INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA
DIRETORIA DE DESENVOLVIMENTO DE ENSINO
COORDENAÇÃO DO CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

IGOR GABRIEL DOS SANTOS BARBOSA SANTANA
JOSÉ DAVI MIGUEL DA SILVA

RELATÓRIO TÉCNICO
Projeto Integrador em Sistemas para Internet (PISI): SGA UBS

GUARABIRA - PB
2026

IGOR GABRIEL DOS SANTOS BARBOSA SANTANA
JOSÉ DAVI MIGUEL DA SILVA

RELATÓRIO TÉCNICO

Projeto Integrador em Sistemas para Internet (PISI): SGA UBS

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Sistemas para Internet, do Instituto Federal da Paraíba – Campus Guarabira, em cumprimento às exigências parciais para a obtenção do título Tecnólogo em Sistemas para Internet.

Orientador(a): Rodrigo Leone Alves

GUARABIRA - PB
2026

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IFPB - GUARABIRA

S232r Santana, Igor Gabriel dos Santos Barbosa
Relatório técnico: Projeto Integrador em Sistemas para Internet (PISI): SGA UBS / Igor Gabriel dos Santos Barbosa Santana; José Davi Miguel da Silva.- Guarabira, 2026.
51f.; il.; color.

Trabalho de Conclusão de Curso (Tecnólogo em Sistemas para Internet). – Instituto Federal da Paraíba, Campus Guarabira. 2026.

"Orientação: Prof. Dr. Rodrigo Leone Alves"

Referências.

1.Sistema para internet 2. Sistema de Agendamento. 3. Unidade Básica de Saúde. 4. Sistema de Informação em Saúde. 5. Desenvolvimento Web. l. I. Título.

CDU 004.4(0.067)

Elaborada por Ana Carine da Costa Gonçalves - CRB 000676

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA

ATA 26/2026 - CCSTSI/DDE/DG/GB/REITORIA/IFPB

ATA DE APRESENTAÇÃO E DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

CCS de Tecnologia em Sistemas para Internet

Aos 06 de agosto de 2026, às 10:00, no Laboratório de Informática, reuniram-se os membros da banca avaliadora, Rodrigo Leone Alves (Orientador), Tannissa Luanna Cardoso de Araujo (Examinador Interno), Daniel Marques Targino Guimarães (Examinador Interno), para avaliarem a apresentação do *Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet* (Relatório Final do Projeto Integrador em Sistemas para Internet - PISI) do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), *campus* Guarabira, desenvolvido pelo(a) aluno(a) **José Davi Miguel da Silva, matrícula de nº 202313810018** intitulado "SGA UBS", protocolado para apresentação de acordo com os requisitos expostos no Projeto Pedagógico de Curso de Tecnologia em Sistemas para Internet. Após a apresentação, a banca apresentou, por unanimidade, pareceres a favor da aprovação do trabalho. Desta forma, o Trabalho de Conclusão de Curso foi **aprovado** e definiu-se a **nota final 91 (Noventa e Um)**.

Nada mais havendo a tratar, às 16:45, encerraram-se os trabalhos, determinando a lavratura desta ata, que, após lida e considerada conforme, será assinada pelos presentes. Eu, Rodrigo Leone Alves, lavrei a presente ata.

Guarabira/PB, em 10 de agosto de 2026.

Documento assinado eletronicamente por:

- **Rodrigo Leone Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 12:33:41.
- **Tannissa Luanna Cardoso de Araujo**, **PEDAGOGO-AREA**, em 10/08/2026 12:49:51.
- **Daniel Marques Targino Guimaraes**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 10/08/2026 22:32:17.

Este documento foi emitido pelo SUAP em 10/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 919935
Código de Autenticação: d91e11ec1e

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA

ATA 29/2026 - CCSTSI/DDE/DG/GB/REITORIA/IFPB

ATA DE APRESENTAÇÃO E DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

CCS de Tecnologia em Sistemas para Internet

Aos 06 de agosto de 2026, às 10:00, no Laboratório de Informática, reuniram-se os membros da banca avaliadora, Rodrigo Leone Alves (Orientador), Tannissa Luanna Cardoso de Araujo (Examinador Interno), Daniel Marques Targino Guimarães (Examinador Interno), para avaliarem a apresentação do *Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet* (Relatório Final do Projeto Integrador em Sistemas para Internet - PISI) do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), *campus* Guarabira, desenvolvido pelo(a) aluno(a) **Igor Gabriel dos Santos Barbosa Santana, matrícula de nº 202313810013** intitulado "SGA UBS", protocolado para apresentação de acordo com os requisitos expostos no Projeto Pedagógico de Curso de Tecnologia em Sistemas para Internet. Após a apresentação, a banca apresentou, por unanimidade, pareceres a favor da aprovação do trabalho. Desta forma, o Trabalho de Conclusão de Curso foi **aprovado** e definiu-se a **nota final 91 (Noventa e Um)**.

Nada mais havendo a tratar, às 16:45, encerraram-se os trabalhos, determinando a lavratura desta ata, que, após lida e considerada conforme, será assinada pelos presentes. Eu, Rodrigo Leone Alves, lavrei a presente ata.

Guarabira/PB, em 10 de agosto de 2026.

Documento assinado eletronicamente por:

- **Rodrigo Leone Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 12:41:23.
- **Tannissa Luanna Cardoso de Araujo**, **PEDAGOGO-AREA**, em 10/08/2026 12:50:24.
- **Daniel Marques Targino Guimaraes**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 10/08/2026 22:26:14.

Este documento foi emitido pelo SUAP em 10/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 919943
Código de Autenticação: 695f0d0e87

Por Igor Gabriel dos Santos Barbosa Santana

A Deus, toda honra e glória. À minha mãe, por seu apoio incondicional, e à minha família, o meu amor e gratidão. e aos meus saudosos pai e avô, cuja memória, ensinamentos e sacrifícios em vida continuam a guiar cada um dos meus passos.
Dedico!

Por José Davi Miguel da Silva

A Deus, por ser minha força e esperança. À minha mãe e irmã por todo amor, apoio e os sacrifícios realizados. E a toda minha família, pelo suporte ao longo desta caminhada. Dedico!

AGRADECIMENTOS

Por José Davi Miguel da Silva

Agradeço primeiramente a Deus, pela força e paciência concedidas durante todo este ciclo. À minha mãe, pelo apoio, pelo amor incondicional e por sempre me dar suporte. À minha irmã, pelo carinho e auxílio. A toda a minha família — avós, tios e primos —, que cuidaram de mim.

Aos professores, cujo conhecimento compartilhado foi fundamental para o meu crescimento acadêmico e pessoal. Aos amigos e colegas de turma, pelas conversas e pelo apoio que tornaram este caminho muito mais agradável.

Por fim, a todos que, direta ou indiretamente, trilharam este caminho comigo e contribuíram para esta vitória, registro minha profunda e sincera gratidão.

Por Igor Gabriel dos Santos Barbosa Santana

Deus foi minha força e sustento todos os dias, diante de tantas dificuldades, perdas e lutos, Ele não permitiu que eu desistisse, por isto, sou imensamente grato a Deus! À minha família, dedico esta conquista, agradeço cada um por todo o apoio, sustento e motivação até chegar à este momento de glória.

Aos professores, agradeço de todo o meu coração por todo o carinho e paciência ao compartilhar seus valiosos conhecimentos com tamanha precisão e preocupação com cada aluno. Aos meus cordiais amigos e colegas que fiz durante este curso, meus sinceros agradecimentos a cada um de vocês, esta jornada se tornou muito mais suave com a presença de vocês, quero que saibam que foi uma honra inestimável.

A todo o corpo docente do IFPB Campus Guarabira, muito obrigado a cada um dos senhores por tudo que proporcionaram às nossas vidas, a partir deste momento, seguiremos nossas vidas levando todo o conhecimento adquirido durante todo o curso, os senhores foram essenciais para nossa formação acadêmica e profissional!

"Supere seus limites. Bem aqui. Agora mesmo."

Yami Sukehiro, Black Clover

RESUMO

As Unidades Básicas de Saúde enfrentam desafios relacionados à organização dos atendimentos, especialmente devido ao uso de processos manuais para marcação de consultas e controle de agendas. Nesse contexto, este trabalho apresenta o desenvolvimento do Sistema de Gestão de Agendamentos para Unidades de Saúde, uma aplicação web destinada a melhorar a administração dos serviços de atendimento e facilitar o acesso dos pacientes às consultas. A solução foi desenvolvida utilizando as tecnologias React, Flask e PostgreSQL, seguindo uma arquitetura baseada em API REST. O sistema oferece funcionalidades para cadastro de pacientes e profissionais, gerenciamento de horários, realização e cancelamento de agendamentos, consulta ao histórico de atendimentos e acompanhamento de métricas gerenciais. Também foram implementadas regras de priorização para filas de espera e mecanismos de segurança baseados em autenticação JWT e armazenamento seguro de senhas. Como resultado, foi obtida uma plataforma capaz de reduzir inconsistências nos agendamentos, aumentar a eficiência dos processos administrativos e proporcionar uma experiência mais organizada para pacientes e profissionais de saúde. O projeto demonstra a aplicação integrada de conhecimentos adquiridos ao longo do curso e apresenta potencial para futuras evoluções e utilização em cenários reais.

Palavras-chave: Sistema de Agendamento; Unidade Básica de Saúde; Atenção Primária à Saúde; Sistema de Informação em Saúde; Desenvolvimento Web.

ABSTRACT

The Basic Health Units face challenges related to the organization of care, especially due to the use of manual processes for appointment scheduling and schedule control. In this context, this work presents the development of the Scheduling Management System for Health Units, a web application aimed at improving the administration of care services and facilitating patients' access to consultations. The solution was developed using React, Flask and PostgreSQL, following an architecture based on REST API. The system offers features for patient and professional registration, schedule management, scheduling and cancellation, consultation of service history and monitoring of management metrics. Prioritization rules for queues and security mechanisms based on JWT authentication and secure password storage have also been implemented. As a result, a platform was obtained capable of reducing scheduling inconsistencies, increasing the efficiency of administrative processes and providing a more organized experience for patients and healthcare professionals. The project demonstrates the integrated application of knowledge acquired throughout the course and presents potential for future developments and use in real scenarios.

Keywords: Appointment Scheduling System; Primary Health Care; Basic Health Unit; Health Information System; Web Development.

LISTA DE ILUSTRAÇÕES

GRÁFICOS

FIGURAS

TABELA

LISTA DE ABREVIATURAS E SIGLAS

APS — Atenção Primária à Saúde

BVSMS — Biblioteca Virtual em Saúde do Ministério da Saúde

CPF — Cadastro de Pessoas Físicas

ESF — Estratégia Saúde da Família

MVP — Minimum Viable Product

OMS — Organização Mundial da Saúde

PNS — Pesquisa Nacional de Saúde

SGA UBS — Sistema de Gestão de Agendamentos para Unidades Básicas de Saúde

SIS — Sistema de Informação em Saúde

SUS — Sistema Único de Saúde

UBS — Unidade Básica de Saúde

SUMÁRIO

1 INTRODUÇÃO

O Sistema Único de Saúde (SUS) consolida-se como o maior sistema público e gratuito do mundo, sendo a principal referência de cuidado para a maioria da população brasileira. Dados recentes indicam que 84% da população depende exclusivamente do SUS, e que 71,1% dos cidadãos procuram os estabelecimentos públicos quando apresentam algum problema de saúde (AGÊNCIA BRASIL, 2025; BVSMS, 2025). Dentro dessa rede, as Unidades Básicas de Saúde (UBS) desempenham um papel fundamental, sendo apontadas por 47,9% da população como a principal porta de entrada para o sistema (BVSMS, 2025).

O governo federal tem investido na modernização da atenção primária, elevando o orçamento da área para R$ 54,1 bilhões em 2024 e ampliando a conectividade das UBS para 94,6% (MINISTÉRIO DA SAÚDE, 2025). No entanto, a digitalização da infraestrutura ainda não se traduziu em melhorias significativas na gestão do acesso aos serviços, escancarando um dos principais gargalos do sistema: o agendamento de consultas e procedimentos.

Esse gargalo se reflete em números preocupantes: o tempo de espera é apontado pelo Ministério da Saúde como um dos principais entraves históricos, com dados revelando que 32,7% dos usuários que não conseguiram atendimento alegaram não ter conseguido vaga ou pegar senha (BVSMS, 2025; AGÊNCIA BRASIL, 2025). Além disso, o Censo Nacional das UBS de 2024 identificou que 60,1% das unidades necessitam de reformas estruturais, o que, somado à dificuldade de agendamento, compromete a resolutividade da atenção básica, sobrecarregando serviços especializados e as emergências hospitalares (MINISTÉRIO DA SAÚDE, 2025).

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma plataforma digital para facilitar o agendamento de consultas nas Unidades Básicas de Saúde (UBS), com o objetivo de:

Digitalizar e organizar o processo de marcação de consultas, reduzindo filas, faltas e retrabalho administrativo; Garantir integridade e consistência dos dados por meio de regras de negócio implementadas no backend, como validação de conflitos de horário, antecedência mínima para cancelamento e priorização na fila de espera; Servir como prova de conceito aplicável a unidades de saúde reais ou simuladas, demonstrando maturidade técnica em desenvolvimento de software, abrangendo modelagem de banco de dados, autenticação, regras de negócio, documentação e testes.

