**Documentação da API: Locadora de Filmes**

Este documento descreve a API para gerenciamento de uma locadora de filmes, incluindo clientes, filmes e locações. Esta API é desenvolvida com Node.js e Express.js, interagindo com um banco de dados MySQL.

**1. Primeiros Passos**

Esta seção detalha como configurar e executar a API da locadora de filmes em sua máquina local.

**1.1 Pré-requisitos**

Antes de começar, certifique-se de ter o seguinte instalado pelo prompt:

* **Node.js**: node install
* **Npm Express**: npm express install
* **npm** : Vem incluído com o Node.js.
* **Mysql2**: node install mysql2.
* **Banco de Dados MySQL**: Ou qualquer outro banco de dados compatível com seu conector db.js.

**1.2 Instalação**

1. **Salve o Código**: Salve o código da API fornecido (do seu arquivo app.js) em um arquivo chamado app.js em seu diretório de projeto.
2. **Crie db.js**: Crie um arquivo chamado db.js no mesmo diretório para lidar com sua conexão de banco de dados. Abaixo está um exemplo usando mysql2/promise:

JavaScript

const mysql = require('mysql2/promise');

const pool = mysql.createPool({

host: 'localhost',

user: 'seu\_usuario\_mysql', // Substitua pelo seu nome de usuário MySQL

password: 'sua\_senha\_mysql', // Substitua pela sua senha MySQL

database: 'nome\_do\_seu\_banco\_de\_dados', // Substitua pelo nome do seu banco de dados

waitForConnections: true,

connectionLimit: 10,

queueLimit: 0

});

module.exports = pool;

**Lembre-se de substituir seu\_usuario\_mysql, sua\_senha\_mysql e nome\_do\_seu\_banco\_de\_dados pelas suas credenciais reais do MySQL.**

1. **Configuração do Banco de Dados**: Você precisará criar as tabelas necessárias em seu banco de dados MySQL. Execute as seguintes instruções SQL CREATE TABLE:

SQL

CREATE TABLE cliente (

id INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(255) NOT NULL,

cpf VARCHAR(14) UNIQUE NOT NULL,

telefone VARCHAR(20),

email VARCHAR(255)

);

CREATE TABLE filmes (

id INT AUTO\_INCREMENT PRIMARY KEY,

titulo VARCHAR(255) NOT NULL,

genero VARCHAR(100),

ano\_Lancamento INT,

classificacao\_Indicativo VARCHAR(10),

estoque\_Disponivel INT NOT NULL DEFAULT 0

);

CREATE TABLE locacao (

id INT AUTO\_INCREMENT PRIMARY KEY,

id\_cliente INT NOT NULL,

data\_locacao DATE NOT NULL,

data\_devolucao DATE,

FOREIGN KEY (id\_cliente) REFERENCES cliente(id)

);

**1.3 Executando a API**

Para iniciar o servidor da API, abra seu terminal no diretório do projeto e execute:

node --watch.\\server.js

Você deverá ver a mensagem: Servidor rodando na porta http://localhost:3000/. A API estará então acessível em http://localhost:3000.

**2. Referência dos Endpoints da API**

**URL Base**

http://localhost:3000

**2.1 Clientes (/cliente)**

**GET /cliente**

* **Descrição**: Recupera uma lista de todos os clientes cadastrados.
* **Status da Resposta**: 200 OK
* **Exemplo de Resposta**:

JSON

[

{

"id": 1,

"nome": "João Silva",

"cpf": "111.222.333-44",

"telefone": "9999-8888",

"email": "joao.silva@example.com"

},

{

"id": 2,

"nome": "Maria Souza",

"cpf": "555.666.777-88",

"telefone": "7777-6666",

"email": "maria.souza@example.com"

}

]

**GET /cliente/:id**

* **Descrição**: Recupera um único cliente pelo seu ID exclusivo.
* **Status da Resposta**: 200 OK se encontrado, 404 Not Found se o cliente não existir.
* **Exemplo de Resposta (Sucesso)**:

JSON

{

"id": 1,

"nome": "João Silva",

"cpf": "111.222.333-44",

"telefone": "9999-8888",

"email": "joao.silva@example.com"

}

* **Exemplo de Resposta (Não Encontrado)**:
* Cliente com id: [id] não encontrado!

