



**CAMPUS
UNIVERSITÁRIO DE
SAQUAREMA**

**UNIVERSIDADE DE VASSOURAS
CAMPUS SAQUAREMA
CURSO DE ENGENHARIA DE SOFTWARE**

SISTEMA DE CADASTRO WEB

**MARCELLA LIMA NUNES AZEVEDO
JOÃO VICTOR DE MORAES DA CRUZ**

**ENGENHARIA DE REQUISITOS E ANÁLISE DE
SISTEMAS
GIOLIANO BERTONI**

**SAQUAREMA – RJ
2025**

Projeto XPRESS

Sobre o projeto

O **XPRESS-site** consiste em uma **plataforma web voltada à intermediação de serviços digitais** e gerenciamento de cadastros de usuários, com foco em **experiência do usuário, acessibilidade, segurança de dados e escalabilidade futura**.

O sistema integra uma camada de Frontend desenvolvida com HTML5, CSS3 e JavaScript, e uma camada de Backend implementada em Python com o framework Flask, conectado a um banco de dados **SQLite3**, de modo a oferecer autenticação segura, persistência de dados e uma interface moderna e responsiva.

Objetivo do Projeto

O projeto XPRESS-site tem como objetivo:

- Criar um **portal web para autenticação e cadastro de usuários**;
- Garantir uma **interface moderna, intuitiva e responsiva**;
- Implementar um **backend seguro e escalável** para manipulação de dados;
- Demonstrar domínio nas tecnologias **HTML5, CSS3, JavaScript, Python (Flask) e SQLite3**;
- Prover base sólida para futuras expansões, como integração com APIs externas e novos módulos.

Arquitetura do Sistema

O sistema é composto por três camadas principais:

- **Frontend:**
Responsável pela interface e interação do usuário. Desenvolvido com HTML5, CSS3 e JavaScript.
- **Backend (Flask):**
Fornece rotas HTTP, autenticação e integração com o banco de dados.
- **Banco de Dados (SQLite3):**
Responsável pelo armazenamento de informações de usuários e demais dados persistentes.

A comunicação entre o Frontend e o Backend ocorre por meio de requisições HTTP, onde os formulários de login e cadastro enviam dados ao servidor Flask, que realiza a validação e o armazenamento no banco SQLite3.

Diagrama de Arquitetura

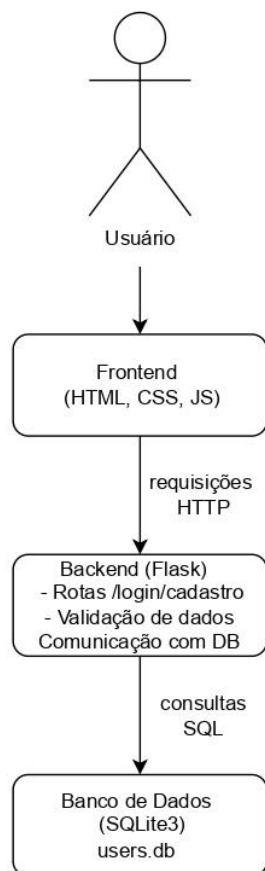


Diagrama de Caso de Uso

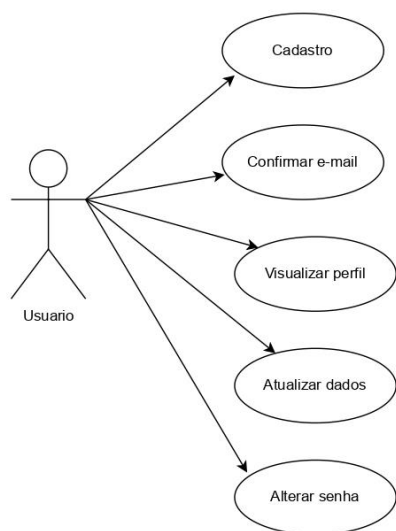


Diagrama de Entidade-Relacionamento (DER)

Usuário			
int	id_usuario	PK	Primary Key
string	nome		
string	cpf		
string	email		
string	celular		
string	endereço		
string	senha		

1. Requisitos Funcionais

1.1 [RF001] Gerenciar cadastro de usuário

O sistema deve permitir que o usuário crie uma conta informando seus dados pessoais — nome completo, CPF, e-mail, celular, endereço, nome de login e senha —, com validação dos campos obrigatórios.

