UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE – CCENS
DEPARTAMENTO DA COMPUTAÇÃO

DAVID PAPA DE SOUZA

SISTEMA DE PRONTUÁRIO MÉDICO ELETRÔNICO PARA UTILIZAÇÃO EM INSTITUIÇÕES DE ENSINO

ALEGRE – ES
AGOSTO DE 2025

DAVID PAPA DE SOUZA

SISTEMA DE PRONTUÁRIO MÉDICO ELETRÔNICO PARA UTILIZAÇÃO EM INSTITUIÇÕES DE ENSINO

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como parte das exigências para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Antonio Almeida de Barros Júnior.

Universidade Federal do Espírito Santo – UFES
Departamento de computação

ALEGRE – ES
AGOSTO DE 2025

David Papa de Souza

SISTEMA DE PRONTUÁRIO MÉDICO ELETRÔNICO PARA UTILIZAÇÃO EM INSTITUIÇÕES DE ENSINO

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em 27 de agosto de 2025.

COMISSÃO EXAMINADORA

Documento assinado digitalmente
ANTONIO ALMEIDA DE BARROS JUNIOR
Data: 28/08/2025 14:03:09-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. Antonio Almeida de Barros Junior
Universidade Federal do Espírito Santo
Orientador

Documento assinado digitalmente
VALERIA ALVES DA SILVA
Data: 28/08/2025 13:34:38-0300
Verifique em https://validar.iti.gov.br

_____

Prof.ª Dra. Valeria Alves da Silva
Universidade Federal do Espírito Santo

Documento assinado digitalmente
BRUNO VILELA OLIVEIRA
Data: 27/08/2025 19:52:29-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. Bruno Vilela Oliveira
Universidade Federal do Espírito Santo

RESUMO

O presente trabalho teve como objetivo desenvolver um sistema web de prontuário eletrônico médico voltado para a Seção de Atenção à Saúde e Assistência Social (SASAS) da UFES, com a finalidade de suprir a ausência de um recurso informatizado para gestão de dados clínicos em instituições de ensino. O sistema foi elaborado a partir de entrevistas exploratórias realizadas com o médico do setor e da análise comparativa de softwares semelhantes, sendo modelado com técnicas de engenharia de *software* e implementado com as tecnologias Angular, Spring Boot e PostgreSQL. A validação concentrou-se em testes técnicos de desempenho e qualidade, os quais indicaram tempo de resposta médio inferior a dois segundos para operações de cadastro e consulta, ausência de falhas em cem execuções consecutivas de operações críticas e consistência de 100% nos dados armazenados. Os resultados evidenciaram a robustez da aplicação, seu alinhamento com os requisitos levantados e a viabilidade prática de sua adoção. Como limitações, destacam-se a ausência de testes com usuários finais, a falta de integração com outros sistemas institucionais e a impossibilidade de comparações práticas com soluções de mercado. Ainda assim, a contribuição técnica deste trabalho consiste na entrega de um *software* funcional e adaptado ao contexto acadêmico, enquanto a contribuição acadêmica reside no aprendizado científico, no método de levantamento de requisitos e na análise crítica de sistemas de gestão em saúde.

Palavras-chave: Saúde; Prontuário Eletrônico; Sistema Web; Gestão da Informação; Instituições de Ensino.

LISTA DE QUADROS

QUADRO 1 - DESCRIÇÃO FUNCIONALIDADE DOS PACOTES DO PROJETO *BACK-END*. 43
QUADRO 2 - DESCRIÇÃO FUNCIONALIDADE DOS PACOTES DO PROJETO *FRONT-END*. 45
QUADRO 3 - DEFEITOS ENCONTRADOS E SOLUÇÕES APLICADAS. 50

