

Descrição do Desafio

Desenvolver uma API CRUD que:

1. Permita realizar as operações básicas de um CRUD em uma entidade chamada Gato/Cachorro.
 2. Consuma a API externa [TheCatAPI](#) / [TheDogAPI](#) para buscar informações sobre raças de gatos/cachorros e imagens correspondentes.
 3. Permita que o usuário do sistema busque imagens de gatos/cachorros com base em suas raças.
-

Requisitos Funcionais

1. Entidade Gato / Cachorro:

- ID (gerado automaticamente).
- Nome do Gato/Cachorro.
- Raça .
- Imagem (referente às IMAGENS fornecidas pela [TheCatAPI](#) / [TheDogAPI](#)).

2. Operações CRUD:

- **Create:** Cadastrar um gato/cachorro, informando o nome, raça, e fornecendo uma imagem através da api correspondente.
- **Read:** Listar todos os gatos/cachorros cadastrados e buscar um gato/cachorro específico por ID.
- **Update:** Atualizar os dados de um gato/cachorro.
- **Delete:** Excluir um gato/cachorro.

3. Integração com [TheCatAPI](#) / [TheDogAPI](#):

- Criar um endpoint para buscar as raças disponíveis na [TheCatAPI](#) / [TheDogAPI](#) e armazenar no banco de dados local (opcional, como pré-processamento).
- Criar um endpoint para buscar imagens de gatos/cachorros com base na raça escolhida.

4. Filtro por Raça:

- Permitir que os usuários filtrem os gatos/cachorros cadastrados pelo nome da raça.

5. Documentação:

- Incluir Swagger para documentar os endpoints.
 - Incluir Postman para documentar os endpoints. (Enviar o curl)
-

Requisitos Técnicos

1. A API deve ser desenvolvida em **Java** usando **Spring Framework**.
 2. O banco de dados pode ser **MongoDB** ou **MYSQL**.
 3. Os dados das raças devem ser obtidos através do endpoint /breeds da **TheCatAPI** / **TheDogAPI**.
 4. Implementar tratamento de erros (ex.: o que acontece se a API externa estiver fora do ar?).
 5. thecatapi.apikey ou thedogapi.apikey deve ser uma variável de ambiente onde a apikey correspondente deve ser informada
-

Critérios de Avaliação

1. Boa organização do código (camadas separadas para controller, service e repository). (Exceeds)
 2. Utilização de boas práticas de API REST.
 3. Funcionalidade completa e funcionamento do CRUD.
 4. Integração bem-sucedida com a API externa.
 5. Uso do Swagger para documentação.
 6. Qualidade dos tratamentos de erro e mensagens amigáveis no retorno.
 7. Opcional: Testes unitários para as principais funcionalidades. (Exceeds)
 8. Mais de 4 status code (Exceeds)
-

Dicas para os Estagiários

- **API Key:** É necessário criar uma conta em **TheCatAPI** / **TheDogAPI** para obter a chave de acesso (API Key).
 - **Swagger:** Adicione a configuração para que os endpoints sejam claros e bem documentados.
 - **Banco de Dados:** Escolha um banco de dados que permita filtrar registros por raça de maneira eficiente.
 - **C# HTTP Client:** Utilize o HttpClient para integrar com a **TheCatAPI** / **TheDogAPI**.
-

Exemplo de Endpoints

1. **CRUD Gato/Cachorro:**
 - POST /api/gatos - Cadastrar um gato.
 - GET /api/gatos - Listar todos os gatos.
 - GET /api/gatos/{id} - Buscar um gato pelo ID.

- PUT /api/gatos/{id} - Atualizar dados de um gato.
- DELETE /api/gatos/{id} - Excluir um gato.
- Para quem usar TheDogAPI, onde se lê “gatos” será “cachorros”

2. **Integração com TheCatAPI / TheDogAPI:**

- GET /api/raças - Listar raças disponíveis (dados da **TheCatAPI** / TheDogAPI).
- GET /api/imagens?raça={breed_id} - Buscar imagens de gatos com base na raça.