Desafio Técnico Desenvolvedor Fullstack

Contexto

A empresa fictícia **Automax** está desenvolvendo um painel interno simples para consultar os **carrinhos de compras** provenientes de um sistema externo de e-commerce.
Esses dados estão disponíveis publicamente através da Fake Store API, e o objetivo é criar uma solução que **busque essas informações**, **armazene localmente** e **permita consultá-las por meio de uma interface web**.

O teste busca avaliar sua capacidade de construir uma aplicação funcional com backend e frontend integrados, aplicando boas práticas de desenvolvimento e organização de código.

Objetivo do Desafio

Criar uma pequena aplicação **fullstack** composta por:

1. Um **backend em Python** que:
    - Consuma os dados de carrinhos a partir da Fake Store API:
      `GET https://fakestoreapi.com/carts`
    - Armazene essas informações em um **banco de dados local** (SQLite ou outro de sua preferência).
    - Exponha endpoints RESTful para consultar os carrinhos armazenados localmente.
      Exemplos:
        - `GET /carts` → retorna todos os carrinhos armazenados.
        - `GET /carts/:id` → retorna os detalhes de um carrinho específico.

2. Um **frontend web** que:
    - Consuma o backend criado e **liste os carrinhos** armazenados.
    - Cada item da listagem deve mostrar ao menos:
        - `id` do carrinho,
        - `data` da criação,
        - `userId`,
        - quantidade total de produtos
    - O layout pode ser simples, mas deve ser funcional e legível.
    - A tecnologia do front é **livre escolha** (React, Vue, Angular, HTML puro, etc.).
      *O uso de **ReactJS** será considerado um diferencial positivo.*

Requisitos Técnicos

- **Backend:** Python (preferencialmente FastAPI ou Flask).
- **Banco de Dados:** SQLite (ou outro de fácil configuração).
- **Frontend:** livre escolha (consumindo os endpoints do backend).

- **Versionamento:** código versionado em Git (GitHub, GitLab, etc.).
- **Documentação:** arquivo `README.md` com:
    - Descrição do projeto.
    - Instruções para executar o backend e frontend.
    - Breve explicação sobre decisões técnicas tomadas.

Requisitos Opcionais (Diferenciais)

- Atualização periódica dos dados dos carrinhos (ex.: a cada execução, sincronizar com a API externa ou utilizando scheduller).
- Implementação de filtros no frontend (ex.: buscar por `userId` ou intervalo de datas).
- Organização modular do código backend (camadas separadas para rotas, serviços e repositórios).
- Uso de ReactJS no front.

Entrega Esperada

O candidato deve enviar:

- Link do repositório Git contendo o código-fonte completo.
- Instruções claras de como executar o projeto localmente.
- (Opcional) Prints de tela ou vídeo mostrando a aplicação em funcionamento.