com a mensagem de erro Review not found. Isso ocorre quando a avaliação com o id fornecido na URL não foi encontrada no banco de dados.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

DELETE /api/review/:id

Essa função é responsável por deletar uma avaliação específica do banco de dados.

Parâmetros

:id (obrigatório) - Identificador único da avaliação a ser deletada.

Resposta

Sucesso

Código: 200 Conteúdo: objeto JSON com a mensagem de sucesso. Exemplo de resposta:

```
{
    "message": "Review deletada com sucesso!",
    "review": {
        "id": 702,
        "userid": 15,
        "movieid": 4,
        "rating": 3,
        "review": "Um bom filme!",
        "ispublic": true,
        "created_at": "2023-09-18T15:01:48.278Z",
        "specialrating": 4
    }
}
```

Erros

Código: 404 - objeto JSON com a mensagem de erro Review não encontrada. Isso pode ocorrer caso o identificador único informado não corresponda a uma avaliação existente no banco de dados.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

PUT /api/review/:id

Essa função é responsável por atualizar completamente uma avaliação específica no banco de dados.

Parâmetros

:id (obrigatório) - Identificador único da avaliação a ser atualizada.

Corpo da requisição: objeto JSON contendo os seguintes campos:

- userid (obrigatório) - Identificador único do usuário que realizou a avaliação.
- movieid (obrigatório) - Identificador único do filme que foi avaliado.
- rating (obrigatório) - Nota atribuída à avaliação, deve ser um número inteiro entre 1 e 5.
- review (opcional) - Texto contendo a avaliação em si.
- ispublic (opcional) - Indica se a avaliação pode ser visualizada por outros usuários. Deve ser um valor booleano (true ou false).
- specialRating (opcional): A classificação do filme de acordo com o seu genero.

Exemplo de uso:

```
{
    "rating": 4,
    "review": "Obra Prima!",
    "specialRating": 4
}
```

Resposta

Sucesso Código: 200

Conteúdo: objeto JSON com a mensagem de sucesso e as informações atualizadas da avaliação.

message: Review atualizada com sucesso!

review: objeto JSON contendo as informações atualizadas da avaliação. Exemplo de resposta:


```
[
    {
        "id": 704,
        "userid": 15,
        "movieid": 21,
        "title": "The Nun",
        "rating": 4,
        "review": "Obra Prima!",
        "ispublic": true,
        "created_at": "2023-09-18T15:05:43.901Z"
    }
]
```


Erros

Código: 400 - objeto JSON com a mensagem de erro Dados inválidos. Isso pode ocorrer caso os dados informados no corpo da requisição sejam inválidos ou estejam incompletos.

Código: 404

- objeto JSON com a mensagem de erro Review não encontrada. Isso pode ocorrer caso o identificador único informado não corresponda a uma avaliação existente no banco de dados.

Código: 500

- objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

PATCH /api/review/:id

Essa função é responsável por atualizar parcialmente uma avaliação específica no banco de dados.

Parâmetros

:id (obrigatório) - Identificador único da avaliação a ser atualizada.

Corpo da requisição: objeto JSON contendo um ou mais dos seguintes campos:

- rating - Nota atribuída à avaliação, deve ser um número inteiro entre 1 e 5.
- review - Texto contendo a avaliação em si.

- ispublic - Indica se a avaliação pode ser visualizada por outros usuários. Deve ser um valor booleano (true ou false).
- specialRating: A classificação do filme de acordo com o seu genero.

Exemplo de uso:

```
{
  "specialrating": 5
}
```

Resposta

Sucesso Código: 200

Conteúdo: objeto JSON com a mensagem de sucesso e as informações atualizadas da avaliação.

message: Review atualizada com sucesso!

review: objeto JSON contendo as informações atualizadas da avaliação.

Exemplo de resposta:

```
{
"message": "Review atualizada com sucesso!",
"review": {
"id": 32,
"userid": 1,
"movieid": 1,
"rating": 4,
"review": "Gostei muito!",
"ispublic": true,
"created_at": "2023-05-15T23:45:52.763Z",
"updated_at": "2023-05-16T01:22:10.523Z"
}
}
```

Erros

Código: 400 - objeto JSON com a mensagem de erro Dados inválidos. Isso pode ocorrer caso os dados informados no corpo da requisição sejam inválidos ou estejam incompletos.

