Gerador de Propostas

1. Visão Geral

Objetivo

O objetivo deste microsaas é fornecer uma ferramenta para prestadores de serviços criarem, gerenciarem e visualizarem propostas comerciais de maneira rápida e fácil. A solução oferece funcionalidades para criar propostas em formato digital, com campos personalizados como título, descrição, cliente, preço, entre outros. O sistema também permite a listagem, edição e exclusão de propostas.

Público-Alvo

Prestadores de serviços autônomos, pequenas e médias empresas que precisam gerar propostas comerciais de maneira eficiente e profissional.

2. Arquitetura do Sistema

Diagrama de Arquitetura

A arquitetura do sistema segue um modelo cliente-servidor, onde o frontend (cliente) se comunica com o backend (servidor) via API RESTful. O backend utiliza o Flask, enquanto o frontend é construído com React.

Frontend: React, Vite, Axios, TailwindCSS
Backend: Flask, SQLAlchemy, SQLite (ou PostgreSQL em produção)
Comunicação: REST API
Formato de Dados: JSON
Banco de Dados: SQLite (em desenvolvimento) ou PostgreSQL (em produção)

3. Requisitos

Requisitos Funcionais

Cadastro e Autenticação de Usuário:

O sistema deve permitir que usuários se cadastrem e façam login.
Autenticação baseada em JWT (JSON Web Token).

Gestão de Propostas:

O sistema deve permitir a criação, visualização, edição e exclusão de propostas.
As propostas devem ser armazenadas no banco de dados e associadas a um usuário.

Geração de PDF:

O sistema deve permitir o download de propostas em formato PDF.

Interface Web:

O sistema deve possuir uma interface web para que os usuários possam gerenciar suas propostas.
As propostas devem ser listadas em uma tabela, com opções para editar, excluir e baixar em PDF.

Requisitos Não Funcionais

Desempenho:

O sistema deve ser rápido e responsivo, garantindo tempos de resposta menores que 2 segundos em 95% das requisições.

Segurança:

A comunicação entre frontend e backend deve ser realizada por meio de HTTPS.
O sistema deve utilizar JWT para autenticação de usuários.

Usabilidade:

A interface deve ser intuitiva e responsiva, adaptando-se a diferentes tamanhos de tela (mobile e desktop).

Escalabilidade:

O sistema deve ser capaz de suportar um aumento gradual no número de usuários e propostas.

4. Design do Sistema

Frontend

Páginas:

Home: Página inicial com informações sobre o sistema.
Propostas: Página de listagem de propostas de um usuário, com opções de visualizar, editar, excluir e baixar as propostas em PDF.
Nova Proposta: Formulário para criação de uma nova proposta, com campos como título, cliente, descrição e preço.

Tecnologias Usadas:

React para criação da interface.
React Router para navegação entre páginas.
Axios para requisições HTTP.
TailwindCSS para estilização.

Backend

Roteamento e Controladores:

POST /proposta: Criar uma nova proposta.
GET /propostas/{usuario_id}: Listar todas as propostas de um usuário.
PUT /proposta/{id}: Atualizar uma proposta existente.
DELETE /proposta/{id}: Excluir uma proposta.
GET /baixar_pdf/{id}: Baixar a proposta em formato PDF.

Tecnologias Usadas:

Flask para construção da API REST.
SQLAlchemy para ORM e interação com o banco de dados.
SQLite ou PostgreSQL para persistência de dados.

5. Fluxo de Dados

1. Criação de Proposta

O usuário preenche um formulário no frontend com informações da proposta.
O frontend envia uma requisição POST para o backend com os dados.
O backend valida os dados, cria uma nova proposta e a armazena no banco de dados.
O backend responde com uma confirmação de criação bem-sucedida.

2. Listagem de Propostas

O frontend solicita as propostas do usuário por meio de uma requisição GET para o backend.
O backend retorna a lista de propostas associadas ao usuario_id.
O frontend exibe as propostas em uma tabela.

3. Edição de Proposta

O usuário seleciona uma proposta para editar e preenche o formulário com novos dados.
O frontend envia uma requisição PUT para o backend com os dados atualizados.
O backend valida os dados, atualiza a proposta e confirma a alteração.

4. Exclusão de Proposta

O usuário clica no botão "Excluir" ao lado de uma proposta.
O frontend envia uma requisição DELETE para o backend para excluir a proposta.
O backend remove a proposta do banco de dados e confirma a exclusão.

5. Geração e Download de PDF

O usuário seleciona a opção "Baixar PDF" para uma proposta.
O frontend envia uma requisição GET para o backend.
O backend gera um arquivo PDF da proposta e retorna-o como um blob.
O frontend inicia o download do arquivo PDF.

6. Testes

Testes Unitários

Backend:

Testes de validação para criação e edição de propostas.
Testes de autenticação e autorização de usuários.
Testes de geração de PDF.

Frontend:

Testes de formulário de criação de proposta (validação de campos obrigatórios).
Testes de interatividade na listagem de propostas.
Testes de chamadas API com Axios.
Testes de Integração

Backend + Frontend:

Testar a integração entre o frontend e o backend com dados reais.
Garantir que as requisições HTTP sejam feitas corretamente e que os dados sejam exibidos corretamente no frontend.

7. Deploy e Produção

Deploy do Backend

O backend será hospedado em plataformas como Heroku ou AWS.
O banco de dados será configurado para produção, possivelmente com o uso de PostgreSQL.
O backend estará acessível via HTTPS, com rotas protegidas por autenticação JWT.

Deploy do Frontend

O frontend será hospedado em plataformas como Vercel, Netlify ou Firebase Hosting.
A aplicação React será compilada e servida a partir de arquivos estáticos.

8. Conclusão

Este microsaas oferece uma solução simples, porém eficaz, para prestadores de serviços gerenciarem suas propostas comerciais. A arquitetura baseada em microserviços e a integração fluida entre frontend e backend proporcionam escalabilidade e flexibilidade ao sistema. O uso de Flask, React, e outras tecnologias modernas garante que o sistema seja de fácil manutenção e extensível.