Documentação de Desenvolvimento - Painel Administrativo

Visão Geral do Projeto

O sistema proposto tem como objetivo desenvolver um painel administrativo para cadastro e gerenciamento de clientes. O painel permitirá aos usuários inserirem dados de clientes, visualizar, editar e excluir esses dados, além de um sistema de autenticação e outras funcionalidades administrativas.

Requisitos do Sistema

- **Front-End:**
    - Utilizar **React** com **Vite**.
- **Back-End:**
    - Utilizar **Nest.js** com **TypeORM** e **PostgreSQL**.
    - Seguir o padrão de desenvolvimento da documentação oficial do **Nest.js**.
- **Tecnologias Gerais:**
    - **TypeScript**.
    - Versões mais recentes das tecnologias utilizadas.
    - Arquitetura baseada em tecnologias da **AWS**.
    - **Docker** e **Docker Compose** para rodar as aplicações.
    - Criar um **README** explicativo.
    - **Gravar vídeo** demonstrando a aplicação.

Diferenciais:

- **Front-End:**
    - Testes **end-to-end** (com **Cypress**).
- **Back-End:**
    - **Observabilidade** com **Prometheus** e **Grafana**.
    - Documentação da API com **Swagger**.
    - Testes **unitários** (com **Jest**).
- **Gerais:**
    - **Deploy** das aplicações.
    - Arquitetura com **RabbitMQ** ou **BullMQ** para mensageria e escalabilidade.

Estimativa de Tempo e Recursos

1. Tempo Necessário

O desenvolvimento do sistema será realizado em **uma sprint de 15 dias** (aproximadamente 2 semanas). Esse prazo é suficiente para a implementação de todas as funcionalidades necessárias, incluindo:

- Cadastro, visualização, atualização e exclusão de clientes.

- Implementação de API RESTful no back-end.
- Desenvolvimento de interface interativa e responsiva no front-end.
- Testes unitários e end-to-end.
- Documentação e entrega do projeto.

2. Número de Desenvolvedores

O projeto será desenvolvido por **2 programadores**:

- **1 programador backend sênior**.
- **1 programador frontend pleno**.

A equipe será responsável por todas as fases do projeto, desde o desenvolvimento até os testes e deploy.

3. Senioridade dos Desenvolvedores

- **Backend**: O desenvolvedor backend será **sênior**, com experiência em **NestJS**, **PostgreSQL**, **TypeORM** e boas práticas de desenvolvimento de APIs. Ele será responsável pela implementação da arquitetura, lógica de negócios e integração com o banco de dados, bem como a integração de tecnologias como **Prometheus**, **Grafana**, **RabbitMQ** e **Swagger**.
- **Frontend**: O desenvolvedor frontend será **pleno**, com experiência em **React**, **Vite**, **TypeScript** e **Styled Components**, além de habilidades em **Cypress** para testes end-to-end e **Zustand** para gerenciamento de estado. Ele será responsável pelo desenvolvimento da interface de usuário, garantindo que seja intuitiva, responsiva e modular.

Tecnologias Utilizadas

Back-End

O back-end do sistema será desenvolvido com as seguintes tecnologias:

- **NestJS**: Framework Node.js baseado em TypeScript, ideal para a criação de APIs escaláveis e de fácil manutenção.
- **TypeORM**: ORM (Object Relational Mapper) para integração com o banco de dados **PostgreSQL**.
- **PostgreSQL**: Banco de dados relacional para armazenamento de informações dos clientes.
- **Docker**: Para garantir um ambiente de desenvolvimento isolado e fácil de configurar.
- **Swagger**: Para documentar a API de forma interativa.
- **Jest**: Framework de testes unitários para garantir a qualidade do código.
- **Prometheus** e **Grafana**: Para monitoramento e visualização de métricas do sistema.
- **RabbitMQ**: Para mensageria, permitindo uma arquitetura escalável.

Front-End

O front-end será desenvolvido utilizando as seguintes tecnologias:

- **React**: Biblioteca JavaScript para construção da interface de usuário.
- **Vite**: Ferramenta de build e desenvolvimento super rápida para projetos React.
- **Styled Components**: Biblioteca para escrever CSS dentro do JavaScript, permitindo uma abordagem flexível para o estilo.

- **MUI Material**: Conjunto de componentes React baseados no Material Design.
- **Cypress**: Framework para testes end-to-end, garantindo a estabilidade da aplicação.
- **Zustand**: Biblioteca de gerenciamento de estado para React, simples e eficiente.
- **React Router DOM**: Para navegação entre páginas da aplicação.
- **Yup**: Biblioteca de validação de schema para formulários.

Arquitetura e Padrões de Desenvolvimento

Back-End

A arquitetura do back-end será baseada na **Clean Architecture**, garantindo que o sistema seja altamente modular e de fácil manutenção. O uso de tecnologias como **Prometheus**, **Grafana**, **OpenTelemetry** e **RabbitMQ** para observabilidade e mensageria permitirá que o sistema seja escalável e tenha um monitoramento efetivo.

Front-End

No front-end, a aplicação será construída com uma arquitetura de **componentes reutilizáveis** e será estruturada seguindo o padrão **SOLID**. Utilizaremos **Styled Components** para manter o CSS isolado e garantir que o design seja modular. A navegação entre as páginas será gerenciada pelo **React Router DOM**, e a internacionalização será feita com a biblioteca **i18n**.

Deploy e Escalabilidade

O sistema será containerizado com **Docker** e **Docker Compose**, permitindo fácil instalação e configuração no ambiente de produção. O **deploy** será realizado em um ambiente de cloud (AWS), garantindo escalabilidade e alta disponibilidade.

Conclusão

Este projeto será desenvolvido utilizando as mais recentes e avançadas tecnologias, garantindo a criação de uma aplicação moderna, eficiente e escalável. A equipe de 2 desenvolvedores seniores / pleno será capaz de entregar a solução completa dentro do prazo estipulado de 15 dias, com foco em qualidade de código, boas práticas de arquitetura e testes robustos.

Como Rodar as Aplicações

1. **Backend**:
    - Clone o repositório do backend.
    - Instale as dependências: `npm install`.
    - Configure as variáveis de ambiente (no arquivo `.env`).
    - Execute o backend: `npm run start:dev`.
2. **Frontend**:
    - Clone o repositório do frontend.

- Instale as dependências: `npm install`.
- Execute o frontend: `npm run dev`.

Vídeo de Demonstração

Um vídeo demonstrando o funcionamento completo das aplicações será gravado, incluindo o processo de cadastro de clientes, visualização, edição e exclusão.