1.2 [RF002] Autenticar usuário (login)

O sistema deve permitir que o usuário acesse sua conta informando e-mail e senha, validando as credenciais no banco de dados antes de conceder o acesso.

1.3 [RF003] Selecionar plano de serviço

O sistema deve permitir que o usuário escolha um plano (Básico, Plus ou Premium) e exibir automaticamente o plano selecionado no formulário de cadastro.

1.4 [RF004] Exibir mensagens de feedback

O sistema deve exibir mensagens dinâmicas de alerta, erro ou sucesso durante os processos de cadastro e login, sem necessidade de recarregar a página.

1.5 [RF005] Persistir e atualizar informações de usuários

O sistema deve registrar, consultar e atualizar informações de usuários no banco de dados SQLite3, garantindo integridade e consistência.

2. Requisitos Não Funcionais

2.1 [RNF001] Responsividade da interface

O sistema deve adaptar automaticamente o layout para diferentes dispositivos (desktop, tablet e smartphone), mantendo legibilidade e boa experiência de navegação.

2.2 [RNF002] Segurança e proteção de dados

O sistema deve proteger dados sensíveis, armazenando senhas com hash criptográfico (ex: biblioteca Werkzeug) e impedindo o envio de informações sem consentimento do usuário.

2.3 [RNF003] Desempenho e tempo de carregamento

O sistema deve carregar suas páginas e scripts em até **3 segundos** em conexões de internet padrão, garantindo fluidez na navegação.

2.4 [RNF004] Usabilidade e design intuitivo

A interface deve ser clara, com campos bem distribuídos e rótulos informativos, garantindo fácil navegação e interação para usuários de diferentes perfis.

2.5 [RNF005] Manutenibilidade e escalabilidade do código

O código deve ser modular e bem estruturado (separando HTML, CSS, JS e Flask), permitindo fácil manutenção, depuração e expansão futura da plataforma

Tecnologias Utilizadas

Camada	Tecnologia	Versão Recomendada	Descrição
Frontend	HTML5	HTML5	Estrutura das páginas
Frontend	CSS3	CSS3	Estilização e layout responsivo
Frontend	JavaScript (ES6+)	JavaScript (ES6+)	Validação e interatividade
Backend	Python	3.11+	Lógica do servidor
Framework	Flask	3.0+	Framework web para backend
Banco de Dados	SQLite3	3.45+	Banco de dados leve e local

Dependências Necessárias

Python 3.11+

Flask 3.0+

Werkzeug (hashing de senhas, geração de tokens, seguros, proteção contra ataques comuns)

Jinja2 (recebe um modelo HTML, recebe dados do Python, combina tudo e devolve uma página pronta para o navegador)

Itsdangerous (biblioteca de segurança usada pelo Flask para gerar e validar tokens seguros).

MarkupSafe (biblioteca de segurança usada pelo Flask (e pelo Jinja2) para impedir ataques XSS)

SQLite3 (já vem no Python)

Como Rodar a Aplicação

Pré-requisitos

- Python 3.11 ou superior
- pip (gerenciador de pacotes do Python)
- Navegador web atualizado

Funcionalidades Principais

- **Cadastro de Usuários** (nome, CPF, e-mail, celular, endereço, login e senha)
- **Login e autenticação** com validação no backend
- **Integração com SQLite3** para persistência de dados
- **Design responsivo** e estilização customizada com CSS
- **Camada de segurança** com hash de senhas

Possíveis Extensões Futuras

- Integração com APIs de pagamento;
- Implementação de painel administrativo;
- Sistema de recuperação de senha;
- Integração com APIs externas para envio de e-mails automáticos.

Conclusão

O XPRESS-site cumpre seu papel como plataforma web de autenticação e cadastro de usuários, aliando boa experiência de uso, segurança de dados e escalabilidade. A documentação apresentada serve de base para futuras expansões e manutenção do sistema.