1.1.2 Objetivos Específicos

São estabelecidos como objetivos específicos do desenvolvimento:

1. Organizar e estruturar um banco de dados centralizado que armazene de forma segura e eficiente as informações de pacientes, profissionais, administradores, horários de consulta e a lista de espera, garantindo que os dados estejam corretos e de fácil acesso.

2. Criar um sistema de segurança robusto que proteja as informações dos usuários através de login autenticado (CPF e senha protegida), controlando o que cada tipo de usuário pode acessar.

3. Desenvolver funcionalidades essenciais, como: cadastro e login de pacientes, gerenciamento de profissionais e agendas, consulta de horários por especialidade, criação e cancelamento de agendamentos (respeitando prazos), acesso ao histórico de consultas e relatórios de gestão para monitorar o atendimento.

4. Implementar mecanismos que impeçam que dois pacientes marquem o mesmo horário simultaneamente.

5. Criar uma fila de espera inteligente que organiza o atendimento automaticamente, priorizando pessoas com mais de 60 anos, gestantes e pessoas com deficiência.

6. Realizar testes automáticos para garantir que o sistema funcione corretamente, verificando, por exemplo, se agendamentos duplicados são evitados e se cancelamentos são processados como esperado.

7. Preparar o sistema para ser facilmente instalado e executado, acompanhado de instruções claras para utilização.

1.2 Justificativa

Nesse contexto, este trabalho justifica-se pela necessidade de oferecer uma solução tecnológica que otimize o fluxo de atendimento, reduza filas de espera e melhore a experiência do usuário no SUS, contribuindo para a modernização da gestão da atenção primária à saúde. Assim, o presente projeto tem como objetivo geral desenvolver um sistema de gestão de agendamentos para Unidades Básicas de Saúde, que permita o agendamento online de consultas e procedimentos, a gestão de filas de espera e o acompanhamento do histórico de atendimentos dos pacientes.

2 REFERENCIAL TEÓRICO

2.1 O Sistema Único de Saúde (SUS) e a Atenção Primária

Instituído pela Constituição Federal de 1988 e regulamentado pela Lei nº 8.080/1990, o Sistema Único de Saúde (SUS) estabelece a saúde como um direito universal e um dever do Estado. Considerado o maior sistema público de saúde do mundo, sua atuação abrange todo o território nacional, contemplando desde a atenção básica até procedimentos de alta complexidade (PAIM et al., 2011; MINISTÉRIO DA SAÚDE, 2025).

A Atenção Primária à Saúde (APS), organizada prioritariamente por meio da Estratégia Saúde da Família (ESF), constitui a porta de entrada preferencial do sistema. Segundo o Ministério da Saúde (2025), 47,9% dos brasileiros identificam as Unidades Básicas de Saúde (UBS) como seu principal ponto de contato com o SUS. Nesse cenário, a ESF apresenta expansão contínua: em 2024, a cobertura nacional atingiu 88%, com destaque para a região Nordeste, onde 94% das UBS contam com equipes especializadas (CARVALHO, 2025).

A relevância da APS é ratificada por sua elevada capacidade resolutiva, estimando-se que até 80% dos problemas de saúde sejam solucionados na atenção básica, sem a necessidade de encaminhamentos para serviços especializados ou hospitalares (MENDES, 2012). Consequentemente, a gestão eficiente das UBS torna-se um fator determinante para o desempenho e a sustentabilidade de toda a rede de saúde.

2.2 Desafios da Gestão em Saúde no Brasil

Apesar dos avanços alcançados, a gestão da saúde no Brasil ainda enfrenta desafios estruturais significativos. O tempo de espera para atendimento permanece como um dos principais entraves históricos do Sistema Único de Saúde (SUS), conforme apontado por AGÊNCIA BRASIL (2025). Dados da Pesquisa Nacional de Saúde (PNS) corroboram essa dificuldade, revelando que 32,7% dos usuários que buscaram atendimento não obtiveram êxito, seja por falta de vagas ou pela indisponibilidade de senhas (BIBLIOTECA VIRTUAL EM SAÚDE – MINISTÉRIO DA SAÚDE, 2025).

Paralelamente, o Censo Nacional das UBS (2024) identificou que 60,1% das unidades necessitam de reformas físicas. Essa precariedade estrutural, somada à deficiência nos processos de agendamento, prejudica a resolutividade da atenção básica, gerando um efeito cascata que sobrecarrega tanto os serviços especializados quanto as unidades de emergência hospitalar (CARVALHO, 2025).

Nesse cenário, a digitalização da gestão em saúde surge como uma estratégia fundamental para mitigar tais gargalos. Embora o Ministério da Saúde (2025) aponte que a telessaúde pode reduzir em até 30% o tempo de espera, a implementação tecnológica requer um planejamento cuidadoso. É imprescindível que os sistemas sejam desenhados especificamente para a gestão do acesso, priorizando a otimização de agendamentos e o controle eficiente das filas de espera, conforme defendido por PEREIRA et al. (2020).

2.3 Sistemas de Informação em Saúde

Sistemas de Informação em Saúde (SIS) são ferramentas tecnológicas que auxiliam na coleta, armazenamento, processamento e análise de dados relacionados à saúde da população e à gestão dos serviços (MARIN; MASSAD, 2016). No contexto do SUS, os SIS têm sido fundamentais para o planejamento, monitoramento e avaliação de políticas públicas.

O Brasil tem avançado na digitalização da atenção primária. O Censo das UBS 2024 aponta que 94,6% das unidades têm acesso à internet e 87% utilizam prontuário eletrônico (MINISTÉRIO DA SAÚDE, 2025). No entanto, a presença de tecnologia não garante, por si só, a melhoria da gestão. Como destacam Santos e Souza (2021), é necessário que os sistemas sejam integrados e projetados para resolver problemas específicos, como o agendamento de consultas e a continuidade do cuidado.

A saúde digital, ou e-Health, é definida pela Organização Mundial da Saúde (OMS, 2018) como o uso de tecnologias de informação e comunicação para apoiar a saúde e áreas relacionadas. Estudos mostram que a implementação de soluções digitais para agendamento pode reduzir significativamente as taxas de ausência (no-show), otimizar a alocação de recursos e melhorar a satisfação dos usuários (GIBSON et al., 2018; PEREIRA et al., 2020).

3 METODOLOGIA

A Metodologia descreve os procedimentos técnicos e científicos adotados para o desenvolvimento do sistema de gestão de agendamentos para Unidades Básicas de Saúde. Este capítulo detalha a abordagem metodológica, o ciclo de vida do software, os procedimentos para levantamento de requisitos, as ferramentas e tecnologias utilizadas, bem como os critérios de validação do sistema.

3.1 Abordagem Metodológica

O presente trabalho caracteriza-se como uma pesquisa aplicada, pois visa à produção de conhecimento para a solução de um problema prático e específico: a ineficiência na gestão de agendamentos em unidades de saúde. Quanto à abordagem, adota-se uma pesquisa qualitativa, uma vez que o foco está na análise dos processos e na proposição de uma solução tecnológica que atenda às necessidades dos usuários (pacientes, profissionais e administradores), sem a pretensão de generalização estatística.

Do ponto de vista dos objetivos, a pesquisa classifica-se como exploratória e descritiva. Exploratória, pois envolve levantamento bibliográfico e análise de sistemas existentes para compreender o estado da arte em gestão de agendamentos. Descritiva, na medida em que descreve o desenvolvimento do sistema, suas funcionalidades e a arquitetura adotada.

3.2 Procedimentos Técnicos

Para o desenvolvimento do sistema, adotou-se o modelo de ciclo de vida incremental, com entregas parciais e funcionais (MVP - Minimum Viable Product). Este modelo foi escolhido por sua flexibilidade, permitindo que o sistema seja desenvolvido em etapas, com validação contínua e ajustes ao longo do processo.

Figura 1 - Modelo de Ciclo de Vida Incremental

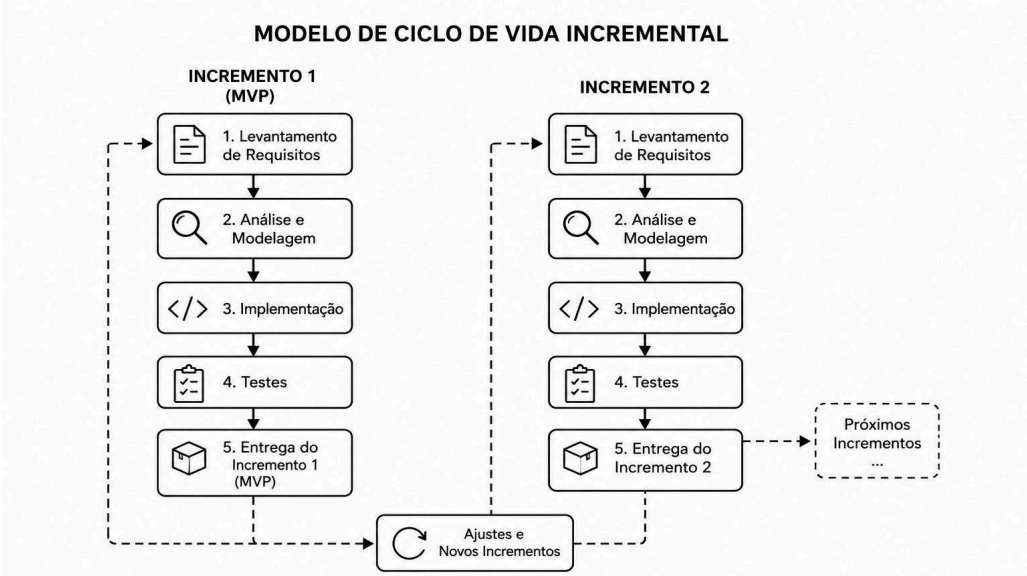


Figura 1 – Modelo de ciclo de vida incremental adotado no desenvolvimento do sistema.

Fonte: Elaborado pelos autores (2026).

Fonte: Elaborado pelos autores (2026).

As etapas do desenvolvimento foram organizadas conforme a listagem a seguir:

- **Levantamento de Requisitos:** Identificação das necessidades dos usuários e definição das funcionalidades do sistema.
- **Análise e Modelagem:** Definição da arquitetura, modelagem do banco de dados e criação dos diagramas UML.
- **Projeto (Implementação):** Codificação do backend (Flask), frontend (React) e integração com o banco de dados (PostgreSQL).
- **Testes:** Validação das funcionalidades por meio de testes automatizados (pytest) e testes manuais de usabilidade.
- **Entrega e Documentação:** Elaboração da documentação técnica e das instruções de execução.

3.2.1 Tecnologias Utilizadas

O desenvolvimento do sistema utilizou tecnologias modernas voltadas ao desenvolvimento de aplicações web. O frontend foi implementado utilizando React, biblioteca JavaScript desenvolvida pela Meta para construção de interfaces de usuário. O backend foi desenvolvido com Flask, framework web Python utilizado na criação da API REST, em conjunto com Flask-RESTful. Para persistência dos dados foi utilizado PostgreSQL, sistema gerenciador de banco de dados relacional de código aberto.

3.3 Requisitos

Esta seção é dedicada à apresentação dos requisitos que foram ponto de partida para o desenvolvimento da aplicação.

3.3.1 Requisitos Funcionais

Os requisitos funcionais descrevem os serviços que o sistema oferece ao usuário, desta forma, apresentamos os requisitos funcionais que foram implementados no sistema:

RF001 – Cadastrar Paciente
Descrição: Permite que um novo paciente se cadastre no sistema fornecendo seus dados pessoais e criando uma credencial de acesso.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF002 – Consultar Paciente
Descrição: Permite visualizar o cadastro de um paciente
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF003 – Atualizar Paciente
Descrição: Permite atualizar o cadastro do paciente
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF004 – Excluir Paciente

Descrição: Permite excluir o cadastro de um paciente
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF005 – Autenticar Usuário
Descrição: Permite que paciente (via CPF), profissional ou administrador (via email) autentiquem-se no sistema, recebendo tokens JWT (access e refresh) para acessar recursos protegidos.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF006 – Cadastrar Profissional
Descrição: Permite que o administrador cadastre um profissional de saúde, incluindo suas credenciais de acesso (login = email) e dados profissionais.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF007 – Consultar Profissional
Descrição: Permite que administradores consultem os profissionais cadastrados, visualizando suas informações pessoais e profissionais.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF008 – Atualizar Profissional
Descrição: Permite que administradores alterem os dados de profissionais previamente registrados.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF009 – Excluir Profissional
Descrição: Permite que administradores removam profissionais cadastrados do sistema, respeitando as regras de integridade dos dados.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF010 – Cadastrar Administrador
Descrição: Permite que um administrador (com permissão) cadastre outro administrador no sistema.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF011 – Consultar Administrador
Descrição: Permite que um administrador consulte outro administrador no sistema.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF012 – Atualizar Administrador
Descrição: Permite que um administrador (com permissão) atualize o cadastro de outro administrador no sistema.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF013 – Excluir Administrador
Descrição: Permite que um administrador (com permissão) exclua o cadastro de outro administrador no sistema.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF014 – Gerenciar Horários Disponíveis
Descrição: Permite que o administrador crie, leia, atualize e delete horários de atendimento para profissionais. A criação deve validar conflito de horário (mesmo profissional, mesma data, horários sobrepostos).
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF015 – Consultar Horários Disponíveis
Descrição: Permite que pacientes (autenticados) visualizem horários disponíveis para agendamento, com filtros opcionais por profissional, data e especialidade.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF016 – Agendar Consulta
Descrição: Permite que um paciente autenticado agende uma consulta em um horário disponível. Deve validar a disponibilidade do horário e impedir que o mesmo paciente agende dois horários conflitantes.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF017 – Cancelar Agendamento
Descrição: Permite que um paciente autenticado cancele uma consulta agendada. Apenas permitido com antecedência mínima de 24 horas em relação ao horário da consulta. Ao cancelar, o horário deve ser liberado (disponível novamente).
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF018 – Consultar Histórico de Agendamentos
Descrição: Permite que pacientes visualizem todos os seus agendamentos (passados, ativos e cancelados), com filtros opcionais por status.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF019 – Consultar Agenda do Profissional
Descrição: Permite que profissionais de saúde autenticados visualizem seus horários e agendamentos do dia/semana.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF020 – Consultar Indicadores Gerenciais
Descrição: Permite que administradores visualizem métricas agregadas do sistema: total de agendamentos por período, taxa de ocupação dos profissionais, taxa de faltas, etc.
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