LISTA DE FIGURAS

Figura 1 - Interface inicial Meditech ERH. 15
Figura 2 - Tela atendimento *web* Ninsaúde. 16
Figura 3 - Portal e Pagina Inicial do Betha Saúde. 17
Figura 4 – Arquitetura do Sistema. 25
Figura 5 – Representação do Kanban. 27
Figura 6 – Representação de um caso de uso. 28
Figura 7 – Representação de um modelo em um diagrama de classes UML. 33
Figura 8 - Diagrama de classes do sistema. 34
Figura 9 - Diagrama de casos de uso. 35
Figura 10 - Modelo físico de banco de dados. 39
Figura 11 - Representação quadro de tarefas do Trello 42
Figura 12 - Representação da estrutura de pacotes do projeto *back-end*. 43
Figura 13 - Representação da estrutura de pacotes do projeto *front-end*. 45
Figura 14 - Conexão com o banco de dados. 47
Figura 15 - Configuração de Segurança da API no *Spring Security*. 48
Figura 16 - Senhas criptografadas de usuários. 48
Figura 17 - Configuração de rotas e *guards* do Angular. 49
Figura 18 – Testes unitários do serviço de AgendaConsulta. 52
Figura 19 – Execução de testes bem-sucedida. 53
Figura 20 – Falha na execução de testes. 53
Figura 21 – Testes de integração via *postman*. Autenticação de usuários. 54
Figura 22 – Execução de teste de API Consultar Pessoas. 55
Figura 23 – Página de autenticação do sistema. 58
Figura 24 – Consulta de agendamentos. 59
Figura 25 – Cadastro de agendamento. 60
Figura 26 – Cadastro de pacientes. 60
Figura 27 – Edição e complemento de cadastro do paciente. 61
Figura 28 – Consulta de pacientes. 62
Figura 29 – Informações do usuário autenticado. 62
Figura 30 – Tela de alteração de senha de um usuário autenticado. 63
Figura 31 – Tela inicial usuário perfil administrativo. 64
Figura 32 – Tela cadastro usuário administrativo. 65
Figura 33 - Tela cadastro usuário administrativo. 65
Figura 34 – Tela de consulta e inativação de usuários. 66
Figura 35 – Tela de cadastro de médico. 67

Figura 36 – Tela edição e complemento de dados médico 67
Figura 37 – Tela de consulta de médicos 68
Figura 38 – Tela inicial usuário autenticado como médico. 69
Figura 39 – Tela de cadastro de consulta, etapa dados paciente. 70
Figura 40 – Tela de cadastro de consulta, selecionando um paciente. 71
Figura 41 – Tela de cadastro de anamnese. 71
Figura 42 – Tela de cadastro de diagnóstico 72
Figura 43 – Cadastro de CIDs. 73
Figura 44 – Cadastro de alergias 73
Figura 45 – Cadastro de prescrições médicas. 74
Figura 46 – Cadastro de medicamentos. 75
Figura 47 – Cadastro de tratamentos. 75
Figura 48 – Solicitação de exames da consulta 76
Figura 49 – Tela de *upload* de arquivos referentes a consulta. 77
Figura 50 – Tela de resumo da consulta 77
Figura 51 – Tela de consulta de agendamentos, botão gerar consulta. 78

LISTA DE TABELAS

Tabela 1 – Comparação entre o Sistema Ninsaúde e o Sistema Proposto. 18
Tabela 2 - Tabela Pessoa. 40
Tabela 3 - Tabela Agenda. 95
Tabela 4 - Tabela AgendaConsulta. 95
Tabela 5 - Tabela AlergiaPaciente. 95
Tabela 6 - Tabela Arquivo. 95
Tabela 7 - Tabela CID. 96
Tabela 8- Tabela Consulta. 96
Tabela 9 - Tabela Contato. 96
Tabela 10 - Tabela Diagnostico. 97
Tabela 11 - Tabela Exame. 97
Tabela 12 - Tabela Endereco. 97
Tabela 13 - Tabela Medico. 98
Tabela 14 – Tabela Paciente. 98
Tabela 15 – Tabela Prescricao. 98
Tabela 16 - Tabela PrescricaoConsultaMedicamento. 99
Tabela 17 – Tabela Tratamento. 99
Tabela 18 - Tabela Usuário. 99