Código: 404 - objeto JSON com a mensagem de erro Review não encontrada. Isso pode ocorrer caso o identificador único informado não corresponda a uma avaliação existente no banco de dados.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

Movies

Descrição

O sistema implementado permite que os usuários possam escolher um filme pelo titulo e escolher aleatoriamente um filme com base no genero.

Endpoints

GET /api/movie/surpriseMe

Essa função é responsável por retornar um filme aleatorio de acordo com o genero escolhido.

Parâmetros

O seguinte parâmetro deve ser enviado na URL da requisição:

:genre (obrigatório): genero do filme que deseja.

Resposta Sucesso

Código: 200 Conteúdo: objeto JSON com os seguintes campos: body: objeto json contendo todas informações do filme. Exemplo de resposta:


```
{
    "body": {
        "movie": {
            "Title": "Missing in Action 2: The Beginning",
            "Year": "1985",
            "Rated": "R",
            "Released": "01 Mar 1985",
            "Runtime": "100 min",
            "Genre": "Action, Drama, Thriller",
            "Director": "Lance Hool",
            "Writer": "Steve Bing, Larry Levinson, Arthur Silver",
            "Actors": "Chuck Norris, Soon-Tek Oh, Steven Williams",
            "Plot": "Prequel to the first Missing In Action, set in the early 1980s it shows
            "Language": "English, Vietnamese",
            "Country": "United States",
            "Awards": "N/A",
            "Poster": "https://m.media-amazon.com/images/M/MV5BOTFhNTdiNDQtZGQ4Ny00MDA1LTg1
            "Ratings": [
                {
                    "Source": "Internet Movie Database",
                    "Value": "5.3/10"
                }
```

```
            ],
            "Metascore": "N/A",
            "imdbRating": "5.3",
            "imdbVotes": "8,951",
            "imdbID": "tt0089604",
            "Type": "movie",
            "DVD": "01 May 2017",
            "BoxOffice": "$10,755,447",
            "Production": "N/A",
            "Website": "N/A",
            "Response": "True"
        }
    }
}
```


Erros

Código: 404 - objeto JSON com a mensagem de erro Filme não encontrado.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

GET /api/movie/

Essa função é responsável por retornar um filme pelo titulo.

Parâmetros

O seguinte parâmetro deve ser enviado na URL da requisição:

:title (obrigatório): titulo do filme que deseja escolher.

Resposta Sucesso

Código: 200 Conteúdo: objeto JSON com os seguintes campos: body: objeto json contendo todas informações do filme. Exemplo de resposta:


```
{
    "body": {
        "movieData": {
            "Title": "Us",
            "Year": "2019",
            "Rated": "R",
            "Released": "22 Mar 2019",
            "Runtime": "116 min",
            "Genre": "Horror, Mystery, Thriller",
            "Director": "Jordan Peele",
```

```
            "Writer": "Jordan Peele",
            "Actors": "Lupita Nyong'o, Winston Duke, Elisabeth Moss",
            "Plot": "A family's serene beach vacation turns to chaos when their doppelgänger
            "Language": "English",
            "Country": "United States, China, Japan",
            "Awards": "85 wins & 132 nominations",
            "Poster": "https://m.media-amazon.com/images/M/MV5BZTliNWJhM2YtNDc1MC00YTk1LWE2N
            "Ratings": [
                {
                    "Source": "Internet Movie Database",
                    "Value": "6.8/10"
                },
                {
                    "Source": "Rotten Tomatoes",
                    "Value": "93%"
                },
                {
                    "Source": "Metacritic",
                    "Value": "81/100"
                }
            ],
            "Metascore": "81",
            "imdbRating": "6.8",
            "imdbVotes": "322,220",
            "imdbID": "tt6857112",
            "Type": "movie",
            "DVD": "04 Jun 2019",
            "BoxOffice": "$175,084,580",
            "Production": "N/A",
            "Website": "N/A",
            "Response": "True"
        },
        "movie": {
            "medianotas": "2.43"
        }
    }
}
```


Erros

Código: 404 - objeto JSON com a mensagem de erro Filme não encontrado.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados.

