# Seleção Labens 2024 - Desafio Backend

#### Desafio

## Criar uma API para o Gerenciamento de Tarefas (to-do list)

## Descrição do Projeto

API para gerenciar uma lista de tarefas (to-do list). A API deve permitir que os usuários criem, atualizem, excluam e visualizem tarefas. Cada tarefa deve ter um título, uma descrição, um prazo (data planejada para conclusão), uma data da conclusão, e uma situação (nova, em andamento, concluída ou cancelada).

## Tecnologias Relacionadas

- Django
- Django Rest Framework
- <u>Docker</u> (opcional)
- Postman/Insomnia (Tecnologia usada para testar e avaliar a aplicação)
- <u>SQLite</u> (Criado pelo proprio framework facilitando avaliação)

## Requisitos Funcionais

- 1. A API deve ter endpoints para as seguintes operações:
  - Listar todas as tarefas (com paginação, onde a requisição retorna apenas 5 tarefas por vez)
  - Função de busca que busque no título e na descrição das tarefas e retorne somente as tarefas compatíveis com a busca
  - 1.3. Criar uma nova tarefa
  - 1.4. Visualizar uma tarefa específica
  - 1.5. Atualizar uma tarefa existente
  - 1.6. Excluir uma tarefa
- 2. Uma tarefa devem ter os seguintes campos:
  - 2.1. Título (máximo de 100 caracteres)
  - 2.2. Descrição (opcional, 250 caracteres)
  - 2.3. Prazo (data)
  - 2.4. Data Conclusão (data)
  - 2.5. Situação (enum com as situações possíveis)
- Criar testes unitários, testando as funções básicas da API, utilizando APITestCase.

- 3.1. Testar as requisições que serão feitas (GET, POST, DELETE. PATCH), verificando se o *status code* foi correto e se a operação solicitada realmente foi executada
- 3.2. Testar pelo menos um caso de exceção
  - 3.2.1. Ex: O json de uma tarefa em um método POST é mandado sem um campo obrigatório, deve se verificar que a requisição falhou e que o objeto não foi criado.
- 4. Criação de script Django para povoar o banco de dados.
  - 4.1. O script deve ser criando usando Django Command
  - 4.2. O script deve criar 15 tarefas com prazos, datas e situações variadas
- 5. **Opcional:** Aplicação rodando via docker.
  - 5.1. Crie um script docker-compose para colocar a API no ar!

## Critérios de Avaliação

- Funcionalidade: A API deve funcionar conforme os requisitos especificados.
- Estrutura do Código: O código deve seguir as melhores práticas de desenvolvimento em Django Rest Framework, incluindo organização, legibilidade e uso eficiente das funcionalidades do framework.
- Testes Unitários: Os testes unitários devem cobrir os principais casos de uso dos endpoints da API.
- Boas Práticas de Desenvolvimento: O código deve seguir boas práticas de desenvolvimento, incluindo tratamento de erros adequado, uso de ambiente virtual, uso de variáveis de ambiente, uso de serializers, viewsets e outros recursos do Django Rest Framework de forma apropriada.
  - O uso de Git e GitHub é obrigatório. Utilize as boas práticas do GitFlow.
- Documentação: Os endpoints da API devem ser documentados de forma clara. Além de fornecer todo o passo a passo de como executar a aplicação.

#### Entrega

Você deve enviar o link para um repositório GitHub público com o código-fonte do projeto do desafio e instruções claras sobre como executar e testar a API (instruções no README.md). A atividade deve ser entregue com período de uma semana após a data de início dessa seleção.

O prazo de entrega é de **7 dias**, iniciando no dia 10/05 e encerrando no dia 17/05 às 23:59.