SUMÁRIO

1 INTRODUÇÃO 8

1.1 O PROBLEMA E SUA IMPORTÂNCIA 9

1.2 OBJETIVOS 10

1.2.1 Objetivo geral 10

1.2.2 Objetivos específicos 10

2 REVISÃO DA LITERATURA 12

2.1 SISTEMAS DE INFORMAÇÃO 12

2.2 O USO DE SISTEMAS DE INFORMAÇÃO NA SAÚDE 13

2.3 SISTEMAS SEMELHANTES 14

2.3.1 Meditech ERH Software 14

2.3.2 Ninsaúde Clinic 15

2.3.3 Betha Saúde 16

2.3.4 Diferenças dos softwares citados com o proposto neste projeto 17

2.4 TENCOLOGIAS EMPREGADAS 18

2.4.1 Linguagens e recursos de programação 19

2.4.2 Frameworks 20

2.4.2.1 Spring *Framework* 20

2.4.2.2 Angular 21

2.4.2.3 Bootstrap 21

2.4.3 Sistema Gerenciador de Banco de Dados (SGDB) 21

2.4.4 Ferramentas CASE 22

2.4.4.1 Astah UML 22

2.4.4.2 Trello 23

2.5 ARQUITETURA DO SISTEMA 23

2.5.1 API 23

2.5.2 REST 24

2.5.3 Desenho da Arquitetura 25

2.5.4 MVC 25

2.6 KANBAN 26

2.7 CASOS DE USO 27

3 METODOLOGIA 29

3.1 LEVANTAMENTO E ANÁLISE DE REQUISITOS 29

3.1.1 Requisitos funcionais 30

3.1.2 Requisitos não funcionais 32

3.1.3 Diagrama de classes 33

3.1.4 Diagrama de caso de uso *34*
3.1.5 Narrativas de casos de uso *35*
3.2 MODELAGEM DE BANCO DE DADOS 38
3.2.1 Modelo físico de dados *38*
3.2.2 Dicionário de dados *40*
4 DESENVOLVIMENTO DA APLICAÇÃO 41
4.1 ORGANIZAÇÃO DOS PROCESSOS 41
4.2 A APLICAÇÃO 42
4.2.1 Estrutura do projeto *42*
4.2.2 Conexão com o banco de dados *46*
4.2.3 Segurança do sistema e autenticação dos usuários *47*
4.2.4 Rotas de navegação *49*
5 GARANTIA DE QUALIDADE E PROCESSOS DE TESTES 50
5.1 TESTES DE DESENVOLVIMENTO 50
5.2 TESTES UNITÁRIOS 51
5.3 TESTES DE INTEGRAÇÃO 53
5.4 IMPACTO E RESULTADO DOS TESTES NO SISTEMA 55
5.5 CRITÉRIOS DE ANÁLISE DOS RESULTADOS 56
6 RESULTADOS ALCANÇADOS 57
6.1.1 Telas do sistema *58*
6.1.2 Fluxo comum aos perfis de usuários *58*
6.1.3 Fluxo exclusivo administrativo *64*
6.1.4 Fluxo exclusivo médico *68*
7 CONCLUSÕES E CONSIDERAÇÕES FINAIS 79
8 CRONOGRAMA 81
APÊNDICE A - NARRATIVAS DE CASOS DE USO 86
APENDICE B - DESCRIÇÃO DAS TABELAS DO BANCO DE DADOS 95

1 INTRODUÇÃO

A saúde pode ser interpretada como a ausência de doenças e, muito além disso, é definida pela Organização Mundial de Saúde (OMS) como uma condição plena de bem-estar físico, mental e social. Sendo ela um direito e um dever assegurado independentemente de questões políticas, sociais e culturais que abrangem a sociedade como um todo (MEC/SEF, 1998). Neste contexto, o Brasil, desde a Constituição de 1988, garante que a saúde deve ser orquestrada pelo Estado e é um direito de todos os cidadãos de maneira igualitária, visando ações que impeçam o agravamento e facilitem a proteção e a recuperação da saúde de um indivíduo (BRASIL, 1988).

Ainda segundo a Constituição brasileira, de acordo com o artigo 6º da Portaria nº 1.820, de 13 de agosto de 2009, para que o indivíduo possa ter acesso aos cuidados assegurados por lei, ele deverá prestar informações apropriadas para a realização do seu atendimento. Diante disso, o assistido apresentará suas queixas e demais informações sobre seu estado de saúde, bem como seu histórico médico, explicitando o uso de medicamentos e/ou enfermidades e hospitalizações anteriores.

Com o advento da tecnologia da informação, torna-se cada vez mais importante que instituições médicas se mobilizem para que o atendimento prestado aos pacientes seja o mais adequado possível. Uma das ferramentas mais utilizadas por profissionais da saúde de todo o mundo é o prontuário médico. O prontuário é um registro documental criado pelo profissional de saúde e contém, de maneira organizada e resumida, todas as informações pertinentes ao paciente, como histórico familiar, anamnese, descrição e progressão de sintomas, resultados de exames, orientações de tratamento e prescrições. Seu propósito principal é facilitar a continuidade do cuidado ao paciente, fornecendo um registro completo e acessível das informações médicas relevantes (FARINA, 1999).

