Desafio Back-end | Sistema de Biblioteca Digital

Descrição Geral

Você deve desenvolver uma API RESTful para gerenciar uma plataforma de biblioteca digital. Usuários autenticados poderão cadastrar, buscar, visualizar e gerenciar diferentes tipos de materiais (livros, artigos, vídeos), associando-os a autores (pessoas ou instituições).

O sistema deve ser seguro, bem validado, seguir boas práticas REST, e conter documentação e testes.

Considere diferenciais como deploy, documentação interativa e um endpoint GraphQL (diferencial).

Requisitos Funcionais

1. Autenticação

Implemente autenticação via email e senha.
Apenas usuários autenticados podem cadastrar, atualizar ou remover materiais.

2. Materiais e Tipos

Usuários podem cadastrar diferentes tipos de materiais: livros, artigos, vídeos, etc.
Cada material deve ter campos genéricos e específicos do tipo cadastrado.
Todo material deve estar obrigatoriamente associado a um autor, que pode ser uma pessoa ou instituição.
O sistema deve permitir cadastro e vínculo de autores a materiais.

3. Permissões de Acesso

Cada material só pode ser alterado ou removido pelo usuário que o cadastrou.
Usuários podem visualizar materiais públicos de outros usuários.
Implemente regras de autorização para proteger rotas sensíveis.

4. Status

Todo material deve ter um campo status com diferentes valores possíveis (exemplo: rascunho, publicado, arquivado).
O status deve ser validado e controlado pela API.

5. Cadastro com Dados de API Externa

Ao cadastrar um livro, permita informar um identificador externo (ISBN).
Caso informado, busque informações na API OpenLibrary Books para preencher automaticamente campos como título e número de páginas, se não forem fornecidos

pelo usuário.
https://openlibrary.org/dev/docs/api/books

6. Busca e Paginação

Implemente endpoint para buscar materiais por título, autor ou descrição.
Os resultados devem ser paginados.

7. Testes

Implemente testes automatizados cobrindo:

- Validações de dados.
- Permissões de acesso.
- Funcionalidades de busca e paginação.
- Consumo da API externa.

8. Documentação

A API deve ser documentada. O README deve explicar rotas, autenticação e exemplos de uso.
Diferencial: documentação interativa (Swagger/Rswag) ou mini frontend para interação.

9. GraphQL

Diferencial: implemente ao menos uma rota de consulta com GraphQL para acessar dados de materiais e autores.

10. Banco de Dados

O projeto deve rodar em Postgres ou MySQL.

Validações Obrigatórias

Usuário

Email

- Deve ser único e válido (formato de email).
- Obrigatório no cadastro.

Senha

- Obrigatória no cadastro.
- Deve ter no mínimo 6 caracteres.

Material (todos os tipos)

Título

- Obrigatório.
- Não pode ser nulo ou vazio.
- Deve ter entre 3 e 100 caracteres.

Descrição

- Opcional, mas se informada, deve ter até 1000 caracteres.

Status

- Obrigatório.
- Só pode receber valores válidos (ex: rascunho, publicado, arquivado).

Autor

- Obrigatório.
- Todo material deve estar associado a um autor (pessoa ou instituição).

Usuário criador

- Obrigatório.
- Todo material deve estar associado ao usuário que cadastrou.

Livro

ISBN

- Obrigatório.
- Deve ser único na base de dados.

- Deve ter exatamente 13 caracteres numéricos (padrão ISBN-13).

Número de páginas

- Obrigatório.
- Deve ser maior que zero.

Artigo

DOI

- Obrigatório.
- Deve ser único na base de dados.
- Deve seguir o formato padrão de DOI (ex: 10.1000/xyz123).

Vídeo

Duração (minutos)

- Obrigatória.
- Deve ser um número inteiro maior que zero.

Autor (Pessoa)

Nome

- Obrigatório.
- Entre 3 e 80 caracteres.

Data de nascimento

- Obrigatória.
- Deve ser uma data válida.
- Não pode ser futura.

Autor (Instituição)

Nome

- Obrigatório.
- Entre 3 e 120 caracteres.

Cidade

- Obrigatória.
- Entre 2 e 80 caracteres.

Validações Relacionais e Regras de Negócio

- Um material não pode ser excluído ou alterado por outro usuário que não seja o criador.
- Não deve ser possível cadastrar dois materiais com o mesmo identificador único (ISBN para livros, DOI para artigos).
- Não deve ser possível associar um material a um autor inexistente.
- Um autor pode ser associado a vários materiais, mas um material deve ter apenas um autor.
- O status de um material só pode ser alterado para um valor válido.
- Os campos obrigatórios devem gerar erro claro e status HTTP apropriado na ausência.

Critérios de Avaliação

- Organização, clareza e padrão REST das rotas e controllers.
- Qualidade das validações e testes.
- Uso correto de associações, permissões e validações.
- Uso correto de padrões do Rails.
- Clareza e detalhamento da documentação.

- Implementação dos diferenciais.

Diferenciais

- Deploy online (Heroku, Render, Fly, Railway etc).
- Documentação interativa (Swagger, Postman, etc).
- Mini front-end simples para consumir a API.
- Cobertura de testes acima de 80%.
- Endpoint GraphQL.

Entrega

- Link para o repositório (GitHub, Gitlab, etc).
- README com setup do projeto, exemplos de uso, instruções de autenticação e explicação das regras de negócio implementadas.

Dicas Gerais

- Reflita sobre como modelar materiais genéricos e específicos, e como associar autores de tipos diferentes a eles.
- Garanta as regras de acesso no código e nos testes.
- Use padrões REST, status HTTP e mensagens de erro claras.
- Estruture o projeto para facilitar manutenção e testes.

Prazo para entrega: você terá 7 dias corridos a partir da data de recebimento deste e-mail para enviar sua solução.