RF021 – Gerenciar Fila de Espera
Descrição: Permite que pacientes entrem em fila de espera para uma especialidade quando não houver horários disponíveis. A fila deve ser priorizada automaticamente com base em critérios: idade ≥ 60 anos (prioridade alta), gestantes/cadeirantes (prioridade média), demais (prioridade baixa).
Prioridade: ☐ Essencial ☐ Importante ☑ Desejável

3.3.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles ligados às propriedades do sistema e não se relacionam a serviços oferecidos diretamente ao usuário. Desta forma, apresentamos os requisitos não funcionais que foram implementados no sistema:

RNF001 – Controle de Concorrência
Descrição: O sistema deve impedir que dois pacientes agendem o mesmo horário simultaneamente. A operação de criação de agendamento deve ser atômica (transação no banco de dados), utilizando lock otimista ou isolamento de transação adequado.
Categoria: Integridade
Prioridade: ☐ Essencial ☑ Importante ☐ Desejável

RNF002 – Padrão de Projeto
Descrição: O código-fonte deve seguir boas práticas de nomenclatura e organização. A API deve ser documentada via Swagger/OpenAPI.
Categoria: Manutenibilidade
Prioridade: ☐ Essencial ☑ Importante ☐ Desejável

RNF003 – Interface e Experiência do Usuário
Descrição: O frontend deve ser responsivo e intuitivo. Mensagens de erro devem ser claras e amigáveis. O fluxo de agendamento deve ser concluído em no máximo 3 telas/cliques.
Categoria: Usabilidade
Prioridade: ☐ Essencial ☑ Importante ☐ Desejável

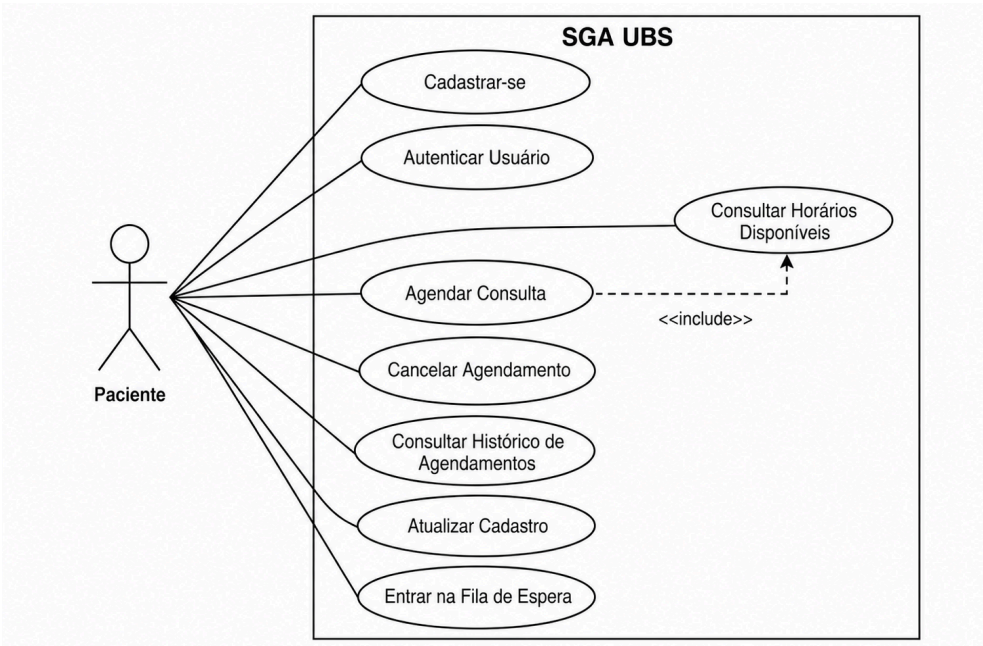
RNF004 – Testes Automatizados
Descrição: O backend deve conter um conjunto mínimo de testes automatizados (pytest) cobrindo as regras de negócio mais críticas: impedimento de duplo agendamento, validação de cancelamento com 24h de antecedência, conflito de horários.
Categoria: Qualidade de Software
Prioridade: ☐ Essencial ☐ Importante ☑ Desejável

RNF005 – Segurança de Dados e Autenticação
Descrição: O sistema deve armazenar senhas utilizando algoritmo de hash. A comunicação entre cliente e servidor deve utilizar tokens JWT. Rotas sensíveis devem validar permissões por tipo de usuário.
Categoria: Segurança
Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

3.3.3 Diagrama de casos de usos

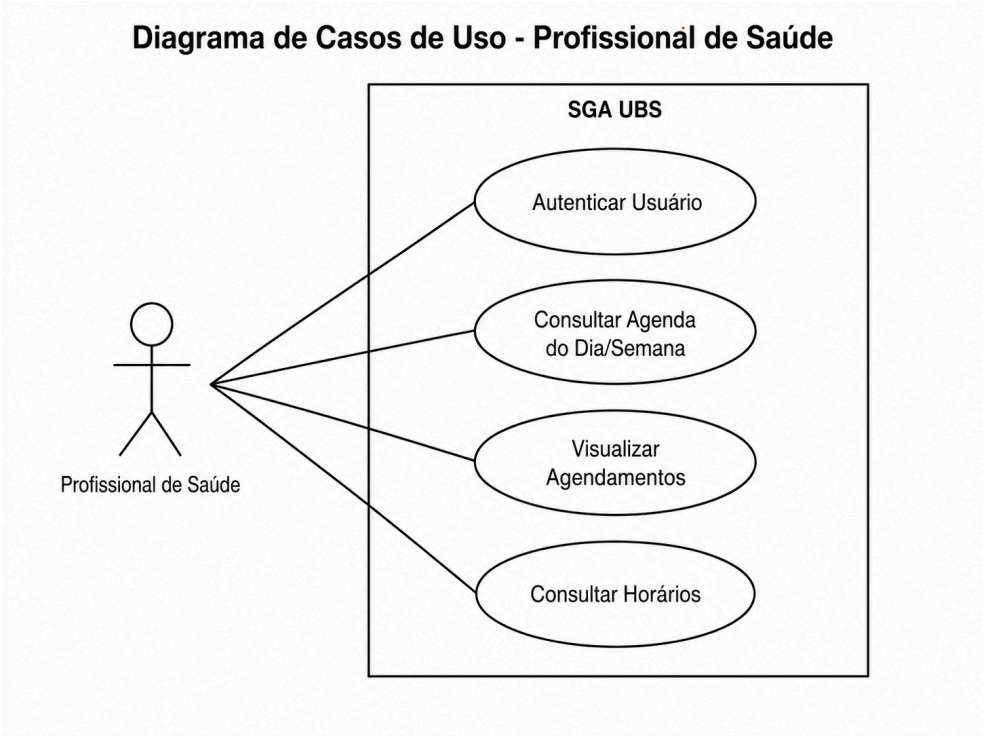
A seção de Diagramas de Casos de Uso apresenta a modelagem visual das interações funcionais entre os diferentes perfis de usuários, pacientes, profissionais de saúde e administradores. Esses diagramas ilustram, de forma clara e estruturada, as fronteiras do sistema e o papel de cada ator na execução de operações essenciais, como a marcação e o cancelamento de consultas, o gerenciamento de disponibilidades na agenda, o controle da fila de espera e o acesso aos relatórios gerenciais, facilitando a compreensão sistêmica dos requisitos estabelecidos.

Figura 2 - Diagrama de casos de usos do Paciente



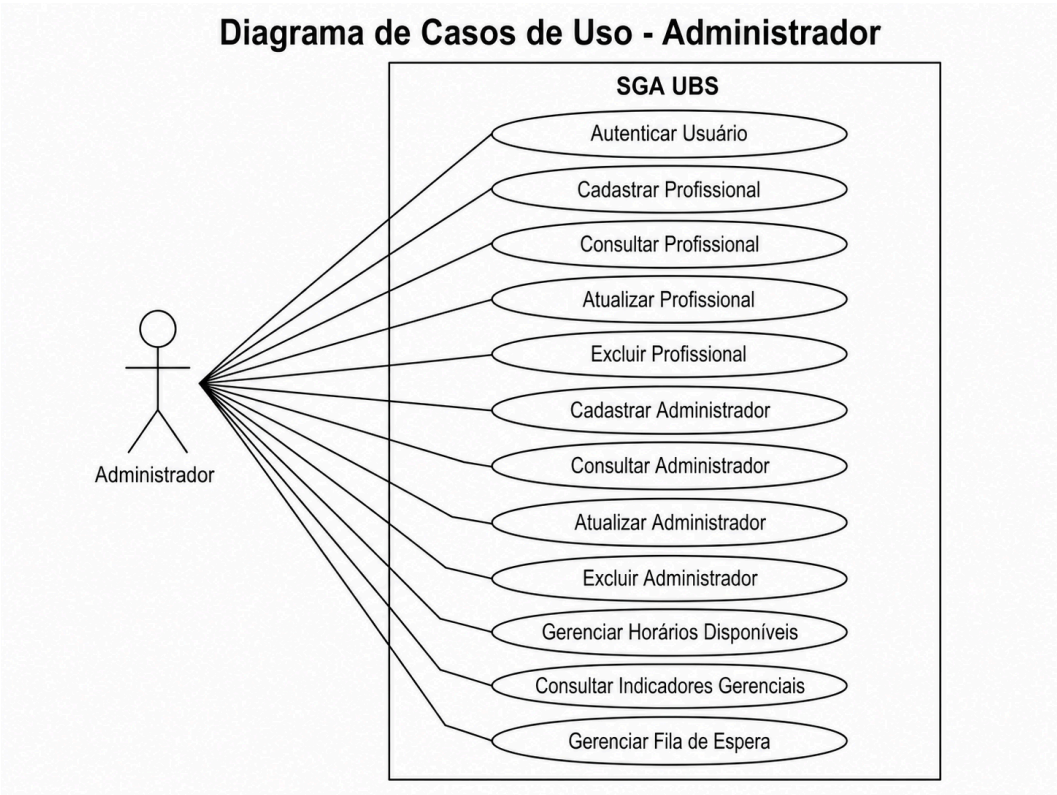
Fonte: Elaborado pelos autores (2026).

Figura 3 - Diagrama de Casos de Uso do Profissional



Fonte: Elaborado pelos autores (2026).

Figura 4 - Diagrama de casos de Uso do Administrador



Fonte: Elaborado pelos autores (2026).

3.4 Tecnologias e Ferramentas

As tecnologias e ferramentas foram selecionadas com base na adequação ao escopo do projeto, na maturidade técnica, na facilidade de implementação e na capacidade de atender aos requisitos funcionais e não funcionais definidos. O sistema adota uma arquitetura cliente-servidor baseada em API REST, com separação clara entre frontend, backend e banco de dados.

3.4.1 Tecnologias usadas

3.4.1.1 React (Frontend)

O React é uma biblioteca JavaScript de código aberto mantida pelo Meta (META PLATFORMS, INC., 2025), amplamente utilizada para a construção de interfaces de usuário interativas e responsivas. Foi escolhido por sua popularidade no mercado, vasto ecossistema de bibliotecas e componentes, suporte a hooks e estado reativo, e pela facilidade de criar interfaces dinâmicas que respondem em tempo real às interações do usuário.

No sistema proposto, o React é responsável por:

- Renderizar as interfaces para os três perfis de usuário: paciente, profissional de saúde e administrador da UBS.
- Consumir a API REST desenvolvida no backend, realizando requisições HTTP para enviar e receber dados.
- Gerenciar o estado da aplicação (ex: dados do usuário logado, lista de horários disponíveis, agendamentos do paciente).
- Garantir uma experiência de usuário fluida, com transições suaves entre telas e feedback imediato às ações do usuário.

3.4.1.2 Flask (Backend)

O Flask é um framework web leve e flexível para o desenvolvimento de aplicações e APIs REST em Python. Foi escolhido por sua simplicidade, extensibilidade e amplo

ecossistema. No projeto, é utilizado com Flask-RESTful para organizar os recursos da API, Marshmallow para serialização e validação dos dados e Flask-SQLAlchemy para integração com o banco de dados.

