



**CAMPUS  
UNIVERSITÁRIO DE  
SAQUAREMA**

**UNIVERSIDADE DE VASSOURAS**

**CAMPUS UNIVERSITÁRIO DE SAQUAREMA**

**ENGENHARIA DE SOFTWARE 2025.2**

**ENGENHARIA DE REQUISITOS E ANÁLISE DE SISTEMAS**

**DOCUMENTAÇÃO DE REQUISITOS SAQUACARE**

Prof. Gioliano Bertoni

Alunas: Larissa Simas, Nayara Emelly e Emenuely de Oliveira.

Saquarema - RJ

2025

## 1. INTRODUÇÃO

O **SaquaCare** representa uma iniciativa inovadora na área da saúde pública do município de Saquarema. Trata-se de uma plataforma web integrada desenvolvida para modernizar, simplificar e otimizar o acesso da população aos serviços de saúde pública, promovendo maior eficiência operacional e melhor experiência do usuário no Sistema Único de Saúde (SUS) local.

A implementação do **SaquaCare** surge como resposta às necessidades identificadas no atual sistema de saúde municipal, onde se observam desafios como:

- Dificuldade no agendamento de consultas.
- Excessivo tempo de espera nas unidades de saúde.
- Fragmentação de informação entre setores.

O **SaquaCare** visa enfrentar esses desafios por meio da tecnologia, criando um canal direto unificado que conecta cidadãos e profissionais de saúde.

## 2. REQUISITOS FUNCIONAIS

**[RF001]** Cadastro de usuário:

**Prioridade:** Alta

**Descrição:** O sistema deve permitir o cadastro de novos usuários, coletando as seguintes informações obrigatórias.

- Nome Completo
- CPF
- Número do Cartão Nacional de Saúde (SUS)
- Senha

**Regras de validação e confirmação:**

1. **Validação do CPF:** O sistema deve validar o formato e os dígitos verificados do CPF informado.
2. **Validação do Cartão do SUS:** O sistema deve validar o formato do número do Cartão Nacional de Saúde, onde tradicionalmente possui 15 dígitos.
3. **Confirmação de senha:** O sistema deve verificar se o campo “Senha” e “Confirmação de Senha” coincidem.

### 3. REQUISITOS NÃO FUNCIONAIS

**[RNF001]** Banco de dados SQLite3:

**Prioridade:** Alta

Inicialmente o sistema deve ser capaz de suportar operações com o banco de dados SQLite3.

**[RNF002]** Linguagem de marcação HTML5:

**Prioridade:** Alta

O sistema de cadastro deve ser desenvolvido com a linguagem de marcação HTML5, e ser compatível com os principais navegadores.

**[RNF003]** Linguagem de estilo CSS:

**Prioridade:** Alta

Os arquivos CSS devem ser desenvolvidos seguindo princípios de código limpo e organização que facilitem a manutenção.

**[RNF004]:** JavaScript

**Prioridade:** Alta

Deve ser utilizado JavaScript para tornar a página mais dinâmica e interativa.

**[RNF005]:** Python + Flask

**Prioridade:** Alta

Foi utilizado Python como linguagem de programação, juntamente com o Flask que é um microframework de desenvolvimento web escrito em Python, para criar a API de integração dos dados de cadastro com o banco de dados SQLite.

**[RNF006]** Interface Intuitiva:

**Prioridade:** Alta

Fácil uso para todos os públicos.

**[RNF007]** Capacidade de usuários:

**Prioridade:** Alta

Suporte a múltiplos usuários simultâneos.

**[RNF008]** Tempo de resposta:

**Prioridade:** Alta

Carregamento rápidos das páginas.

**[RNF009]** Disponibilidade:

**Prioridade:** Alta

O sistema deve ficar 99% do tempo disponível.

**[RFN0010]** Backup de dados:  
**Prioridade:** Alta  
Proteção contra perda de dados.

**[RFN006]** Proteção de dados:  
**Prioridade:** Alta  
Criptografia dos dados sensíveis, conforme LGPD.

**[RNF007]** Proteção contra ataques:  
**Prioridade:** Alta  
Segurança contra ameaças.

**[RNF008]** Criptografia de senha:  
**Prioridade:** Alta  
As senhas dos usuários são criptografadas antes do armazenamento no banco de dados utilizando a biblioteca Werkzeug do Flask, que implementa o algoritmo PBKDF2 com SHA-256. Garantindo que mesmo em caso de acesso não autorizado ao banco, as senhas permaneçam protegidas.

