Subsecretaria
de Tecnologia
da Informação

Documentação do Projeto: Sistema Biométrico Fullstack

1. Raiz do Projeto

Descrição:

Esta é a pasta raiz do projeto "sistema-biométrico". Ela contém a estrutura principal do sistema, dividida em diferentes módulos e arquivos de configuração.

Conteúdo Principal Observado:

- backend-node/: Diretório contendo a lógica do backend desenvolvida em Node.js.
- backend-python/: Diretório com a lógica do backend desenvolvida em Python, focada no processamento biométrico.
- database: Contém scripts e configurações relacionadas ao banco de dados PostgreSQL.
- frontend/: Diretório da aplicação frontend, responsável pela interface do usuário.
- README.md: Arquivo principal com a descrição geral do projeto, instruções de instalação e uso.
- package.json, package-lock.json: Arquivos de gerenciamento de dependências do Node.js para o projeto.
- .gitignore: Específica arquivos e pastas a serem ignorados pelo Git.

2. Backend Node.js - Raiz

Descrição:

Este diretório contém a aplicação backend principal desenvolvida em Node.js, utilizando o framework Express.js. É responsável por gerenciar as APIs REST, autenticação, lógica de negócios e comunicação com o banco de dados.

Conteúdo Principal Observado:

- src/: Pasta principal contendo o código-fonte da aplicação Node.js.
- config/: Pasta que contém o arquivo db.js responsável pelo pool de conexões com o banco de dados.

- controller/: Pasta que contém os controladores da lógica de negócio.
- middleware/: Pasta que contém arquivos com a lógica de autenticação e configuração do middleware do Multer..
- routes/: Pasta que contém os arquivos de rotas para navegação.
- .gitignore: Arquivo que define arquivos e pastas ignorados especificamente para o backend Node.js.
- package.json: Define as dependências e scripts do projeto Node.js.
- tsconfig.json: Configuração do TypeScript, indicando que o projeto utiliza TypeScript.

3. Backend Python - Raiz

Descrição:

Este diretório abriga a aplicação backend desenvolvida em Python. Dada a natureza do projeto, este backend é especializado no processamento de dados biométricos, comunicação com hardware de leitura biométrica e algoritmos de reconhecimento.

Conteúdo Principal Observado:

- app/: Pasta contendo o código fonte da aplicação Python.
- controller/: Pasta que contém os controladores da lógica de negócio.
- routes/: Pasta que contém os arquivos de rotas para navegação.
- .env: Arquivo de exemplo para variáveis de ambiente.
- .gitignore: Arquivos ignorados para o backend Python.
- server.py: Ponto de entrada principal da aplicação Python usando Flask.
- requirements.txt: Lista as dependências Python do projeto.

4. Banco de Dados - Schema Completo

Descrição:

Banco de dados PostgreSQL do sistema Biométrico. Ele define todas as tabelas, colunas, tipos de dados, chaves primárias, chaves estrangeiras e funções trigger.

Conteúdo Esperado (Exemplos de Tabelas):

- usuários: Tabela para armazenar informações dos usuários do sistema (administradores, gestores e quiosques).
- registros_ponto: Tabela para armazenar informações dos registros de ponto.
- secretarias: Tabela para armazenar informações das secretarias.
- unidade: Tabela para armazenar informações das unidades.
- férias: Tabela para informações das férias dos funcionários.

5. Frontend - Código Fonte (app)

Descrição:

A pasta app dentro de frontend é característica do App Router do Next.js e contém a estrutura principal da interface do usuário, incluindo layouts, páginas, componentes e lógica de cliente.

Conteúdo Principal Observado:

- (page)/: Diretório que agrupa diferentes seções ou páginas da aplicação. A nomenclatura com parênteses é uma convenção do Next.js para organização de rotas de grupo sem afetar a URL.
- auth/: Páginas relacionadas à autenticação (login, registro).
- dashboard/: Página principal do dashboard após o login, com informações da secretaria.
- funcionários/: Páginas para gerenciamento de funcionários.
- quiosque/: Página para efetuar o registro de ponto.
  registros/: Página para gerenciar registros de ponto.
- relatorios/: Página para gerenciar os relatórios dos funcionários.
- unidades/: Página para gerenciar as unidades da secretaria.
- login/: Página responsável pela lógica de autenticação do usuário.
- components/: Componentes React reutilizáveis, como botões, modais, tabelas e formulários.
- ui/: Componentes básicos de UI, possivelmente provenientes de uma biblioteca como shadcn/ui.
- contexto/: Contextos React para gerenciamento de estado global ou compartilhado.
- globals/: Estilos globais da aplicação.
- hooks/: Custom Hooks React para encapsular lógica reutilizável.

- layout.tsx/: Arquivo de layout principal da aplicação Next.js, definindo a estrutura comum a todas as páginas.
- page.tsx/: Arquivo da página inicial da aplicação (rota /).
  lib/api/: Funções para realizar chamadas às APIs backend.
- utils/: Funções utilitárias para o frontend.