Arquivo README/Documentação do projeto Desafio

Back-End em python:

Instale as dependências do projeto rodando no terminal/Shell:
- pip install Flask flask-restful flask-cors pandas numpy

Depois inicie o Back-End:
- python back.py

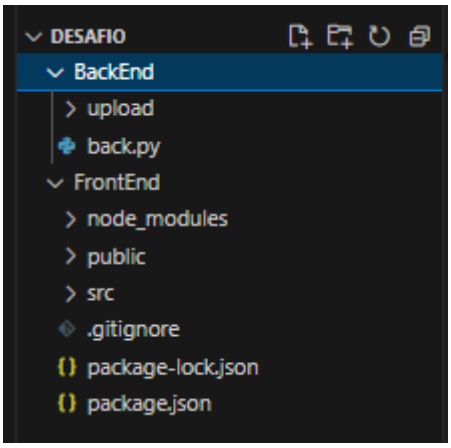
Front-End em Node, React:

Instale as dependências do projeto rodando no terminal/Shell:
- npm install

Depois inicie o Front-End:
- npm start

Atenção:
Certifique-se de ter o ambiente de desenvolvimento, Python e Node instalado e devidamente configurado.

Depois dos procedimentos terminados a estrutura de diretórios deve se parecer como na imagem abaixo:



Descrição da lógica Back-End:
Este código em Python utiliza o framework Flask para criar uma API RESTfull. A API aceita uploads de planilhas, processa essas planilhas usando a biblioteca pandas, calcula métricas relacionadas a assinaturas (Receita Recorrente Mensal - MRR e Taxa de Cancelamento - Churn Rate), e retorna os resultados em formato JSON. A aplicação também lida com erros durante o processo de upload, retornando uma mensagem de erro em caso de falha. O

código inclui suporte para Cross-Origin Resource Sharing (CORS) para permitir solicitações entre diferentes domínios. A API é iniciada e executada na porta 3001 em modo de depuração quando o script é executado como um programa independente. O endpoint da API para upload de planilhas é /subscriptions/upload.

Descrição da lógica Front-End:
Este código é um componente React que representa uma aplicação front-end. Ele cria uma interface de usuário para upload de planilhas, envia essas planilhas para um servidor Flask, processa os dados recebidos e exibe um gráfico interativo usando a biblioteca Chart.js.

Aqui está uma explicação mais detalhada do que o código faz:

Importação de Bibliotecas:
React, useState, useRef, useEffect: Hooks do React para gerenciar o estado e o ciclo de vida do componente.

axios: Biblioteca para fazer requisições HTTP.

Chart: Componente de gráfico da biblioteca Chart.js.

format e ptBR do date-fns: Biblioteca para formatação de datas.

App.css: Arquivo de estilo CSS para estilizar o aplicativo.

Estado do Componente:
file: Armazena o arquivo da planilha selecionado pelo usuário.

showUserIDs: Um estado booleano para controlar se os IDs de usuário no gráfico devem ser exibidos.

rawData: Armazena os dados brutos recebidos do servidor após o upload da planilha.

Funções de Manipulação de Eventos:
handleFileChange: Atualiza o estado file quando um arquivo é selecionado.

handleUpload: Envia a planilha para o servidor, recebe os dados processados e exibe um gráfico usando a função createChart.

toggleShowUserIDs: Alterna o estado showUserIDs para mostrar ou ocultar os IDs de usuário no gráfico.

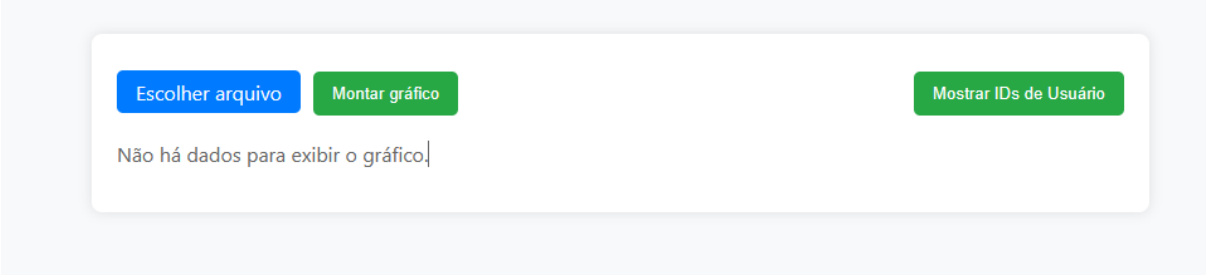
Função createChart:
Processa os dados brutos do servidor para formatá-los corretamente para o gráfico.
Cria um gráfico de barras usando a biblioteca Chart.js com três datasets: MRR (Receita Recorrente Mensal), Churn Rate (Taxa de Cancelamento) e User IDs (IDs de Usuário).
Utiliza o Chart.js para renderizar o gráfico no elemento <canvas>.

Hooks useRef e useEffect:
chartContainer e chartInstance são utilizados para manter uma referência ao elemento <canvas> e à instância do gráfico Chart.js, permitindo sua manipulação durante o ciclo de vida do componente.
Um efeito é utilizado para destruir a instância do gráfico quando o componente é desmontado.

Renderização do Componente:
Renderiza a interface de usuário com um botão para escolher e enviar uma planilha, um botão para alternar a exibição dos IDs de usuário e um elemento <canvas> para exibir o gráfico.
Se não houver dados a serem exibidos, uma mensagem informando a ausência de dados é exibida.

Em resumo, este componente React representa uma aplicação web simples que permite ao usuário fazer upload de planilhas, processa essas planilhas em um servidor, e exibe as métricas resultantes em um gráfico interativo.

Projeto em funcionamento:



Escolher arquivo | Montar gráfico | Mostrar IDs de Usuário