## 4. DIAGRAMA DE CASO DE USO

### 1. Ator principal Paciente

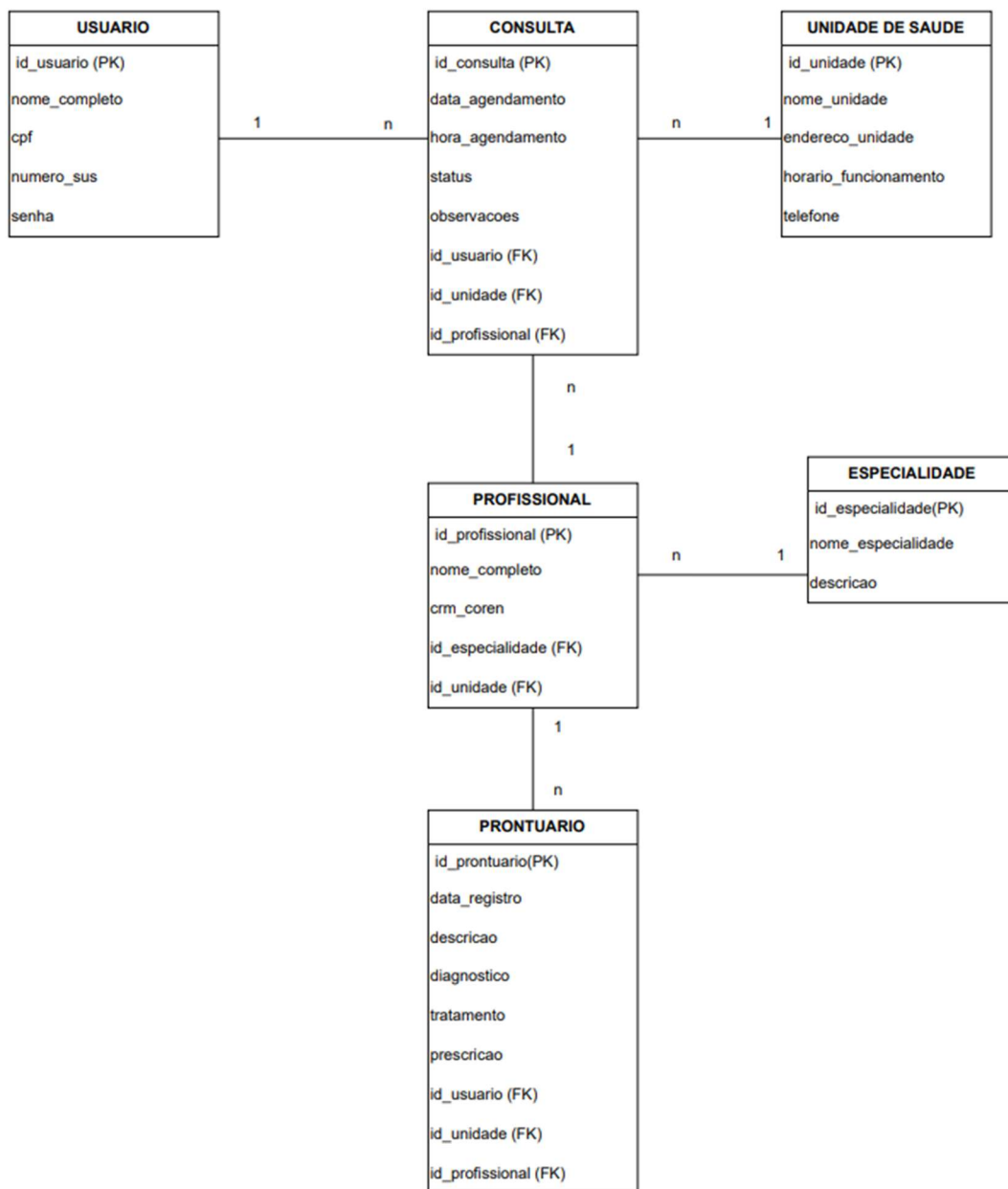
O paciente é o usuário final que interage com o sistema para agendar e gerenciar suas consultas. Suas funcionalidades incluem:

**Cadastrar-se:** Realizar o primeiro acesso ao sistema, fornecendo dados pessoais, o paciente precisa informar os dados como nome, CPF, número do SUS e senha. Ou seja, o usuário (Paciente) executa o caso de uso cadastrar-se.

#### DIAGRAMA CASO DE USO - CADASTRO DE USUÁRIO SAQUACARE



## 5. DIAGRAMA DE ENTIDADE E RELACIONAMENTO



## 6. DOCUMENTAÇÃO TÉCNICA: IMPLEMENTAÇÃO FLASK + SQLITE PARA CADASTRO DE USUÁRIOS

### 6.1 Arquitetura do Sistema

O sistema SaquaCare foi desenvolvido seguindo uma arquitetura cliente-servidor, onde:

- **Frontend:** Interface web desenvolvida em HTML5, CSS3 e JavaScript
- **Backend:** API REST desenvolvida com Flask (Python)
- **Banco de Dados:** SQLite para persistência de dados
- **Comunicação:** JSON (JavaScript Object Notation) via requisições HTTP

#### 6.1.1 Backend - Flask Framework

- **Flask 2.3.3:** Microframework web para Python
- **Flask-SQLAlchemy 3.0.5:** ORM (Object-Relational Mapping) para integração com banco de dados.
- **Flask-CORS:** Habilitação de Cross-Origin Resource Sharing.
- **Werkzeug 2.3.7:** Biblioteca para segurança (hash de senhas).

#### 6.1.2 Banco de Dados – SQLite

- **SQLite3:** Banco de dados relacional embarcado
- **Arquivo:** database.db na pasta backend/database/
- **Vantagens:** Leve, não requer servidor dedicado, ideal para protótipos

#### 6.1.3 Frontend

- **HTML5:** Estrutura semântica do formulário de cadastro
- **CSS3:** Estilização responsiva e design moderno
- **JavaScript:** Validações em tempo real e integração com API

### 6.2 Fluxo do cadastro

- Usuário preenche formulário no frontend
- JavaScript valida dados localmente (CPF, senhas)
- Dados são enviados via POST para /api/usuários.

- Flask valida unicidade de CPF e número SUS
- Senha é criptografada antes do armazenamento
- Usuário é salvo no banco SQLite
- Frontend recebe confirmação e exibe modal de sucesso.