**POST /cliente**

* **Descrição**: Adiciona um novo cliente ao banco de dados.
* **Corpo da Requisição**

JSON

{

"nome": "Carlos Pereira",

"cpf": "999.888.777-66",

"telefone": "5555-4444",

"email": "carlos.pereira@example.com"

}

* **Status da Resposta**: 201 Created
* **Exemplo de Resposta**:

JSON

{

"nome": "Carlos Pereira",

"cpf": "999.888.777-66",

"telefone": "5555-4444",

"email": "carlos.pereira@example.com",

"id": 3

}

**PUT /cliente/:id**

* **Descrição**: Atualiza as informações de um cliente existente.
* **Corpo da Requisição (application/json)**: Todos os campos são obrigatórios para a atualização.

JSON

{

"nome": "João Silva Atualizado",

"cpf": "111.222.333-44",

"telefone": "1111-2222",

"email": "joao.atualizado@example.com"

}

* **Status da Resposta**: 200 OK se bem-sucedido, 404 Not Found se o cliente não existir.
* **Exemplo de Resposta (Sucesso)**:

JSON

{

"nome": "João Silva Atualizado",

"cpf": "111.222.333-44",

"telefone": "1111-2222",

"email": "joao.atualizado@example.com",

"id": 1

}

* **Exemplo de Resposta (Não Encontrado)**:
* Cliente com id: [id] não encontrado!

**DELETE /cliente/:id**

* **Descrição**: Exclui um cliente do banco de dados.
* **Status da Resposta**: 204 No Content se bem-sucedido, 404 Not Found se o cliente não existir.
* **Exemplo de Resposta (Sucesso)**: (Nenhum conteúdo retornado para 204)
* Cliente deletado com sucesso!
* **Exemplo de Resposta (Não Encontrado)**:
* Cliente não encontrado para deletar!

**2.2 Filmes (/filmes)**

**GET /filmes**

* **Descrição**: Recupera uma lista de todos os filmes cadastrados.
* **Status da Resposta**: 200 OK
* **Exemplo de Resposta**:

JSON

[

{

"id": 1,

"titulo": "A Origem",

"genero": "Ficção Científica",

"ano\_Lancamento": 2010,

"classificacao\_Indicativo": "14",

"estoque\_Disponivel": 5

},

{

"id": 2,

"titulo": "O Senhor dos Anéis: A Sociedade do Anel",

"genero": "Fantasia",

"ano\_Lancamento": 2001,

"classificacao\_Indicativo": "12",

"estoque\_Disponivel": 3

}

]

**GET /filmes/:id**

* **Descrição**: Recupera um único filme pelo seu ID exclusivo.
* **Status da Resposta**: 200 OK se encontrado, 404 Not Found se o filme não existir.
* **Exemplo de Resposta (Sucesso)**:

JSON

{

"id": 1,

"titulo": "A Origem",

"genero": "Ficção Científica",

"ano\_Lancamento": 2010,

"classificacao\_Indicativo": "14",

"estoque\_Disponivel": 5

}

* **Exemplo de Resposta (Não Encontrado)**:
* filmes com id: [id] não encontrado!

**POST /filmes**

* **Descrição**: Adiciona um novo filme ao banco de dados.
* **Corpo da Requisição:**

JSON

{

"titulo": "Interestelar",

"genero": "Ficção Científica",

"ano\_Lancamento": 2014,

"classificacao\_Indicativo": "10",

"estoque\_Disponivel": 7

}

* + **Observação**: O *endpoint* POST /filmes no seu código original espera apenas titulo, genero, ano\_Lancamento e classificacao\_Indicativo. Considere atualizar seu código para incluir estoque\_Disponivel no INSERT se desejar gerenciar o estoque desde a criação do filme.
* **Status da Resposta**: 201 Created
* **Exemplo de Resposta**:

JSON

{

"titulo": "Interestelar",

"genero": "Ficção Científica",

"ano\_Lancamento": 2014,

"classificacao\_Indicativo": "10",

"estoque\_Disponivel": 7,

"id": 3

}

**PUT /filmes/:id**

* **Descrição**: Atualiza as informações de um filme existente.
* **Corpo da Requisição (application/json)**: Todos os campos são obrigatórios para a atualização.