List

Descrição

O CRUD implementado é um sistema básico que permite que os usuários gerenciem listas de filmes. As funções incluem criar uma nova list, selecionar todas as listas existentes, selecionar uma lista específica por seu ID, apagar uma lista, atualizar uma lista por completo e atualizar parcialmente uma lista.

Endpoints

POST /api/list

Essa função é responsável por criar uma nova lista no banco de dados.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

- name (obrigatório): O nome da lista.
- description (obrigatório): Uma breve descrição sobre a lista.
- movieTitles (obrigatório): titulos dos filmes da lista.
- ispublic (opcional): Um valor booleano que indica se a avaliação deve ser pública. O padrão é false.

Exemplo de uso:

```
{
  "name": "Melhores filmes de terror 2023 2 ",
  "description": "Uma lista dos melhores filmes de terror.",
  "movieTitles": ["The medium", "X", "Pearl"],
  "isPublic": true
}
```

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Review criada com sucesso! - list: objeto json contendo todos dados da nova lista.

Exemplo de resposta:

```
{
    "message": "Lista criada com sucesso!",
    "body": {
        "list": {
            "id": 19,
            "name": "Melhores filmes de terror 2023 2 ",
            "description": "Uma lista dos melhores filmes de terror.",
            "movies": [
```

```
                "The medium",
                "X",
                "Pearl"
            ],
            "ispublic": true,
            "userid": 14,
            "created_at": "2023-09-19T15:11:22.534Z"
        }
    }
}
```


Erros

Código: 400 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Name is required: nome não foi fornecido no corpo da requisição.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

GET /api/allLists

Essa função é responsável por retornar todas as listas do usuario autenticado existentes no banco de dados.

Parâmetros Nenhum parâmetro é necessário para essa rota.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - list: objeto json contendo todos dados da lista.

Exemplo de resposta:


```
[
    {
        "id": 20,
        "user": "testandoweb4",
        "list_name": "Melhores filmes de terror 2023",
        "movie_titles": [
            "The medium",
            "X",
            "Pearl"
        ],
        "list_description": "Uma lista dos melhores filmes de terror.",
        "created_at": "2023-09-19T15:15:05.181Z"
    },
```

```
    {
        "id": 21,
        "user": "testandoweb4",
        "list_name": "Melhores filmes",
        "movie_titles": [
            "Interstellar",
            "The texas chainsaw massacre",
            "Elite squad"
        ],
        "list_description": "Uma lista dos melhores filmes",
        "created_at": "2023-09-19T15:16:01.716Z"
    }
]
```


Erros

Código: 400 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - O usuário não possui listas: Quando o banco de dados não retorna nenhuma lista.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

GET /api/listById/:id

Essa função é responsável por retornar uma lista específica do banco de dados, com base no id fornecido.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id (obrigatório): id da lista que se deseja buscar.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Lista encontrada com sucesso! - list: objeto json contendo todos dados da lista.

Exemplo de resposta:


```
{
    "message": "Lista encontrada com sucesso!",
    "body": {
        "Lista": {
            "user": "testandoweb4",
            "list_name": "Melhores filmes de terror 2023",
            "movie_titles": [
```

```
                "The medium",
                "X",
                "Pearl"
            ],
            "list_description": "Uma lista dos melhores filmes de terror.",
            "created_at": "2023-09-19T15:15:05.181Z"
        }
    }
}
```


Erros

Código: 400 - Conteúdo: objeto JSON com a mensagem de erro correspondente a um dos seguintes casos: - Não foi possível encontrar a lista com o ID fornecido: Quando o banco de dados não retorna nenhuma lista.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

DELETE /api/list/:id

Essa função é responsável por deletar uma lista específica do banco de dados.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id (obrigatório): id da lista que se deseja buscar.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Lista deletada com sucesso! - list: objeto json contendo todos dados da lista.

