Desafio: Sistema de NPS e CSAT com Streamlit, Python e Boas Práticas DevOps

Desenvolva um aplicativo web completo para coleta e visualização de NPS (Net Promoter Score) e CSAT (Customer Satisfaction Score). O objetivo é validar conhecimentos em:

- Git (fluxo de versionamento e commits incrementais)
- Python (estrutura, modularidade e boas práticas)
- Streamlit (framework para dashboards)
- Docker (containerização)
- PostgreSQL (modelagem e consultas)
- Segurança básica (autenticação, variáveis de ambiente)
- Data Analytics & BI (ETL, métricas, visualização)
- Integração com LLM (geração programática de massa de dados)

1. Requisitos Funcionais

1. Página de Login
    - Autenticação simples (usuário/senha armazenados em tabela PostgreSQL)
    - Proteção de rota: somente usuários autenticados podem acessar o dashboard
2. Dashboard Home
    - Exibição de dois gráficos: NPS e CSAT por período (dia, semana, mês)
    - Filtros interativos para data e canal de atendimento
    - Resumo numérico: valor médio de NPS e de CSAT
3. Massa de Dados
    - Uso de um modelo de LLM (por exemplo, OpenAI, HuggingFace ou similar)
    - Geração de resultados simulados de NPS e CSAT (score 0–10 e 1–5, respectivamente)
    - Pelo menos 1.000 registros com timestamps e atributos (canal, tipo de cliente etc.)
4. Back-end & Banco
    - API ou camada de acesso a dados em Python
    - Esquema em PostgreSQL: tabelas de usuários, respostas NPS/CSAT
5. Infraestrutura & DevOps
    - Dockerfile para aplicação
    - docker-compose com serviços: app, PostgreSQL
    - Configuração via variáveis de ambiente (.env)
    - Instruções claras de build e run no README

2. Fluxo de Commits (Milestones)

1. **Inicialização do Repositório**
    - Criação da estrutura básica de pastas
    - Commit: “Projeto iniciado: estrutura inicial”
2. **Configuração do Banco**
    - Docker Compose com PostgreSQL e script de criação de tabelas
    - Commit: “Database: schema NPS/CSAT e usuários”
3. **Esqueleto do App & Autenticação**
    - Streamlit com página de login simples
    - Validação de usuário em PostgreSQL
    - Commit: “Auth: página de login implementada”
4. **Integração LLM para Dados**
    - Módulo Python para chamar LLM e popular tabela de respostas
    - Commit: “Data Gen: integração com LLM para geração de massa”
5. **Dashboard e Visualizações**
    - Gráficos NPS e CSAT usando st.line_chart ou Plotly
    - Filtros de data e canal
    - Commit: “Dashboard: visualizações NPS/CSAT”
6. **Containerização**
    - Dockerfile refinado, docker-compose final
    - Variáveis de ambiente para credenciais e tokens da LLM
    - Commit: “Docker: containerização completa”
7. **Documentação & Boas Práticas**
    - README com instruções de setup, execução e deploy
    - VERBOSE de gitignore, estrutura de branches (main, develop, feature/)
    - Commit: “Docs: guia de uso e melhores práticas”

3. Critérios de Avaliação

- Clareza e organização dos commits
- Uso de branches e pull requests (modelagem de fluxo Git)
- Tratamento de erros e validações (login, geração de dados, consulta SQL)
- Qualidade e responsividade da interface Streamlit

- Estrutura do Dockerfile e compose (camadas, tamanho da imagem)
- Segurança básica (armazenamento de segredos, sanitização de inputs)
- Documentação: instruções passo-a-passo e diagrama de arquitetura

4. Sugestões para Tornar o Desafio Mais Rico

- Integre testes unitários em Python (pytest) para funções centrais
- Adicione análise de texto livre: use LLM para categorizar comentários por sentimento

5. Requisitos & Fluxo de Commits

Envie commit para o GitHub a cada entrega funcional conforme o cronograma sugerido a seguir.

6. Cronograma Sugerido (30 Dias)

Período	Atividade
Dias 1 – 3	Inicialização do repositório (estrutura e README)
Dias 4 – 7	Configuração do banco PostgreSQL e schema
Dias 8 – 12	Esqueleto do app Streamlit + página de login
Dias 13 – 17	Integração LLM para geração de massa de dados
Dias 18 – 22	Dashboard: NPS e CSAT com filtros e gráficos
Dias 23 – 26	Dockerfile, docker-compose e variáveis de ambiente
Dias 27 – 29	Documentação, gitignore, branches e boas práticas
Dia 30	Revisão final, deploy opcional e demonstração

7. Entrega e Avaliação

- **Deadline**: 30 dias corridos a partir do recebimento do desafio.
- Compartilhe o link do repositório público até o dia final.
- Agende uma demo de 30 min para apresentação do workflow, arquitetura e dashboard.