JSON

{

"titulo": "A Origem (Edição Especial)",

"genero": "Ficção Científica",

"ano\_Lancamento": 2010,

"classificacao\_Indicativo": "14",

"estoque\_Disponivel": 4

}

* **Status da Resposta**: 200 OK se bem-sucedido, 404 Not Found se o filme não existir.
* **Exemplo de Resposta (Sucesso)**:

JSON

{

"titulo": "A Origem (Edição Especial)",

"genero": "Ficção Científica",

"ano\_Lancamento": 2010,

"classificacao\_Indicativo": "14",

"estoque\_Disponivel": 4,

"id": 1

}

* **Exemplo de Resposta (Não Encontrado)**:
* filmes com id: [id] não encontrado!

**DELETE /filmes/:id**

* **Descrição**: Exclui um filme do banco de dados.
* **Status da Resposta**: 204 No Content se bem-sucedido, 404 Not Found se o filme não existir.
* **Exemplo de Resposta (Sucesso)**: (Nenhum conteúdo retornado para 204)
* filmes deletado com sucesso!
* **Exemplo de Resposta (Não Encontrado)**:
* filmes não encontrado para deletar!

**2.3 Locações (/locacao)**

**GET /locacao**

* **Descrição**: Recupera uma lista de todas as locações de filmes registradas.
* **Status da Resposta**: 200 OK
* **Exemplo de Resposta**:

JSON

[

{

"id": 1,

"id\_cliente": 1,

"data\_locacao": "2023-01-10",

"data\_devolucao": "2023-01-15"

},

{

"id": 2,

"id\_cliente": 2,

"data\_locacao": "2023-02-01",

"data\_devolucao": null

}

]

**GET /locacao/:id**

* **Descrição**: Recupera um único registro de locação pelo seu ID exclusivo.
* **Status da Resposta**: 200 OK se encontrado, 404 Not Found se o registro de locação não existir.
* **Exemplo de Resposta (Sucesso)**:

JSON

{

"id": 1,

"id\_cliente": 1,

"data\_locacao": "2023-01-10",

"data\_devolucao": "2023-01-15"

}

* **Exemplo de Resposta (Não Encontrado)**:
* locacao com id: [id] não encontrado!

**POST /locacao**

* **Descrição**: Cria um novo registro de locação.
* **Corpo da Requisição (application/json)**:

JSON

{

"id\_cliente": 1,

"data\_locacao": "2023-03-20",

"data\_devolucao": null

}

* **Status da Resposta**: 201 Created
* **Exemplo de Resposta**:

JSON

{

"id\_cliente": 1,

"data\_locacao": "2023-03-20",

"data\_devolucao": null,

"id": 3

}

**PUT /locacao/:id**

* **Descrição**: Atualiza um registro de locação existente. Isso é tipicamente usado para definir a data\_devolucao.
* **Corpo da Requisição (application/json)**: Todos os campos são obrigatórios para a atualização.

JSON

{

"id\_cliente": 1,

"data\_locacao": "2023-03-20",

"data\_devolucao": "2023-03-25"

}

* **Status da Resposta**: 200 OK se bem-sucedido, 404 Not Found se o registro de locação não existir.
* **Exemplo de Resposta (Sucesso)**:

JSON

{

"id\_cliente": 1,

"data\_locacao": "2023-03-20",

"data\_devolucao": "2023-03-25",

"id": 3

}

* **Exemplo de Resposta (Não Encontrado)**:
* locacao com id: [id] não encontrado!

**DELETE /locacao/:id**

* **Descrição**: Exclui um registro de locação do banco de dados.
* **Status da Resposta**: 204 No Content se bem-sucedido, 404 Not Found se o registro de locação não existir.
* **Exemplo de Resposta (Sucesso)**: (Nenhum conteúdo retornado para 204)
* locacao deletado com sucesso!
* **Exemplo de Resposta (Não Encontrado)**:
* locacao não encontrado para deletar!

**3. Tratamento de Erros**

A API inclui um tratamento básico de erros. Se uma consulta ao banco de dados falhar, o erro será registrado no console e um 500 Internal Server Error será retornado ao cliente. Para requisições GET, PUT e DELETE onde um ID é fornecido, um status 404 Not Found é retornado se o recurso especificado não existir.