Exemplo de resposta:


```
{
    "message": "Lista deletada com sucesso!",
    "list": {
        "id": 20,
        "name": "Melhores filmes de terror 2023",
        "description": "Uma lista dos melhores filmes de terror.",
        "movies": [
            "The medium",
            "X",
            "Pearl"
        ],
        "ispublic": true,
```

```
        "userid": 15,
        "created_at": "2023-09-19T15:15:05.181Z"
    }
}
```


Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

PUT /api/list/:id

Essa função é responsável por atualizar completamente uma lista específica no banco de dados. #### Parâmetros

O seguinte parâmetro deve ser enviado na URL da requisição:

:id (obrigatório): id da lista que se deseja buscar.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Lista atualizada com sucesso! - list: objeto json contendo todos dados da lista.

Exemplo de resposta:


```
{
    "message": "Lista atualizada com sucesso!",
    "list": {
        "id": 21,
        "name": "Minha Lista de Filmes Atualizada",
        "description": "Uma descrição atualizada da minha lista de filmes.",
        "movies": [
            "Inception",
            "The Dark Knight",
            "Interstellar",
            "Dunkirk"
        ],
        "ispublic": true,
        "userid": 15,
        "created_at": "2023-09-19T15:16:01.716Z"
    }
}
```

Erros

Código: 400 - Name is required: nome não foi fornecido no corpo da requisição. - O usuário não possui listas: Quando o banco de dados não retorna nenhuma lista.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

PUT /api/list/:id

Essa função é responsável por atualizar completamente uma lista específica no banco de dados. #### Parâmetros

O seguinte parâmetro deve ser enviado na URL da requisição:

:id (obrigatório): id da lista que se deseja buscar.

Exemplo de uso:

```
{
  "name": "Minha Lista de Filmes Atualizada",
  "description": "Uma descrição atualizada da minha lista de filmes.",
  "movieTitles": ["Inception", "The Dark Knight", "Interstellar", "Dunkirk"],
  "isPublic": true
}
```

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Lista atualizada com sucesso! - list: objeto json contendo todos dados da lista.

Exemplo de resposta:

```
{
    "message": "Lista atualizada com sucesso!",
    "list": {
        "id": 21,
        "name": "Minha Lista de Filmes Atualizada",
        "description": "Uma descrição atualizada da minha lista de filmes.",
        "movies": [
            "Inception",
            "The Dark Knight",
            "Interstellar",
            "Dunkirk"
        ],
        "ispublic": true,
        "userid": 15,
```

```
        "created_at": "2023-09-19T15:16:01.716Z"
    }
}
```

Erros

Código: 400 - Name is required: nome não foi fornecido no corpo da requisição. - O usuário não possui listas: Quando o banco de dados não retorna nenhuma lista.

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

PATCH /api/list/:id

Essa função é responsável por atualizar parcialmente uma lista específica no banco de dados.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id (obrigatório): id da lista que se deseja buscar.

Exemplo de uso:

```
{
  "movieTitles": ["Inception", "The Dark Knight", "Interstellar"]
}
```

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Lista atualizada com sucesso! - list: objeto json contendo todos dados da lista.

Exemplo de resposta:

```
{
    "message": "Lista atualizada com sucesso!",
    "list": {
        "id": 21,
        "name": "Minha Lista de Filmes Atualizada",
        "description": "Uma descrição atualizada da minha lista de filmes.",
        "movies": [
            "Inception",
            "The Dark Knight",
            "Interstellar"
        ],
        "ispublic": true,
```

```
        "userid": 15,
        "created_at": "2023-09-19T15:16:01.716Z"
    }
}
```


Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

Comment
Descrição O CRUD implementado é um sistema básico que permite que os usuários gerenciem comentarios de avaliações. As funções incluem comentar uma avaliação, selecionar comentarios específicos por usuario, selecionar comentarios especificos por avaliação, apagar um comentario, atualizar um comentario.
Endpoints

POST /api/comment

Essa função é responsável por criar um nova comentario no banco de dados.