Apesar da relevância, muitos prontuários ainda são manuscritos, o que gera dificuldades de consulta, risco de perda de informações e limitações na gestão dos dados. Estudos apontam que registros em papel podem apresentar problemas como ilegibilidade, fragmentação das informações e dificuldade de compartilhamento entre os profissionais de saúde, impactando diretamente a qualidade do atendimento (CUNHA; CUNHA, 2009). Esses aspectos evidenciam a necessidade de informatização para garantir maior agilidade, segurança e acessibilidade às informações médicas.

Nesse contexto, destaca-se a Seção de Atenção à Saúde e Assistência Social (SASAS) da UFES, que oferece diversos serviços voltados à comunidade acadêmica, como Psicologia, Odontologia, Enfermagem, Serviço Social e Serviço Médico. Nos atendimentos médicos, são realizadas consultas com prescrições de medicamentos, solicitações de exames, perícia médica e encaminhamentos para profissionais da rede pública de saúde quando necessário. Contudo, observa-se que a ausência de um prontuário eletrônico integrado dificulta a centralização das informações dos pacientes, levando à redundância na coleta de dados, à lentidão na recuperação de informações e, por vezes, à repetição de procedimentos já realizados. Assim, a ausência de informatização no SASAS/UFES constitui um problema relevante, que justifica a necessidade de pesquisas e propostas para a implementação de sistemas de prontuário eletrônico capazes de otimizar a gestão das informações em saúde e aprimorar a qualidade dos atendimentos prestados.

1.1 O PROBLEMA E SUA IMPORTÂNCIA

Tendo em vista que a importância de um bom atendimento passa diretamente pelas informações colhidas nos relatos dos pacientes, bem como o acompanhamento de uma série de consultas realizadas pelo mesmo, torna-se imprescindível que tais informações sejam mantidas em segurança de modo que facilite seu acesso e manutenção.

Ter um histórico de exames, bem como o acompanhamento de medicamentos já utilizados e enfermidades anteriores, pode ser um enorme diferencial para o tratamento de doenças que possam surgir futuramente. Além disso, pessoas com doenças crônicas devem ter sua atenção dobrada para possíveis evoluções de quadros clínicos, tornando necessário o acompanhamento e atualização de um histórico detalhado de saúde. De acordo com um estudo feito pela Pesquisa Nacional de Saúde (PNS) em 2019, cerca de 52% dos brasileiros com mais de 18 anos receberam o diagnóstico de ao menos uma doença crônica (GLOBAL CARE, 2020).

Entretanto, sabe-se que muitas vezes o processo de armazenamento e gestão como um todo de documentos gerados a partir desse acompanhamento, pode ser uma tarefa trabalhosa e que demanda um elevado grau de organização quando feito pelo manuseio de papéis, documentos ou outros artefatos físicos. Documentos esses, que podem variar de resultados de exames, encaminhamentos médicos, atestados, receituários, entre outros. Ainda assim, ter o

controle dessas informações é de suma importância e garantirá que sua saúde e qualidade de vida não venham sofrer impactos significativos repentinamente.

Uma vez destacada a importância dos dados de um paciente, é fundamental que instituições de saúde no âmbito de ensino, estejam munidas de sistemas informatizados que possam facilitar o gerenciamento dos relatos dos pacientes.

1.2 OBJETIVOS

Serão apresentados nesta seção os objetivos gerais e específicos que determinarão as diretrizes de desenvolvimento do presente trabalho.

1.2.1 Objetivo geral

O objetivo principal deste trabalho foi o desenvolvimento de um sistema web de prontuário médico eletrônico destinado ao uso dos profissionais de saúde da Seção de Atenção à Saúde e Assistência Social (SASAS/UFES), possibilitando o gerenciamento integrado dos dados de saúde de cada indivíduo atendido. O sistema foi desenvolvido para armazenar e organizar informações de forma estruturada, permitindo o acompanhamento clínico, contribuindo para diagnósticos mais precisos e aumentando a confiabilidade e eficiência dos serviços prestados.

1.2.2 Objetivos específicos

Conforme os objetivos gerais especificados, esboçamos os objetivos específicos da seguinte maneira:

- Desenvolvimento de um sistema *web* para prontuário médico, caracterizado por uma interface intuitiva e de fácil acesso por meio de navegadores de internet;

- Implementação de mecanismos de gerenciamento seguro e eficiente dos dados pessoais e de saúde dos pacientes;
- Gerenciamento das características e dados pessoais do paciente;
- Gerenciamento dos dados profissionais e pessoais dos médicos cadastrados no sistema;
- Acompanhamento de informações pertinentes ao histórico médico de um determinado indivíduo, incluindo informações sobre doenças, exames e consultas médicas já realizadas, bem como o cadastro de novos que poderão ser realizados;
- Gerenciamento e monitoramento de necessidades gerais de um determinado indivíduo, permitindo o controle de medicamentos utilizados e tratamentos médicos que devem ser seguidos;
- Gerenciamento de artefatos gerados a partir de consultas e exames já realizados, sendo, receituários de medicamentos, atestados médicos, resultados de exames, entre outros;
- Avaliação dos impactos da utilização do sistema em relação à agilidade no acesso às informações e à redução de redundâncias no processo de atendimento.

