Desafio

Seja bem-vindo! Este desafio foi projetado para avaliar a sua capacidade técnica como candidato(a) à vaga de **Desenvolvedor(a) Back-end - Jr (Presencial)**.

Descrição

Desenvolva uma API em Node.js para gerenciar uma lista de desejos de filmes. A API deve permitir que o usuário, depois de autenticado, adicione filmes, mova-os entre diferentes estados (a assistir, assistido, avaliado, recomendado/não recomendado) e consulte o histórico completo de um filme. O foco é a construção de um **middleware de logs eficiente e expansível**, que vincule um identificador único a cada filme e registre todas as ações realizadas.

Requisitos

1. **Autenticação**:
   - Use autenticação básica (usuário e senha fixos) para proteger os endpoints da API. N*ão é necessário criar meios de criação de novos usuários!*.

2. **Integração com API Externa**:
   - Integre a API com uma API pública de filmes **The Movie Database - TMDB**.
   - Ao adicionar um filme, busque informações como título, sinopse, ano de lançamento e gênero na API externa (qualquer informação adicional pode ser adicionada desde que faça sentido para o projeto).

3. **Middleware de Logs**:
   - Crie um middleware que registre todas as requisições recebidas, incluindo:

- Método HTTP, URL, status da resposta, timestamp e identificador único do filme (se aplicável, o filme deve ser válido, ou seja, existente na API sugerida).
- Para cada filme adicionado, gere um **identificador único** (ex: UUID) e o vincule a todas as ações futuras relacionadas a esse filme (ex: mover para assistido, avaliar, etc.).
- Armazene os logs em banco de dados (ex: SQLite ou MongoDB).

4. **Estados do Filme**:
   - Um filme pode estar em um dos seguintes estados:
     1. **A assistir** (estado inicial ao adicionar, sempre).
     2. **Assistido**.
     3. **Avaliado** (com uma nota de 0 a 5).
     4. **Recomendado** ou **Não recomendado**.
   - Crie endpoints para mover o filme entre esses estados.

5. **Histórico de um Filme**:
   - Crie um endpoint que retorne o histórico completo de um filme, incluindo todas as ações realizadas (ex: adicionado, movido para assistido, avaliado, etc.) com timestamps e a identificação do usuário que realizou a ação.

6. **Swagger**:
   - Documente a API usando Swagger, incluindo exemplos de requisições e respostas para todos os endpoints.

Endpoints sugeridos

1. **Filmes**:

- `POST /filme` → Adiciona um filme à lista de desejos. Busca informações na API externa e gera um identificador único.
- `GET /filme` → Lista todos os filmes na lista de desejos.
- `GET /filme/:id` → Retorna detalhes de um filme específico.
- `PUT /filme/:id/estado` → Move o filme para um novo estado (ex: assistido, avaliado, recomendado).
- `POST /filme/:id/avaliar` → Avalia o filme com uma nota de 0 a 5.
- `GET /filme/:id/historico` → Retorna o histórico completo de um filme.

2. **Logs**:
- `GET /logs` → Retorna todos os logs registrados (para fins de debug).

Exemplo de funcionamento

1. **Adicionar um Filme**:
- O usuário faz uma requisição `POST /filme` com o nome do filme.
- A API busca informações na API externa (TMDB) e adiciona o filme à lista de desejos com o estado "A assistir".
- Um identificador único é gerado e vinculado ao filme.
- O middleware registra a ação no log.

2. **Mover para Assistido**:
- O usuário faz uma requisição `PUT /filme/:id/estado` com o novo estado "Assistido".
- O middleware registra a ação no log, vinculada ao identificador único do filme.

3. **Avaliar o Filme**:
- O usuário faz uma requisição `POST /filme/:id/avaliar` com uma nota de 0 a 5.
- O middleware registra a ação no log, vinculada ao identificador único do filme.

4. **Consultar Histórico**:
   - O usuário faz uma requisição `GET /filme/:id/historico` e recebe o histórico completo do filme, incluindo todas as ações realizadas.

Diferenciais

1. **Expansibilidade do Middleware**:
   - Projete o middleware de logs para ser expansível, permitindo a adição de novos tipos de logs no futuro (ex: logs de erros, logs de desempenho).

2. **Validações**:
   - Adicione validações para garantir que um filme só possa ser movido para "Avaliado" após ser "Assistido", e para "Recomendado/Não recomendado" após ser "Avaliado".

3. **Testes Unitários**:
   - Escreva testes unitários para o middleware de logs e para os endpoints principais.

4. **Adicionar Paginação**:
   - Para a listagem de filmes (`GET /filme`), implemente paginação para evitar retornar muitos dados de uma vez.

5. **Filtrar por Estado**:
   - Adicione um parâmetro de filtro no endpoint `GET /filme` para listar filmes por estado (ex: apenas filmes "A assistir").

6. **Testes de Integração**:

- Além dos testes unitários, escreva testes de integração para garantir que a API externa e o middleware de logs funcionem corretamente.

7. **Docker**:
   - Containerize a aplicação com Docker.

Critérios de Avaliação

- ☐ Boas práticas
- ☐ Reprodutibilidade de ambiente
- ☐ Eficiência

Entrega

A entrega do desafio deve ser enviada por email (querosercarefy@carefy.com.br) com as soluções dos problemas (arquivos, links, documentos etc.). São esperados para este desafio os seguintes entregáveis:

- ☐ Todos os códigos e orientações podem ser disponibilizados a partir de um repositório GIT.

Bom trabalho!