No sistema proposto, o Flask é responsável por:

- Expor os endpoints REST para todas as operações do sistema (cadastro, login, agendamento, cancelamento, etc.).
- Implementar as regras de negócio, como validação de conflitos de horário, antecedência mínima para cancelamento (24h) e priorização automática na fila de espera.
- Gerenciar a autenticação JWT (JSON Web Tokens), garantindo que apenas usuários autenticados acessem recursos protegidos.
- Disponibilizar a documentação da API via Swagger/OpenAPI, facilitando o entendimento e o uso dos endpoints.

3.4.1.3 PostgreSQL (Banco de Dados)

O PostgreSQL é um sistema gerenciador de banco de dados relacional (SGBD) de código aberto, reconhecido por sua confiabilidade, robustez e conformidade com padrões SQL. Foi escolhido por seu suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), integridade referencial e capacidade de lidar com consultas complexas.

No sistema proposto, o PostgreSQL é responsável por:

- Armazenar persistentemente todas as informações do sistema: usuários, horários disponíveis, agendamentos, fila de espera e notificações.
- Garantir a integridade dos dados por meio de restrições de chave estrangeira, unicidade e validações no nível do banco.
- Suportar a implementação de locks otimistas e isolamento de transações, fundamentais para o controle de concorrência (RNF001) e para evitar que dois pacientes agendem o mesmo horário simultaneamente.

3.4.1.4 JWT (Autenticação)

O JWT (JSON Web Token) é um padrão para autenticação stateless em APIs REST. Foi escolhido por permitir que o servidor não mantenha estado de sessão, tornando a autenticação escalável e adequada para arquiteturas distribuídas.

No sistema proposto, o JWT é responsável por:

- Autenticar os três perfis de usuário (paciente, profissional e administrador) no momento do login.
- Assinar digitalmente os tokens, garantindo integridade e autenticidade nas requisições subsequentes.
- Controlar o acesso a recursos protegidos, diferenciando permissões por tipo de usuário.

3.4.1.5 Pytest (Testes Automatizados)

O Pytest é um framework para testes automatizados em Python, amplamente utilizado no ecossistema da linguagem. Foi escolhido por sua sintaxe expressiva, suporte a fixtures e integração simplificada com aplicações Flask. No sistema proposto, o Pytest é responsável por:

- Validar as regras de negócio mais críticas, como impedimento de duplo agendamento, validação de cancelamento com 24h de antecedência e conflito de horários.
- Garantir que o sistema se comporte conforme o esperado antes de cada nova implementação.
- Facilidade de execução local, permitindo que os desenvolvedores verifiquem a integridade do código durante o desenvolvimento.Pytest (Testes Automatizados).

3.4.1.6 Swagger/OpenAPI (Documentação)

O Swagger/OpenAPI é uma especificação para descrever, produzir, consumir e visualizar APIs REST. Foi escolhido por permitir uma documentação interativa e padronizada dos endpoints. No projeto, a especificação é mantida em arquivos locais e disponibilizada pela aplicação. No sistema proposto, o Swagger/OpenAPI é responsável por:

- Documentar todos os endpoints da API, incluindo parâmetros, respostas e códigos de status.
- Permitir que desenvolvedores e testadores visualizem e testem os endpoints diretamente no navegador.
- Facilitar a compreensão do sistema por outros desenvolvedores, gestores e avaliadores.

3.4.2 Ferramentas de Apoio ao Desenvolvimento

3.4.2.1 Visual Studio Code (VSCode)

O Visual Studio Code (Disponível em: https://code.visualstudio.com) é um editor de código-fonte desenvolvido pela Microsoft, amplamente utilizado pela comunidade de desenvolvimento por sua leveza, extensibilidade e suporte a múltiplas linguagens de programação. Foi escolhido como ambiente de desenvolvimento por oferecer:

- Suporte nativo a Python e JavaScript: Com extensões como Python, Pylance e ESLint, o VSCode oferece autocompletar, depuração e análise estática de código, acelerando o desenvolvimento.
- Controle de versionamento integrado: O VSCode possui suporte nativo ao Git, permitindo visualizar alterações, resolver conflitos e realizar commits sem sair do editor.
- Extensibilidade: O vasto ecossistema de extensões permite personalizar o ambiente conforme as necessidades do projeto, adicionando suporte a formatação de código, snippets e ferramentas de produtividade.

No desenvolvimento do sistema, o VSCode foi utilizado para edição do código-fonte do frontend (React) e do backend (Flask), além da execução de comandos de terminal para gerenciar dependências e rodar testes.

3.4.2.2 GitHub (Versionamento)

O GitHub (Disponível em: https://docs.github.com/pt) é uma plataforma de hospedagem de código-fonte e controle de versionamento baseada em Git, amplamente utilizada por desenvolvedores e equipes de software. Foi escolhido por sua integração com o Git, interface amigável e recursos de colaboração. No sistema proposto, o GitHub foi utilizado para:

- Versionamento do código-fonte: Todos os arquivos do projeto (backend, frontend, banco de dados e documentação) foram armazenados em um repositório Git, permitindo o rastreamento de alterações e o histórico completo do desenvolvimento.
- Controle de versões: O uso de commits e branches permitiu organizar o desenvolvimento em funcionalidades específicas, facilitando o trabalho colaborativo entre os membros da equipe.

- Documentação do projeto: O arquivo README.md foi utilizado para descrever o projeto, as tecnologias utilizadas e as instruções de execução, servindo como guia inicial para novos desenvolvedores e avaliadores.
- Backup e acesso remoto: O repositório no GitHub garante que o código-fonte esteja disponível em qualquer lugar, funcionando como um backup seguro do projeto.

3.4.3 Arquitetura do Sistema

O sistema adota uma arquitetura cliente-servidor baseada em API REST, organizada em três camadas principais: Camada de Apresentação (frontend), Camada de Negócio (backend) e Camada de Dados (banco de dados). Essa abordagem, amplamente consolidada no desenvolvimento de software moderno, garante a separação de responsabilidades, facilitando a manutenção, a escalabilidade e a evolução do sistema.

3.4.3.1 Camada de Apresentação (Frontend - React)

A Camada de Apresentação é responsável por toda a interação com o usuário. Implementada utilizando a biblioteca React, esta camada é responsável por renderizar as interfaces gráficas que permitem a comunicação entre os usuários e o sistema. Através do React, foram desenvolvidas interfaces específicas para os três perfis de usuário: paciente, profissional de saúde e administrador da UBS.

Para cada perfil, a interface oferece um conjunto de funcionalidades adequado ao seu papel. O paciente pode visualizar horários disponíveis, realizar agendamentos, cancelar consultas (com antecedência mínima de 24 horas) e acompanhar seu histórico de atendimentos. O profissional de saúde tem acesso à sua agenda diária e semanal, podendo visualizar os pacientes agendados e registrar atendimentos realizados. O administrador da UBS, por sua vez, conta com um painel gerencial completo, que inclui a gestão de profissionais, a definição de horários de trabalho, o acompanhamento de agendamentos, a visualização de relatórios gerenciais e o gerenciamento da fila de espera.

A comunicação entre o frontend e o backend ocorre por meio de requisições HTTP, utilizando o formato JSON para o envio e recebimento de dados. Essa abordagem garante que a interface seja dinâmica, responsiva e capaz de refletir em tempo real as alterações realizadas no sistema, como a confirmação de um agendamento ou a atualização de um horário disponível.

3.4.3.2 Camada de Negócio (Backend - Flask)

A Camada de Negócio é o coração do sistema, sendo responsável por toda a lógica de processamento, validação e gestão dos dados. Implementada utilizando o framework Flask em Python, em conjunto com Flask-RESTful, esta camada expõe um conjunto de endpoints REST que atendem às requisições enviadas pelo frontend.

Na Camada de Negócio são implementadas todas as regras de negócio que garantem o correto funcionamento do sistema. Entre as principais regras, destacam-se:

- Validação de Conflitos de Horário: Ao criar ou atualizar horários de atendimento, o sistema verifica se não há sobreposição para o mesmo profissional na mesma data, garantindo que não haja agendamentos conflitantes.

- Verificação de Disponibilidade para Agendamento: Antes de confirmar um agendamento, o sistema valida se o horário solicitado está realmente disponível e se o paciente não possui outro agendamento conflitante no mesmo período.

- Regra de Antecedência Mínima para Cancelamento: O cancelamento de uma consulta é permitido apenas se for solicitado com pelo menos 24 horas de antecedência em relação ao horário agendado. Ao cancelar, o horário é automaticamente liberado para novos agendamentos.

- Priorização Automática na Fila de Espera: A fila de espera é gerenciada com base em critérios de prioridade: pacientes com idade igual ou superior a 60 anos recebem prioridade alta, gestantes e cadeirantes recebem prioridade média, e os demais pacientes recebem prioridade padrão.

Além das regras de negócio, o backend também é responsável pela autenticação e autorização dos usuários. Utilizando o padrão JWT (JSON Web Tokens), o sistema garante que apenas usuários autenticados possam acessar recursos protegidos, e que cada perfil de usuário tenha acesso apenas às funcionalidades compatíveis com sua função. As senhas dos usuários são armazenadas com hash utilizando o algoritmo bcrypt, garantindo a segurança dos dados mesmo em caso de acesso indevido ao banco de dados.

Outra funcionalidade importante do backend é o controle de concorrência (RNF001). A operação de criação de agendamento é realizada dentro de uma transação atômica no banco de dados, utilizando mecanismos de lock otimista ou isolamento de transação adequado. Isso garante que, mesmo que dois pacientes tentem agendar o mesmo horário simultaneamente,

apenas um deles consiga concluir a operação com sucesso, evitando inconsistências nos dados.

O backend também é responsável por disponibilizar a documentação da API via Swagger/ OpenAPI. A especificação é mantida em arquivos locais e servida em uma URL específica, na qual são listados os endpoints disponíveis, seus parâmetros, formatos de resposta e códigos de status, facilitando o entendimento do sistema por outros desenvolvedores, gestores e avaliadores.

3.4.3.3 Camada de Dados (Banco de Dados - PostgreSQL)

A Camada de Dados é responsável pelo armazenamento persistente de todas as informações do sistema. Utilizando o PostgreSQL, um sistema gerenciador de banco de dados relacional robusto e confiável, esta camada garante a integridade, a consistência e a durabilidade dos dados.

- O modelo lógico do banco de dados foi projetado para atender a todas as necessidades do sistema, contemplando as seguintes entidades principais:

- Usuários: Armazena os dados de pacientes, profissionais de saúde e administradores. A tabela inclui informações como nome, CPF, email, senha (hash), tipo de usuário, data de nascimento e outros dados relevantes.

- Horários Disponíveis: Registra os horários de atendimento de cada profissional. A tabela inclui o profissional associado, a data, o horário de início e fim, e o status (disponível ou ocupado). A validação de conflitos é garantida por restrições de unicidade e por regras de negócio implementadas no backend.

- Agendamentos: Armazenar as consultas marcadas pelos pacientes. A tabela inclui o paciente, o horário agendado, a data de criação, o status (agendado, realizado, cancelado ou faltou) e, opcionalmente, o atendimento realizado e observações.

- Fila de Espera: Gerencia os pacientes que aguardam por uma vaga em uma especialidade. A tabela inclui o paciente, a especialidade desejada, a data de entrada na fila, a prioridade calculada (alta, média ou baixa) e a posição na fila.

- Notificações: Registra as notificações enviadas aos pacientes, como lembretes de consulta, confirmações de agendamento e avisos de disponibilidade de vagas.

O PostgreSQL foi escolhido por seu suporte a transações ACID, que garantem que todas as operações sejam executadas de forma atômica, consistente, isolada e durável. Isso é

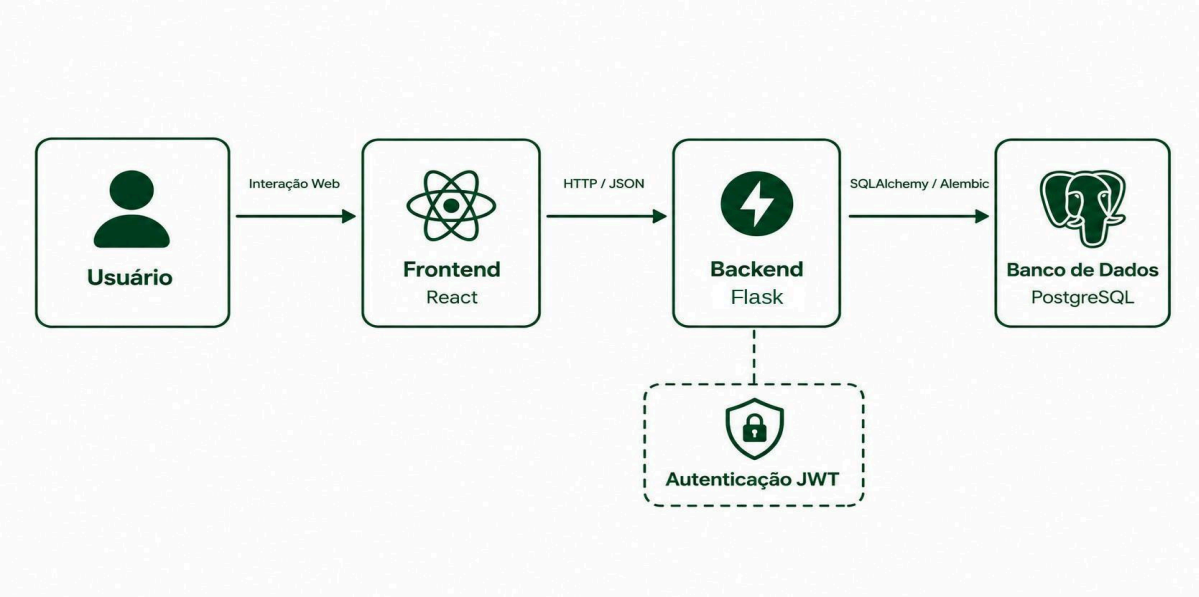
fundamental para a integridade dos dados, especialmente em operações críticas como o agendamento de consultas, onde é necessário garantir que dois pacientes não agendem o mesmo horário simultaneamente. A integridade referencial é garantida por meio de restrições de chave estrangeira, e consultas complexas são otimizadas por meio de índices adequados, garantindo boa performance mesmo com um volume significativo de dados.