Parâmetros Os seguintes parâmetros devem ser enviados no corpo da requisição:

- reviewId (obrigatório): O id da review desejada.
- comment (obrigatório): O comentário sobre a avaliação.

Exemplo de uso:

```
{
  "reviewId": 716,
  "comment": "Gostei da sua review!"
}
```

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Comentario criado com sucesso! - body: objeto json contendo todos dados do novo comentario.

Exemplo de resposta:

```
{
    "message": "Comentário criado com sucesso!",
    "body": {
        "comment": {
            "id": 9,
            "reviewid": 716,
            "userid": 15,
            "comment": "Gostei da sua review!",
            "createdat": "2023-09-19T15:33:10.659Z"
        }
    }
}
```

Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

GET /api/comment/user

Essa função é responsável por retornar todos os comentarios de um determinado usuario.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id(obrigatório) - id do usuario que deseja pesquisar os comentarios.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - body: objeto json contendo todos dados dos comentario.

Exemplo de resposta:

```
[
    {
        "id": 8,
```

```
        "reviewid": 658,
        "movieid": 21,
        "title": "The Nun",
        "comment": "asdasdasdas",
        "createdat": "2023-09-16T21:56:09.002Z"
    },
    {
        "id": 7,
        "reviewid": 659,
        "movieid": 21,
        "title": "The Nun",
        "comment": "Amei a sua review!",
        "createdat": "2023-09-16T21:31:58.434Z"
    },
    {
        "id": 6,
        "reviewid": 659,
        "movieid": 21,
        "title": "The Nun",
        "comment": "Odiei a sua review!",
        "createdat": "2023-09-16T21:31:46.654Z"
    },
    {
        "id": 5,
        "reviewid": 659,
        "movieid": 21,
        "title": "The Nun",
        "comment": "Pessimo comentario",
        "createdat": "2023-09-16T21:26:29.701Z"
    },
    {
        "id": 9,
        "reviewid": 716,
        "movieid": 31,
        "title": "Pearl",
        "comment": "Gostei da sua review!",
        "createdat": "2023-09-19T15:33:10.659Z"
    }
]
```


Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

GET /api/comment/review

Essa função é responsável por retornar todos os comentarios de uma determinada review.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id(obrigatório) - id da review que deseja pesquisar os comentarios.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - body: objeto json contendo todos dados dos comentario.

Exemplo de resposta:


```
[
    {
        "id": 9,
        "userid": 15,
        "username": "testandoweb4",
        "comment": "Gostei da sua review!",
        "createdat": "2023-09-19T15:33:10.659Z"
    }
]
```


Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

DELETE /api/comment/:id

Essa função é responsável por deletar um comentario específico do banco de dados.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id(obrigatório) - id do comentario que deseja deletar o comentario.

Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Comentário deletado com sucesso! - body: objeto json contendo todos dados dos comentario.

Exemplo de resposta:

```
{
    "message": "Comentário deletado com sucesso!",
    "comment": {
        "id": 9,
        "reviewid": 716,
        "userid": 15,
        "comment": "Gostei da sua review!",
        "createdat": "2023-09-19T15:33:10.659Z"
    }
}
```


Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.

PUT /api/comment/:id

Essa função é responsável por atualizar um comentario específico do banco de dados.

Parâmetros O seguinte parâmetro deve ser enviado na URL da requisição:

:id(obrigatório) - id do comentario que deseja atualizar o comentario.

Exemplo de uso:


```
{
    "comment": "Eu amei esse filme"
}
```


Resposta Sucesso - Código: 201 - Conteúdo: objeto JSON com os seguintes campos: - message: Comentário atualizado com sucesso! - body: objeto json contendo todos dados dos comentario.

Exemplo de resposta:


```
{
    "message": "Comentário atualizado com sucesso!",
    "comment": {
        "id": 6,
        "reviewid": 659,
        "userid": 15,
        "comment": "Eu amei esse filme",
        "createdat": "2023-09-16T21:31:46.654Z"
```

```
    }
}
```

Erros

Código: 500 - objeto JSON com a mensagem de erro Internal server error. Isso pode ocorrer em caso de falha ao executar alguma operação no banco de dados ou ao gerar o token de autenticação.