2 REVISÃO DA LITERATURA

Neste capítulo apresenta-se o referencial teórico utilizado para a construção deste Trabalho de Conclusão de Curso (TCC). Nas seções seguintes, são discutidos aspectos relacionados a sistemas de informação e suas aplicações na área da saúde, bem como plataformas semelhantes já disponíveis no mercado. Por fim, são descritas as linguagens e ferramentas utilizadas, assim como os motivos que fundamentaram sua escolha.

2.1 SISTEMAS DE INFORMAÇÃO

Segundo Mattos (2017), sistema de informação é um sistema qualificado no processamento e comunicação de dados ou de informações. É constituído por um conjunto de componentes físicos e lógicos que se comunicam entre si. As relações entre esses componentes lógicos e físicos se dão por meio de aplicações executadas pelo sistema de informação. As pessoas também fazem parte do sistema, de nada adianta o emprego de técnicas e equipamentos se as pessoas que devem, não aceitarem ou não os usarem adequadamente.

Para Rezende (2006), os sistemas de informação, seja qual for seu nível ou classificação, tem como objetivo auxiliar no processo de tomada de decisões nas organizações. Deste modo, é necessário um planejamento adequado e que os sistemas ofereçam a qualidade necessária para atender características inerentes de organizações que necessitam empregá-lo. Tais características destacam-se o grande volume de dados e informações, complexidade dos processos, o extenso contexto das organizações, o ambiente mutável e dinâmico, diversas integrações de sistemas e tecnologias, suporte à tomada de decisões e a variedade de usuários e clientes envolvidos.

De forma geral, sistemas de informações consistem em um conjunto integrado de recursos, incluindo *hardware*, *software*, dados, redes e recursos humanos, com o objetivo de gerar, armazenar e processar informações resultantes da interação desses elementos.

2.2 O USO DE SISTEMAS DE INFORMAÇÃO NA SAÚDE

Administrar um serviço de saúde envolve gerenciar tanto aspectos organizacionais quanto operacionais, semelhante ao que acontece em qualquer outra instituição. Isso significa que os gestores de saúde precisam lidar com tarefas administrativas, como controle de estoque, gerenciamento financeiro e recursos humanos, que impactam diretamente a estrutura e o funcionamento dos serviços.

Paralelamente, é fundamental lidar com os elementos resultantes do atendimento direto ao paciente. Nesse contexto, os Sistemas de Informação em Saúde (SIS) assumem papel essencial, pois permitem registrar e organizar dados clínicos e epidemiológicos, fornecendo informações sobre o estado de saúde do paciente, histórico médico, fatores ambientais e sociais que influenciam o processo saúde-doença (CARVALHO; EDUARDO, 1998).

De acordo com os mesmos autores, os SIS também possibilitam compreender as características sociais, econômicas, físicas e demográficas que influenciam a saúde de uma população, bem como identificar problemas percebidos tanto pelas autoridades de saúde quanto pela própria comunidade. Além disso, abrangem informações administrativas, servindo de suporte para decisões estratégicas e gestão de recursos.

Mais recentemente, a literatura aponta que os SIS evoluíram para contemplar não apenas a gestão administrativa, mas também a qualidade assistencial, permitindo maior rastreabilidade, segurança na prescrição e continuidade do cuidado (CUNHA; CUNHA, 2009; SILVA et al., 2018). Isso demonstra que a informatização dos serviços de saúde tem impacto direto na eficiência e na precisão do atendimento, reduzindo erros médicos e perdas de informações.

Entretanto, nota-se que grande parte das soluções existentes são desenvolvidas para hospitais e clínicas privadas, com foco em integração de processos administrativos complexos, como faturamento e gestão financeira. Pouco se encontra, na literatura e na prática, sobre sistemas de prontuário eletrônico voltados especificamente para instituições de ensino que prestam atendimento básico à comunidade acadêmica. Essa lacuna é particularmente relevante no caso do SASAS/UFES, onde o registro ainda é predominantemente manual e não há uma ferramenta informatizada adaptada ao contexto acadêmico.