4 RESULTADOS E DISCUSSÃO

4.1 Arquitetura do Sistema

O SGA UBS foi desenvolvido com arquitetura cliente-servidor baseada em API REST. O frontend utiliza React para construção das interfaces e interação com os usuários, enquanto o backend foi implementado com Flask, responsável pelas regras de negócio, autenticação JWT e comunicação com o banco de dados. O PostgreSQL realiza o armazenamento das informações, com acesso gerenciado pelo SQLAlchemy e controle de alterações estruturais pelo Alembic.

Figura 5 - Arquitetura do Sistema



Fonte: Elaborado pelos autores (2026).

O frontend foi desenvolvido com React, responsável pela construção das interfaces e pela interação dos usuários com o sistema. A aplicação foi organizada em páginas, formulários e componentes reutilizáveis, permitindo o acesso às funcionalidades conforme o perfil autenticado. A comunicação com o backend ocorre por meio de requisições HTTP, com envio e recebimento de dados no formato JSON.

O backend foi estruturado para concentrar as regras de negócio, validação e operações relacionadas aos usuários, horários, agendamentos, cancelamentos, fila de espera e indicadores gerenciais. Essa camada recebe as solicitações do frontend, processa os dados e retorna respostas padronizadas, mantendo a interface separada da lógica principal do sistema.

O Flask foi utilizado como framework para implementação da API REST, em conjunto com Flask-RESTful. Por meio dele, foram criados os recursos e endpoints responsáveis pelas operações do sistema, incluindo autenticação, cadastros, consultas e atualizações. A serialização e a validação dos dados são realizadas com Marshmallow, enquanto a documentação interativa da API é disponibilizada por meio do Swagger/OpenAPI.

A autenticação foi implementada utilizando JSON Web Token (JWT). Após o login, o sistema gera tokens que identificam o usuário e seu perfil de acesso. Esses tokens são enviados nas requisições às rotas protegidas, permitindo que o backend controle quais funcionalidades podem ser acessadas por pacientes, profissionais e administradores.

O SQLAlchemy foi utilizado para realizar a comunicação entre o backend e o banco de dados. Por meio do mapeamento objeto-relacional, as tabelas foram representadas como classes Python, facilitando a criação de consultas, relacionamentos e operações de inclusão, alteração e exclusão de registros.

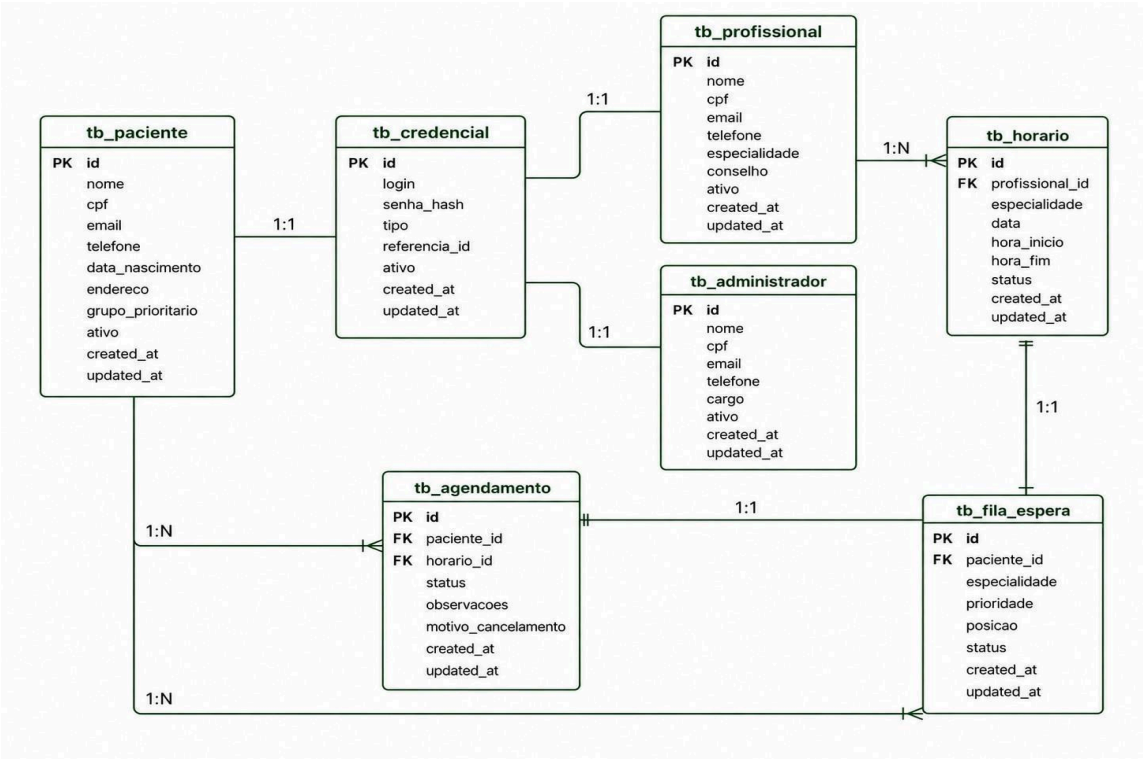
O PostgreSQL foi utilizado como sistema gerenciador de banco de dados relacional. Nele são armazenadas as informações de usuários, horários, agendamentos e filas de espera. Sua utilização permitiu aplicar restrições de integridade, relacionamentos entre tabelas e transações nas operações mais críticas do sistema.

O Alembic foi empregado no gerenciamento das alterações estruturais do banco de dados. As migrações permitiram criar e atualizar tabelas, colunas e relacionamentos de forma versionada, mantendo o esquema do PostgreSQL compatível com a evolução do backend durante o desenvolvimento.

4.2 Modelagem do Sistema

A imagem apresenta o modelo lógico da base de dados do SGA UBS, destacando as principais entidades do sistema e seus relacionamentos. A estrutura contempla dados de pacientes, profissionais, administradores, credenciais de acesso, horários, agendamentos e fila de espera, organizados por meio de chaves primárias e estrangeiras. Esses relacionamentos garantem a integridade das informações e permitem controlar os vínculos entre usuários, consultas, disponibilidade de horários e prioridades de atendimento.

Figura 6 - Modelo lógico da base de dados



Fonte: Elaborado pelos autores (2026).

4.3 Testes de Carga

Os testes de carga foram realizados com o Apache JMeter 5.6.3, com o objetivo de avaliar o comportamento da API do SGA UBS diante de diferentes níveis de acessos simultâneos. Foram definidos três cenários com 100, 500 e 1.000 usuários virtuais, utilizando apenas uma execução por usuário.

Tabela 1 - Especificações da Máquina

Processador	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz (2.11 GHz)
RAM instalada	16,0 GB (utilizável: 15,8 GB)

Placa gráfica	Intel(R) UHD Graphics (128 MB)
Armazenamento	1 TB (HDD), 128 GB (SSD)
Sistema Operacional	Windows 11 Home Single Language

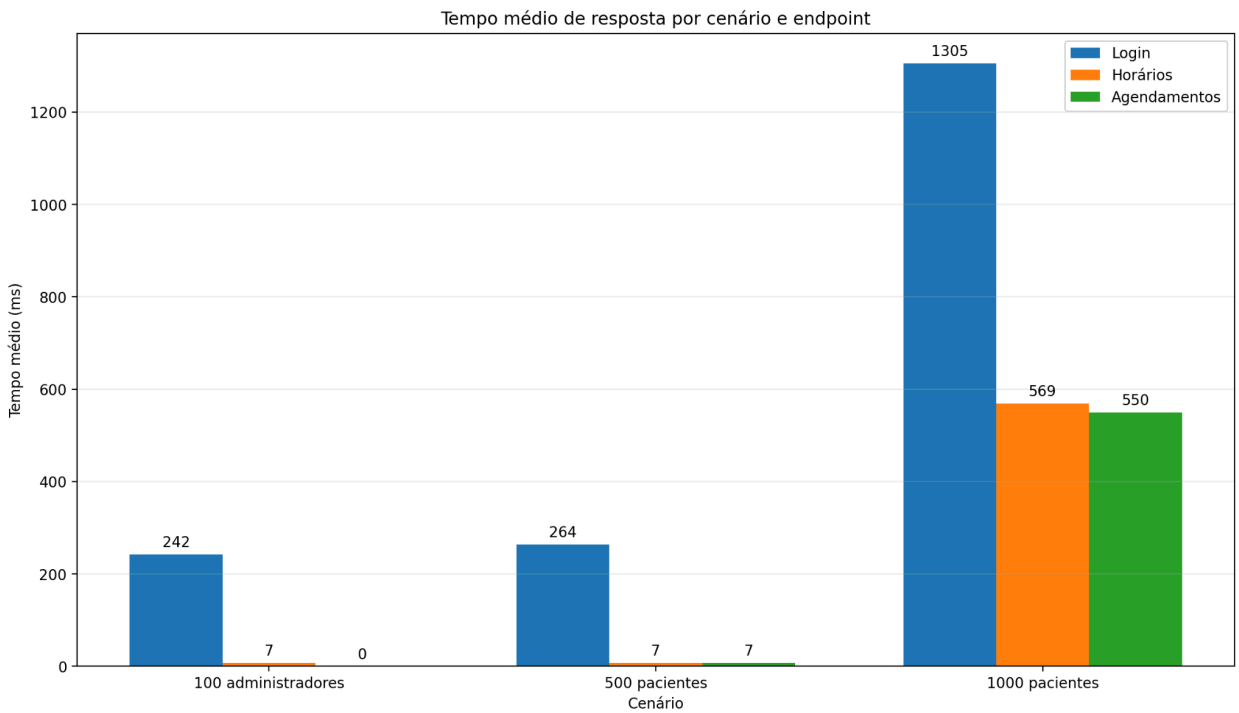
No primeiro cenário, foram simulados 100 administradores, com ramp-up de 20 segundos. Cada usuário realizou a autenticação e uma requisição do tipo GET ao endpoint /horarios. No segundo cenário, foram utilizados 500 pacientes, com ramp-up de 40 segundos. No terceiro, foram simulados 1.000 pacientes, com ramp-up de 60 segundos. Nos dois últimos cenários, cada paciente realizou autenticação, consulta ao endpoint /horarios e tentativa de agendamento por meio de uma requisição POST ao endpoint /agendamentos.

Nos testes de agendamento, todos os usuários virtuais de um mesmo cenário tentaram reservar o mesmo horário. Essa configuração foi adotada propositalmente para verificar a regra de negócio responsável por impedir que mais de um paciente agende a mesma vaga.

4.3.1 Tempo de resposta

O gráfico abaixo apresenta o tempo médio de resposta dos principais endpoints avaliados.

Gráfico 1 – Tempo médio de resposta por cenário e endpoint



Fonte: Elaborado pelos autores (2026).

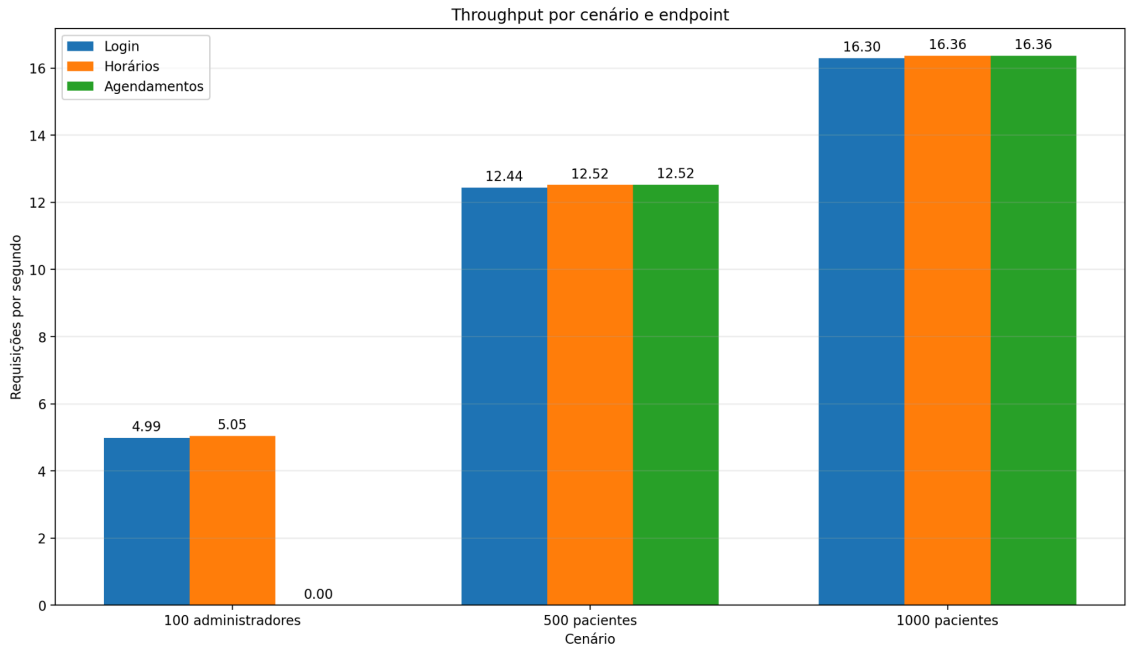
No cenário com 100 administradores, o login apresentou tempo médio de aproximadamente 242 ms, enquanto a consulta de horários permaneceu próxima de 7 ms. No cenário de 500 pacientes, o login registrou cerca de 264 ms, e as operações de consulta e agendamento permaneceram próximas de 7 ms.

