Descrição Geral do Projeto

Este projeto consiste em uma **aplicação web de cadastro e login de usuários** construída com **Flask** (Python) e **SQLite** para gerenciamento de dados. O foco principal está na segurança e na validação de dados de entrada, como o uso de **criptografia de senha** (werkzeug.security) e a criptografia opcional de dados sensíveis (CPF) usando a biblioteca **Fernet** (cryptography).

A arquitetura se divide em:

1. **Backend (app.py):** Lida com o roteamento, lógica de negócios, conexão com o banco de dados e gerenciamento de sessões. Implementa as funcionalidades de criação de banco de dados, registro (/cadastrar), login (/login), exibição de página principal restrita (/home) e logout. Utiliza flash para exibir mensagens de status ao usuário.
2. **Frontend (HTML, CSS, JS):**

- **Páginas HTML (cadastro.html, login.html, confirmacao.html, home.html):** Estrutura a interface do usuário.
- **CSS (style.css):** Estiliza a aplicação, aplicando um tema em tons de azul.
- **JavaScript (script.js):** Implementa **máscaras** (CPF e telefone) e **validações** robustas no lado do cliente (nome, CPF, telefone, força da senha e confirmação de senha) antes do envio ao servidor, melhorando a experiência do usuário.

Requisitos Funcionais

[RF001] Cadastro de Usuário: O sistema deve permitir que um novo usuário insira nome, CPF, email, telefone, data de nascimento e uma senha.

[RF002] Validação de Entrada (Frontend): O sistema deve validar em tempo real (client-side) o formato e integridade dos dados, como:

- Aplicação de máscara para **CPF** (formato 000.000.000-00).
- Aplicação de máscara para **Telefone** formato (00) 00000-0000 ou (00) 0000-0000).
- **Validação do Nome** (somente letras e espaços, sem números).
- **Validação de Força da Senha** (mínimo de 8 caracteres, com maiúsculas, minúsculas, números e especiais).
- **Confirmação de Senha** (verificar se as senhas digitadas são iguais).
- **Validação final do CPF** (algoritmo de dígitos verificadores) no submit.

[RF003] Persistência de Dados: O sistema deve armazenar os dados do usuário (nome, email, telefone, CPF, data de nascimento, senha hash) no banco de dados SQLite.

[RF004] Verificação de Email Único: O sistema não deve permitir o cadastro de emails duplicados.

[RF005] Criptografia de Senha: A senha do usuário deve ser armazenada como um *hash* seguro (usando generate_password_hash).

[RF006] Login de Usuário: O sistema deve permitir que um usuário existente faça login com email e senha.

[RF007] Autenticação: O sistema deve verificar se a senha fornecida no login corresponde ao hash armazenado (check_password_hash).

[RF 008] Gerenciamento de Sessão: O sistema deve criar uma sessão após o login para manter o estado autenticado do usuário (session["user_id"]).

[RF009] Proteção de Rota: A página de *Home* (/home) deve ser acessível apenas para usuários logados.

[RF010] Logout: O sistema deve permitir que o usuário encerre a sessão (/logout).

[RF011] Confirmação de Cadastro: O sistema deve exibir uma página de confirmação após o registro bem-sucedido.

Requisitos Não Funcionais

[RNF001] Segurança (Criptografia de Senha): As senhas devem ser armazenadas de forma irrecuperável (hash).

[RNF002] Segurança (Criptografia de CPF): O CPF deve ser criptografado antes do armazenamento, se uma chave Fernet válida for fornecida no ambiente (FERNET_KEY).

[RNF003] Usabilidade (UX/UI): A interface deve ser clara e usar um tema consistente (cores e fontes definidas em style.css).

[RNF004] Desempenho (Validação Client-Side): As validações devem ser executadas no navegador (JS) para fornecer feedback imediato ao usuário e reduzir a carga do servidor.

[RNF005] Confiabilidade: O sistema deve usar um mecanismo de *flashing* (flash) para informar o usuário sobre erros de login ou cadastro (ex: email já cadastrado).

[RNF006] Manutenibilidade: O código deve ser organizado com separação de lógica (Backend em Python, Frontend em HTML/CSS/JS)

Diagrama de Identidade e relacionamento:

usuario	
PK	usuario_id int NOT NULL
	nome varchar(50) NOT NULL cpf char(14) NOT NULL email varchar(100) NOT NULL telefone varchar(15) NOT NULL data_nascimento date NOT NULL senha_hash varchar(255) NOT NULL

Diagrama de Caso De Uso: