⚠️ INSTRUÇÕES ⚠️

☝🏻 Apenas um dos integrantes deve fazer a entrega do grupo.

⏳ A entrega deve ser feita pela atividade correspondente no classroom, entrega de outras formas **não** serão aceitas. Passado o prazo no classroom o sistema **não** admite mais entregas.

🤖 Fazer a atividade com a ajuda de IA é **diferente** da IA fazer sua atividade. Use-a com **sabedoria**, como para entender o que você fez de errado e como fazer da forma correta. Seu futuro profissional agradece.

A entrega deve ser feita por meio de um LINK do repositório no github, com indentificação do grupo (ex: `Entrega atividade ... Grupo 10`)

Construção de uma API Simples com Flask

Descrição do Projeto

O objetivo deste exercício é desenvolver uma aplicação web simples usando o framework Flask para criar uma API RESTful que gerencia usuários. A aplicação deverá ser capaz de realizar todas as quatro operações fundamentais de CRUD (Create, Read, Update, Delete) sobre um recurso de usuário. Os dados dos usuários serão armazenados temporariamente em uma estrutura de dados em memória (por exemplo, uma lista de dicionários), simulando um banco de dados simples.

Requisitos Funcionais

A API deve expor os seguintes **endpoints** (rotas) e responder aos métodos HTTP indicados:

- `POST /users` (Create)
  - Esta rota deve criar um novo usuário.
  - A requisição deve conter um corpo JSON com os campos `nome` (string) e `email` (string). Sugestão: Utilize `request.json` para obter esses dados, que serão automaticamente convertidos em um dicionário Python.
  - A aplicação deve gerar um ID numérico único para o novo usuário.
  - O novo usuário (com o ID gerado) deve ser adicionado à lista de usuários.
  - A resposta deve ser um JSON contendo os dados do usuário recém-criado e o status HTTP `201 Created`.
- `GET /users` (Read All)
  - Esta rota deve retornar uma lista JSON de todos os usuários cadastrados.
  - Se não houver usuários, deve retornar uma lista vazia `[]`.
  - A resposta deve ter o status HTTP `200 OK`.
- `GET /users/<int:user_id>` (Read Single)
  - Esta rota deve buscar um único usuário pelo seu ID.
  - O `<int:user_id>` na rota indica que o ID deve ser um número inteiro. Se um valor não numérico for fornecido, o Flask deve retornar um erro 404 Not Found automaticamente.
  - Se o `user_id` for encontrado, a resposta deve ser um JSON com os dados do usuário e o status `200 OK`.

  - Se o `user_id` não for encontrado, a resposta deve ser um JSON de erro e o status `404 Not Found`.

- `PUT /users/<int:user_id>` (Update)
  - Esta rota deve atualizar os dados de um usuário existente.
  - O `<int:user_id>` na rota identifica o usuário a ser atualizado.
  - A requisição deve conter um corpo JSON com os campos `nome` e/ou `email` a serem atualizados.
  - Se o `user_id` for encontrado, os dados do usuário devem ser atualizados. A resposta deve ser um JSON com os dados do usuário atualizado e o status `200 OK`.
  - Se o `user_id` não for encontrado, a resposta deve ser um JSON de erro e o status `404 Not Found`.

- `DELETE /users/<int:user_id>` (Delete)
  - Esta rota deve remover um usuário da lista.
  - O `<int:user_id>` na rota identifica o usuário a ser removido.
  - Se o `user_id` for encontrado, o usuário deve ser removido. A resposta deve ser um JSON de sucesso (ex: `{"message": "Usuário excluído com sucesso"}`) e o status `200 OK`.
  - Se o `user_id` não for encontrado, a resposta deve ser um JSON de erro e o status `404 Not Found`.

Requisitos Técnicos

1. Estrutura de Dados: Use uma lista global de dicionários para armazenar os usuários. Cada dicionário deve ter as chaves `id`, `nome` e `email`.
2. Geração de ID: Crie um contador simples para gerar IDs únicos. Por exemplo, uma variável global `current_id` que é incrementada a cada novo usuário criado.
3. Uso de Flask: A aplicação deve ser implementada em um único arquivo Python (ex: `app.py`).
4. Formato de Dados: Todas as requisições e respostas de dados devem usar o formato JSON. Utilize a função `jsonify` do Flask para retornar respostas JSON.
5. Tratamento de Erros: Assegure que a aplicação retorne os códigos de status HTTP corretos para cada cenário (sucesso, criação, não encontrado, etc.).