Com 1.000 pacientes simultâneos, observou-se uma elevação significativa dos tempos médios: o login atingiu aproximadamente 1.305 ms, a consulta de horários chegou a 569 ms e o agendamento registrou cerca de 550 ms. Esses resultados indicam que o sistema manteve desempenho estável nos cenários de 100 e 500 usuários, mas apresentou degradação mais evidente sob a carga de 1.000 usuários virtuais.

4.3.2 Throughput

O gráfico 2 apresenta a quantidade média de requisições processadas por segundo em cada cenário.

Gráfico 2 – Throughput por cenário e endpoint



Fonte: Elaborado pelos autores (2026).

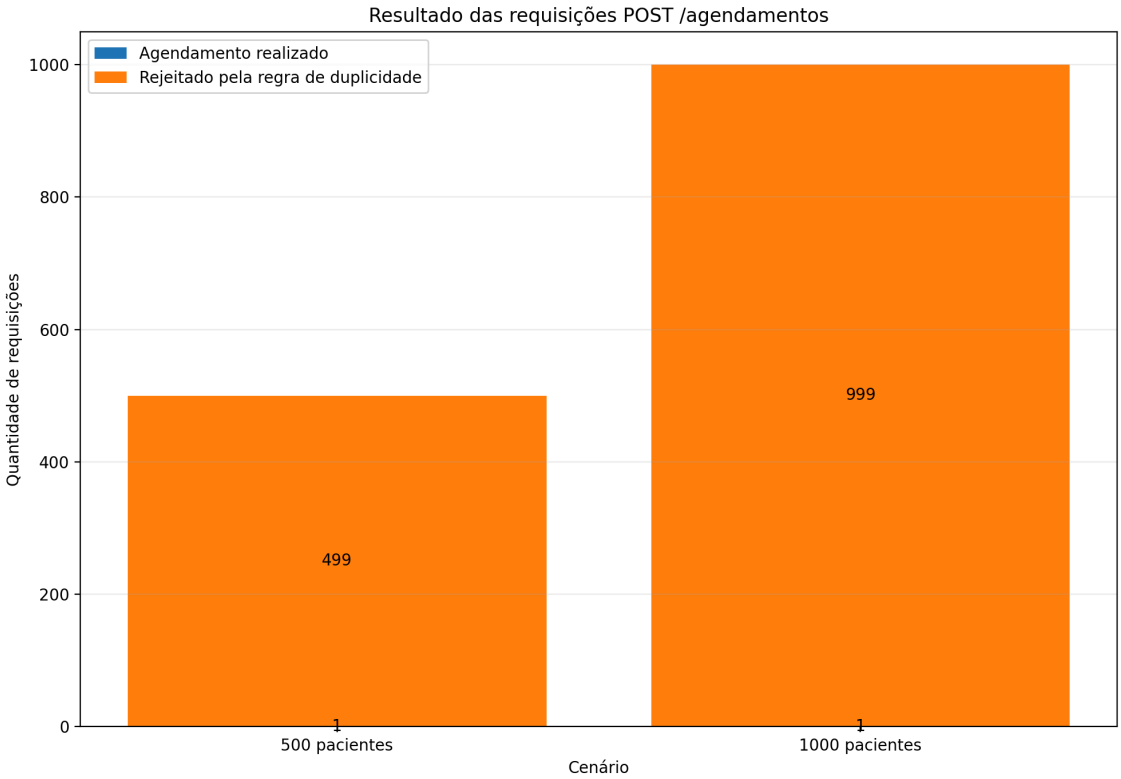
O throughput aumentou conforme a quantidade de usuários virtuais foi ampliada. No cenário de 100 administradores, o sistema processou aproximadamente 5 requisições por segundo. Com 500 pacientes, o valor ficou próximo de 12,5 requisições por segundo, enquanto no cenário de 1.000 pacientes foram processadas cerca de 16,3 requisições por segundo.

Apesar do crescimento do throughput, o aumento observado entre 500 e 1.000 usuários não foi proporcional à duplicação da carga. Somado ao aumento dos tempos de resposta, esse comportamento sugere que o sistema se aproximou do limite de capacidade do ambiente utilizado durante o cenário mais intenso.

4.3.3 Validação do controle de concorrência

O gráfico 3 apresenta os resultados das requisições realizadas ao endpoint POST /agendamentos.

Gráfico 3 – Resultado das requisições de agendamento concorrente



Fonte: Elaborado pelos autores (2026).

No cenário com 500 pacientes, apenas uma requisição conseguiu efetivar o agendamento, enquanto as outras 499 foram rejeitadas. De forma semelhante, no cenário com 1.000 pacientes, apenas um agendamento foi concluído e as 999 solicitações restantes foram recusadas.

As taxas de erro de 99,8% e 99,9% apresentadas pelo JMeter não representam falhas de disponibilidade da aplicação. Esses resultados ocorreram porque todos os usuários disputaram o mesmo horário e, conforme a regra de negócio definida, somente uma solicitação poderia ser aceita.

Dessa forma, os testes demonstraram que o mecanismo de controle de concorrência funcionou corretamente, impedindo a criação de agendamentos duplicados mesmo diante de centenas de requisições simultâneas.

4.3.4 Síntese dos resultados

Os resultados indicam que a API apresentou comportamento satisfatório nos cenários de 100 e 500 usuários virtuais, mantendo tempos de resposta reduzidos e estabilidade nas operações avaliadas. No cenário com 1.000 pacientes, houve aumento dos tempos de resposta e menor crescimento proporcional do throughput, evidenciando o início da degradação do desempenho.

Entretanto, mesmo no cenário de maior carga, a regra de integridade relacionada ao duplo agendamento permaneceu válida. Assim, o sistema demonstrou capacidade de preservar a consistência dos dados em situações de alta concorrência, embora sejam necessárias futuras otimizações de infraestrutura e processamento para melhorar o desempenho em cargas próximas ou superiores a 1.000 usuários simultâneos.

4.4 Telas do Sistema

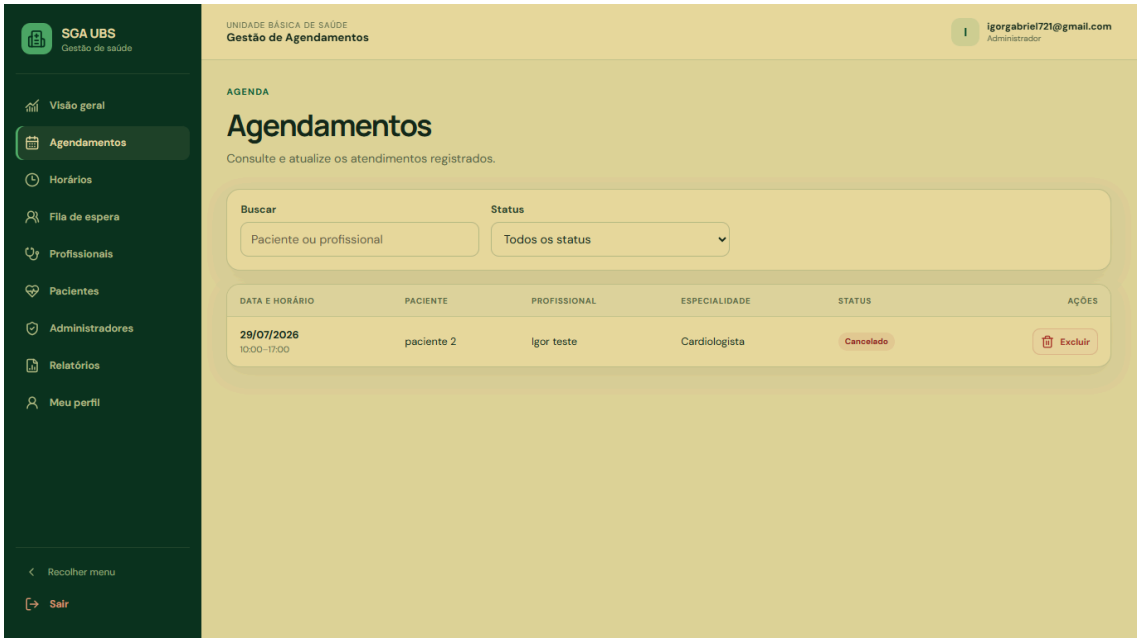
Esta seção apresenta as principais interfaces desenvolvidas para o SGA UBS, destacando as funcionalidades disponíveis para os diferentes perfis de usuário. As telas foram organizadas de modo a facilitar o acesso às operações de agendamento, gerenciamento de horários, controle da fila de espera, administração de usuários e consulta de informações gerenciais.

Figura 7 - Tela de Login

Fonte: Elaborado pelos autores (2026).

A tela de login representa o ponto inicial de acesso ao sistema. Nela, pacientes, profissionais de saúde e administradores podem informar suas credenciais para autenticação. A interface também disponibiliza uma opção de cadastro para novos pacientes, permitindo o acesso inicial ao sistema de forma simplificada.

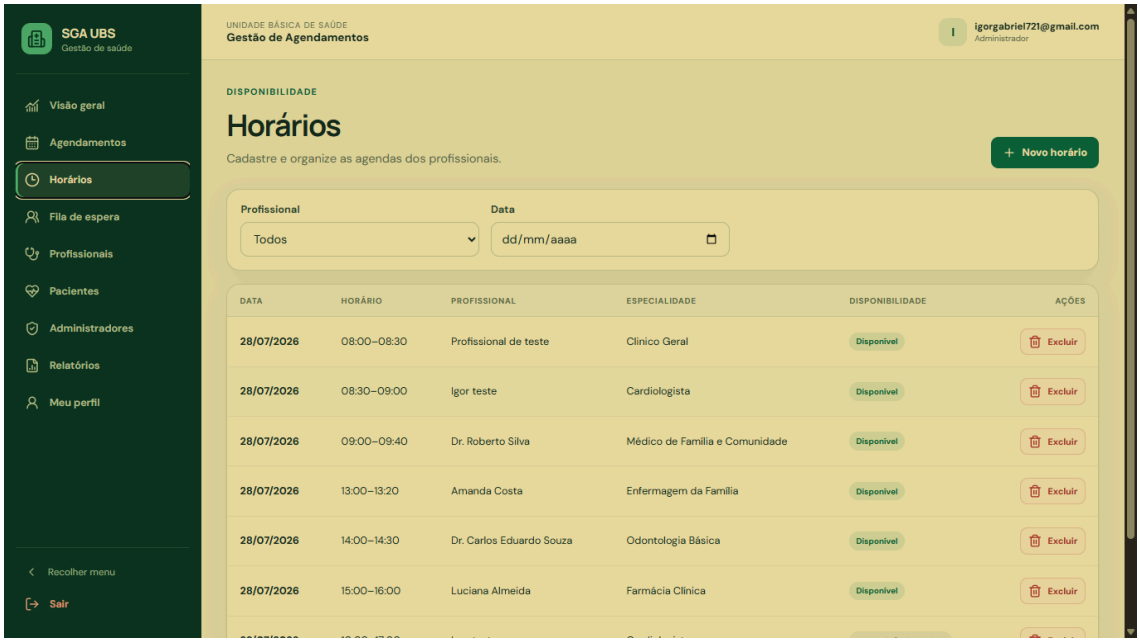
Figura 8 - Tela de Agendamentos



Fonte: Elaborado pelos autores (2026).

A tela de agendamentos permite visualizar e acompanhar as consultas registradas no sistema. São apresentados dados como paciente, profissional responsável, especialidade, horário e situação do agendamento. Também são disponibilizados recursos de busca e filtragem, facilitando a localização e o gerenciamento das consultas cadastradas.

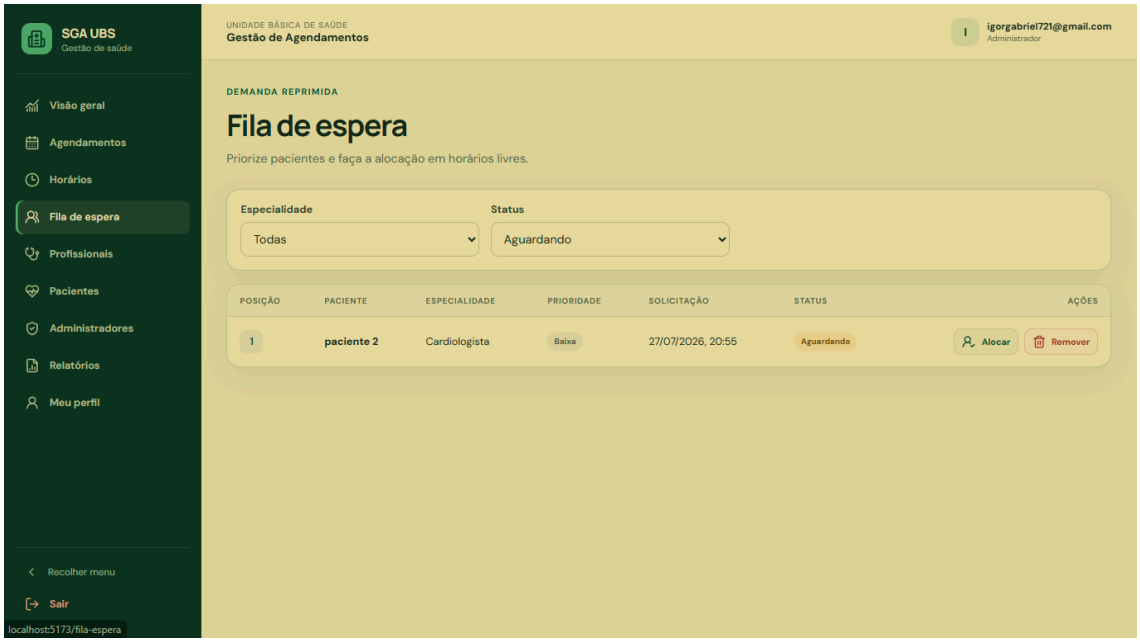
Figura 9 - Tela de Horários



Fonte: Elaborado pelos autores (2026).

A tela de horários apresenta as disponibilidades de atendimento cadastradas para os profissionais de saúde. O usuário pode consultar informações como data, faixa de horário, profissional, especialidade e situação da disponibilidade. Para usuários com permissão administrativa, a interface também possibilita o cadastro e a exclusão de horários.

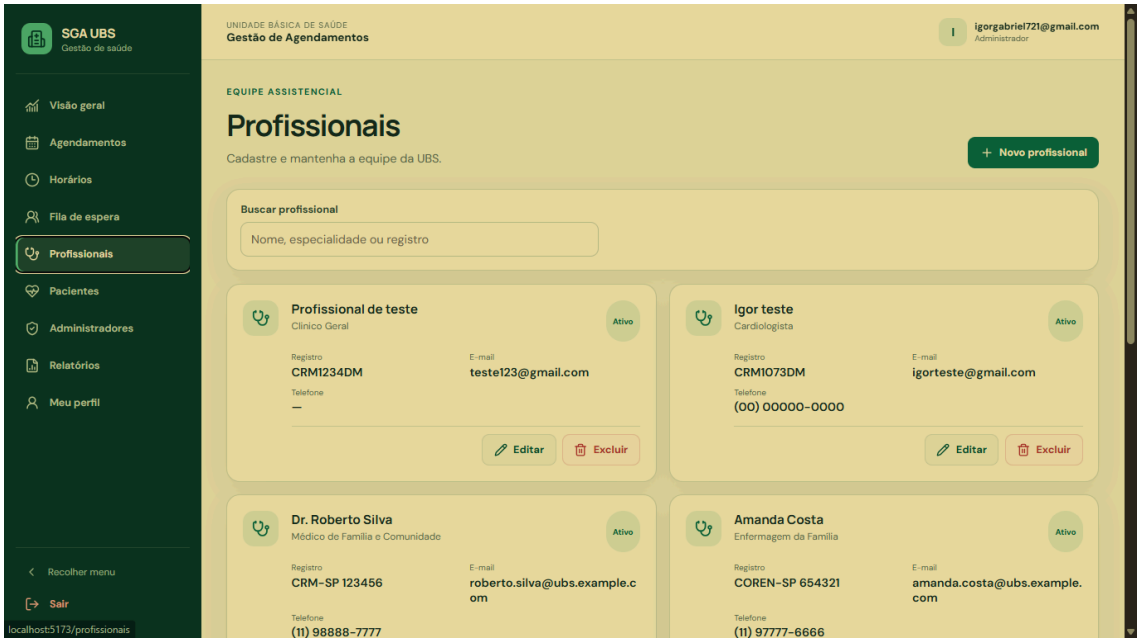
Figura 10 - Tela de Fila de espera



Fonte: Elaborado pelos autores (2026).

A tela de fila de espera permite acompanhar os pacientes que aguardam por uma vaga de atendimento. A interface apresenta informações como posição na fila, paciente, especialidade, prioridade, data da solicitação e status. Essa funcionalidade auxilia a organização da demanda quando não existem horários disponíveis para determinada especialidade.

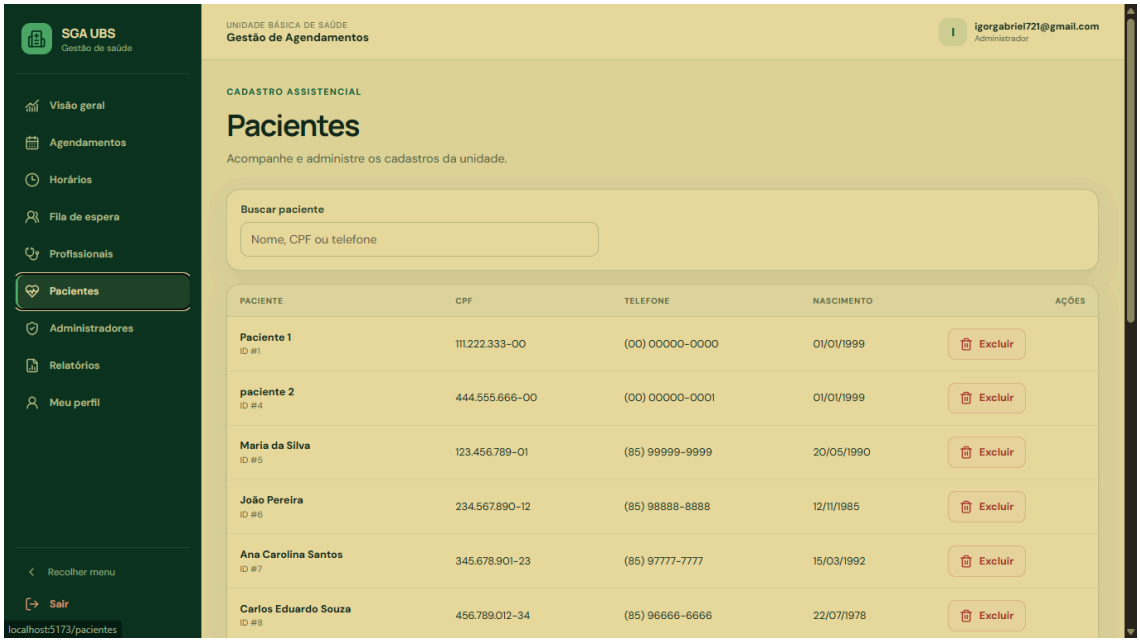
Figura 11 - Tela de Profissionais



Fonte: Elaborado pelos autores (2026).

A tela de profissionais reúne os dados dos profissionais de saúde cadastrados na unidade. A interface permite realizar buscas e visualizar informações como nome, especialidade, registro profissional, contato e situação do cadastro. Também são disponibilizadas ações para inclusão, edição e exclusão de profissionais, conforme as permissões do usuário administrador.

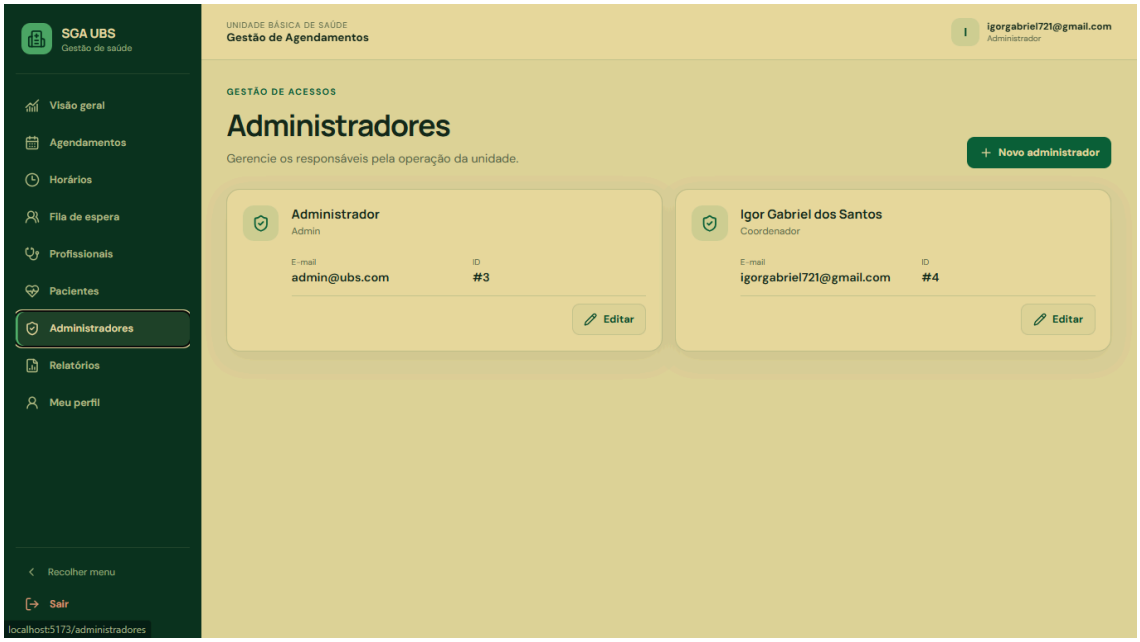
Figura 12 - Tela de Pacientes



Fonte: Elaborado pelos autores (2026).

A tela de pacientes apresenta os usuários cadastrados no sistema, permitindo ao administrador consultar informações básicas dos pacientes. A busca pode ser realizada por dados como nome ou CPF, facilitando a localização dos registros. A interface também disponibiliza ações administrativas relacionadas ao gerenciamento desses cadastros.

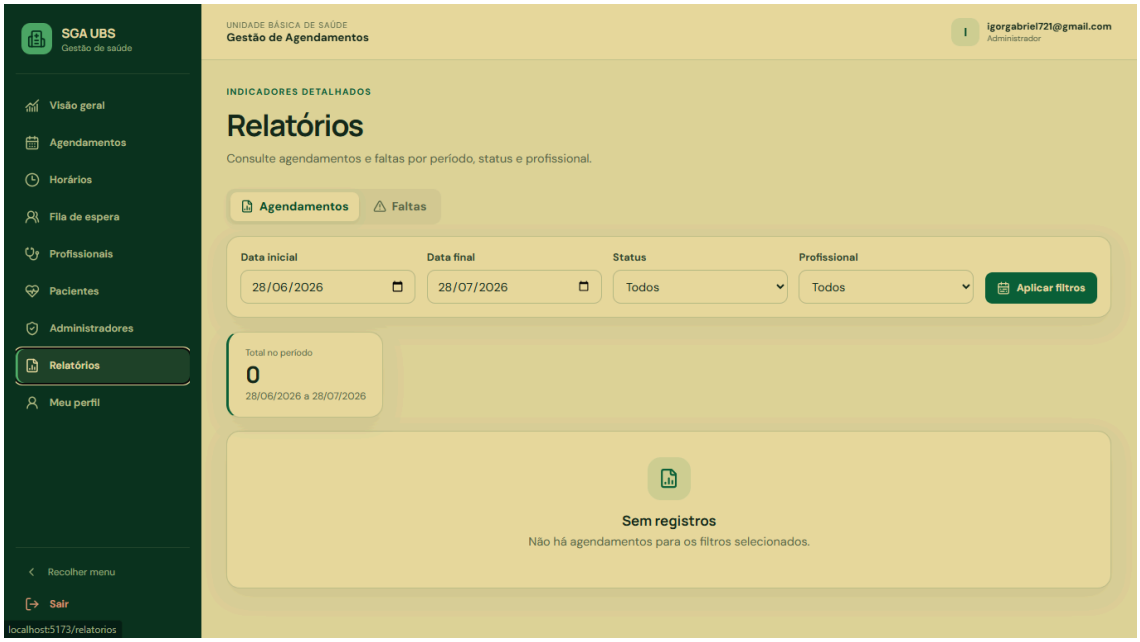
Figura 13 - Tela de Administradores



Fonte: Elaborado pelos autores (2026).

A tela de administradores é destinada ao gerenciamento dos usuários responsáveis pela administração do sistema. Nela são exibidas informações dos administradores cadastrados, como nome, e-mail e cargo. A interface possibilita ainda a inclusão de novos administradores e a edição dos registros existentes.

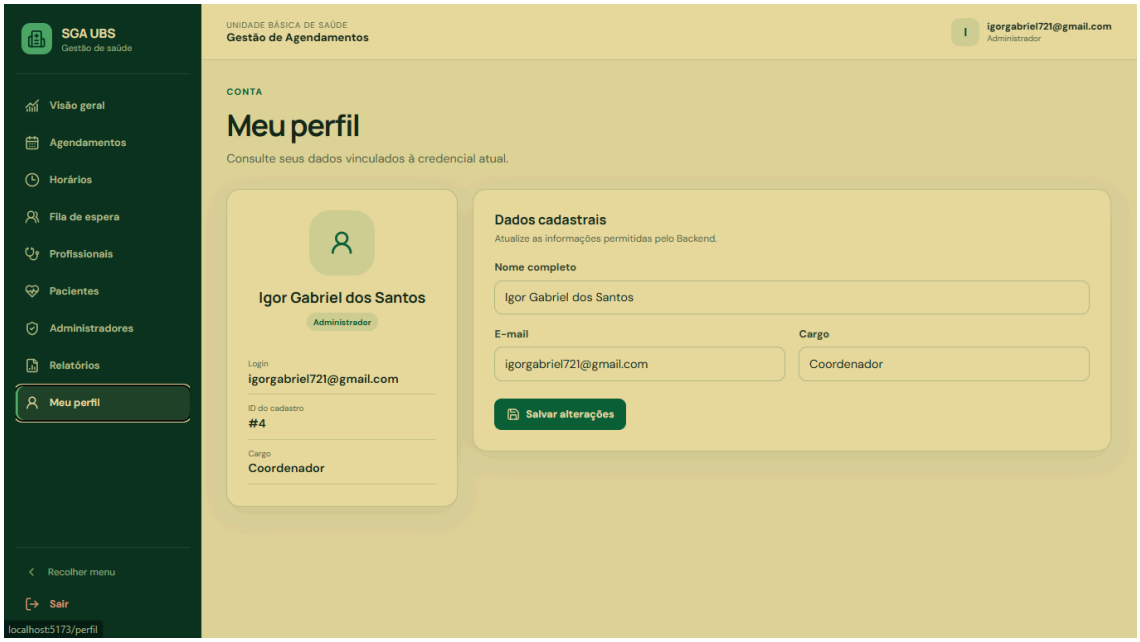
Figura 14 - Tela de Relatórios



Fonte: Elaborado pelos autores (2026).

A tela de relatórios permite consultar informações gerenciais relacionadas aos atendimentos do sistema. O administrador pode utilizar filtros por período, status e profissional, além de alternar entre informações relacionadas a agendamentos e faltas. Essa funcionalidade contribui para o acompanhamento das atividades da unidade e para a análise dos dados registrados.

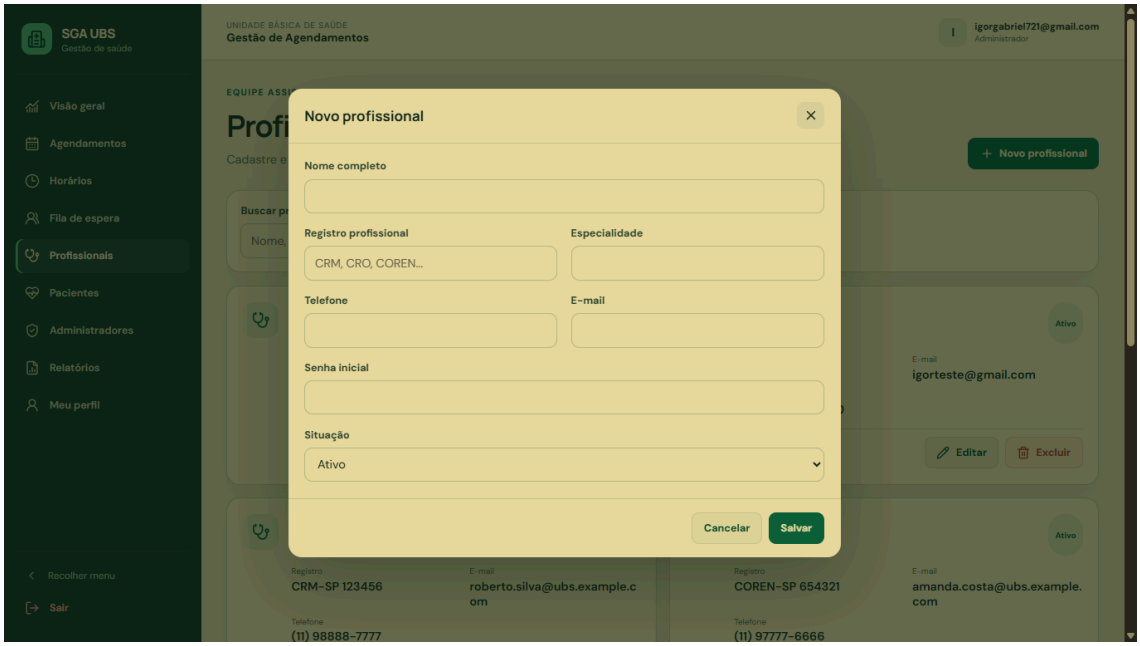
Figura 15 - Tela de Perfil do Usuário



Fonte: Elaborado pelos autores (2026).

A tela de perfil apresenta os dados associados ao usuário autenticado. Nela são exibidas informações como nome, e-mail e cargo, além da identificação do perfil de acesso. A interface permite que determinados dados cadastrais sejam atualizados pelo próprio usuário.

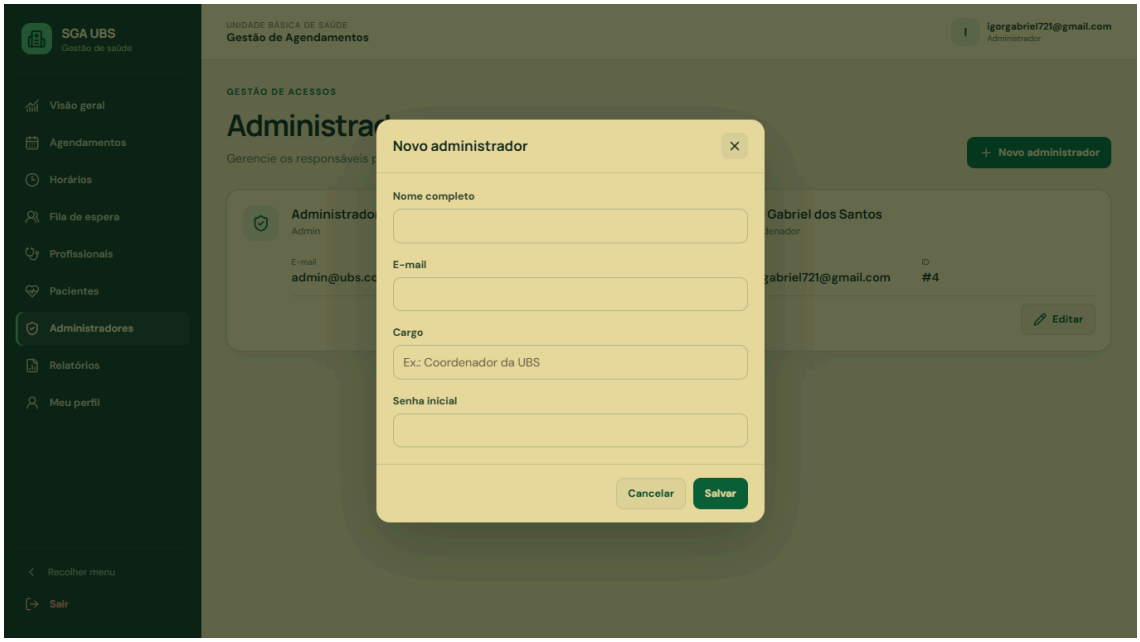
Figura 16 - Tela de Cadastro de Profissional



Fonte: Elaborado pelos autores (2026).

A tela de cadastro de profissional é apresentada em formato de formulário e permite ao administrador registrar novos profissionais de saúde no sistema. São solicitadas informações como nome, registro profissional, especialidade, telefone, e-mail, senha inicial e situação do cadastro. Dessa forma, o profissional passa a integrar a base de usuários e pode ser associado aos horários de atendimento.

Figura 17 - Tela de Cadastro de Administrador



Fonte: Elaborado pelos autores (2026).

A tela de cadastro de administrador permite a criação de novos usuários com perfil administrativo. O formulário solicita informações como nome, e-mail, cargo e senha inicial. Esse recurso possibilita ampliar a equipe responsável pela gestão do sistema, mantendo o controle de acesso por perfil de usuário.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento do Sistema de Gestão de Agendamentos para Unidades Básicas de Saúde (SGA UBS), com o objetivo primordial de mitigar os gargalos históricos relacionados à marcação de consultas no âmbito do Sistema Único de Saúde (SUS). Por meio da concepção e implementação de uma plataforma digital, foi possível atingir a meta de digitalizar e organizar o fluxo de atendimento, oferecendo uma solução prática e resolutiva para pacientes, profissionais de saúde e gestores.

Do ponto de vista técnico, a adoção de uma arquitetura cliente-servidor, utilizando React no frontend e o micro-framework Flask no backend, demonstrou-se altamente adequada para suprir os requisitos de usabilidade, leveza e manutenibilidade. A integração com o banco de dados relacional PostgreSQL garantiu a segurança e a integridade das informações, permitindo a aplicação rigorosa de regras de negócio fundamentais. Destacam-se, nesse contexto, a prevenção de agendamentos conflitantes por meio de controle de concorrência e a gestão automatizada da fila de espera, que respeita critérios de prioridade (como idade avançada e gestação). Além disso, os testes de carga forneceram evidências do comportamento da API sob diferentes níveis de demanda e confirmaram o funcionamento do controle de concorrência.

Os resultados obtidos evidenciam que a adoção de sistemas de informação em saúde, quando projetados especificamente para a gestão do acesso, pode gerar impactos positivos diretos na administração pública. O SGA UBS não apenas minimiza a dependência de processos manuais e a formação de filas presenciais nas madrugadas, mas também mune a gestão da UBS com indicadores métricos essenciais. Através do monitoramento de taxas de ocupação e histórico de absenteísmo (faltas), a ferramenta viabiliza tomadas de decisão baseadas em dados, otimizando a alocação de recursos físicos e humanos.

Embora o sistema desenvolvido cumpra com êxito os objetivos estabelecidos, servindo como uma prova de conceito madura e funcional, existem oportunidades valiosas para sua expansão. Como propostas para trabalhos futuros, recomenda-se: a integração do sistema com bases de dados governamentais (como o prontuário eletrônico do e-SUS APS e o CadÚnico); o desenvolvimento de um módulo de notificações automatizadas via SMS ou WhatsApp para reforçar o lembrete de consultas e diminuir ainda mais as taxas de ausência; e a expansão do frontend para uma aplicação móvel (mobile) nativa, ampliando a acessibilidade para a população.

Conclui-se, portanto, que o projeto entregou uma solução tecnológica robusta, alinhada às demandas da Atenção Primária à Saúde, e consolidou na prática os

conhecimentos acadêmicos adquiridos ao longo do curso de Tecnologia em Sistemas para Internet. O SGA UBS apresenta-se como um passo promissor rumo à modernização das Unidades Básicas de Saúde, utilizando a tecnologia como instrumento de democratização e eficiência no acesso à saúde pública.

REFERÊNCIAS

AGÊNCIA BRASIL. **SUS: 76% da população brasileira têm atendimento gratuito de saúde.** Brasília, 19 set. 2025. Disponível em: https://webradioamnesia.com/noticia/1869923/sus-76-da-populacao-tem-atendimento-direto-gratuito-de-saude. Acesso em: 6 jul. 2026.

BIBLIOTECA VIRTUAL EM SAÚDE – MINISTÉRIO DA SAÚDE. **71% dos brasileiros têm os serviços públicos de saúde como referência.** Brasília, 2025. Disponível em: https://bvsms.saude.gov.br/71-dos-brasileiros-tem-os-servicos-publicos-de-saude-como-referencia/. Acesso em: 6 jul. 2026.

CARVALHO, Davi. **Censo das UBS revela avanços e desigualdades na Atenção Primária à Saúde no Brasil.** Plataforma Região e Redes, 2025. Disponível em: https://www.regiaoeredes.org.br/censo-das-ubs-revela-avancos-e-desigualdades-na-atencao-primaria-a-saude-no-brasil/. Acesso em: 6 jul. 2026.

CATRACA LIVRE. **84% da população brasileira depende exclusivamente do SUS**. São Paulo, 14 abr. 2025. Disponível em: https://catracalivre.com.br/saude-bem-estar/84-da-populacao-brasileira-depende-exclusivamente-do-sus/. Acesso em: 6 jul. 2026.

MINISTÉRIO DA SAÚDE. **Mais de 87% das Unidades Básicas de Saúde utilizam prontuário eletrônico e quase a totalidade tem acesso à internet.** Brasília, 18 jun. 2025. Disponível em: https://www.gov.br/saude/pt-br/assuntos/noticias/2025/junho/mais-de-87-das-unidades-basicas-de-saude-utilizam-prontuario-eletronico-e-quase-a-totalidade-tem-acesso-a-internet. Acesso em: 6 jul. 2026.

PALLETS PROJECTS. *Flask Documentation*. Disponível em: https://flask.palletsprojects.com/en/stable/. Acesso em: 23 jul. 2026.

META PLATFORMS, INC. *React Documentation*. Disponível em: https://react.dev/. Acesso em: 23 jul. 2026.

GITHUB. **Documentos do GitHub**. Disponível em: https://docs.github.com/pt. Acesso em: 22 jul. 2026.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL Documentation*. Disponível em: https://www.postgresql.org/docs/. Acesso em: 23 jul. 2026.

PYTHON SOFTWARE FOUNDATION. *Python Documentation*. Disponível em: https://docs.python.org/3/. Acesso em: 23 jul. 2026.

SQLALCHEMY AUTHORS. *SQLAlchemy Documentation*. Disponível em: https://docs.sqlalchemy.org/. Acesso em: 23 jul. 2026.

JONES, M.; BRADLEY, J.; SAKIMURA, N. *JSON Web Token (JWT)*. RFC 7519. Internet Engineering Task Force (IETF), 2015. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 23 jul. 2026.