Assim, ao mesmo tempo em que a literatura confirma a importância dos sistemas de informação em saúde, ela também evidencia a ausência de soluções direcionadas a ambientes

educacionais. Foi justamente nesse ponto que este trabalho se inseriu: desenvolveu-se um sistema informatizado de prontuário médico, adequado ao contexto universitário, que buscou superar as limitações dos sistemas existentes e atender às necessidades específicas de uma instituição pública de ensino.

2.3 SISTEMAS SEMELHANTES

Nesta seção são apresentados alguns sistemas já existentes que se assemelham com a proposta desenvolvida neste trabalho.

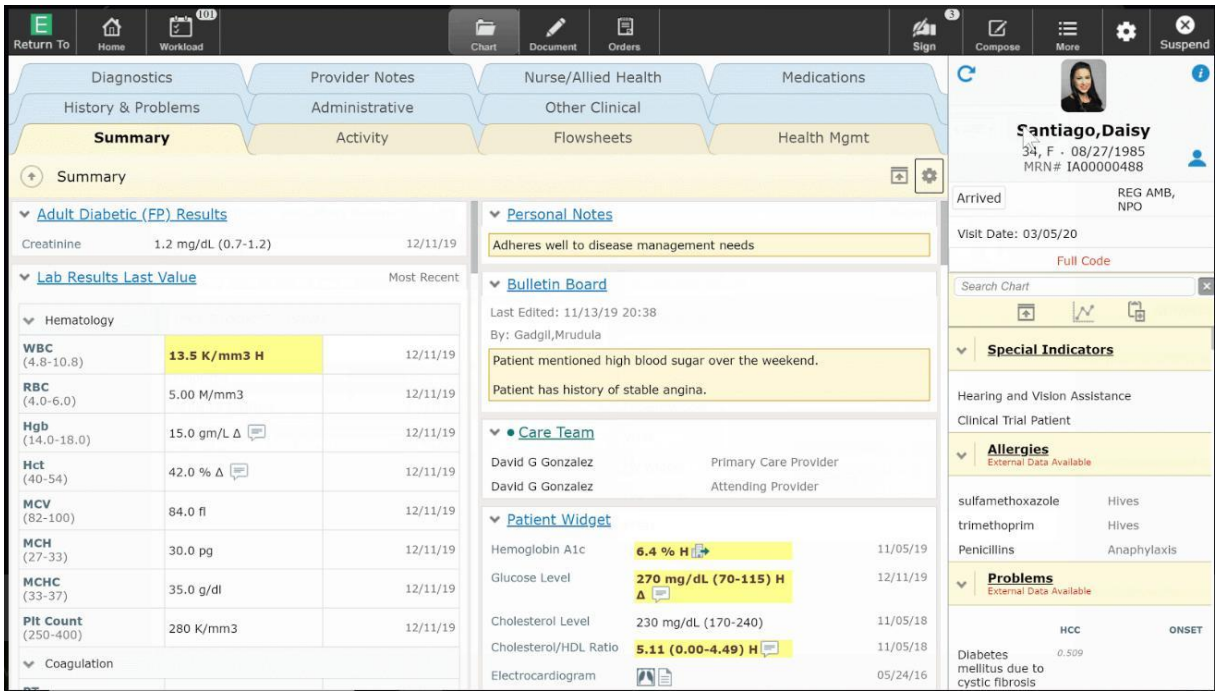
2.3.1 Meditech ERH Software

O Meditech oferece uma variedade de soluções de registro eletrônico de saúde para hospitais e sistemas de saúde, incluindo módulos para registros clínicos, prescrição eletrônica, gestão de pedidos e resultados de laboratório.

Os sistemas da Meditech são projetados para fornecer uma ampla gama de funcionalidades para apoiar as operações clínicas e administrativas de uma instituição de saúde. Isso inclui módulos para registros médicos eletrônicos (EMR), prescrição eletrônica, gestão de pedidos de laboratório e radiologia, gerenciamento de resultados de exames, agendamento de pacientes, faturamento e codificação, entre outros (MEDITECH, 2025).

A Figura 1 apresenta a interface inicial do aplicativo Meditech, com áreas destinadas a diagnósticos, medicamentos e outros aspectos relativos ao atendimento médico.

Figura 1 - Interface inicial Meditech ERH.



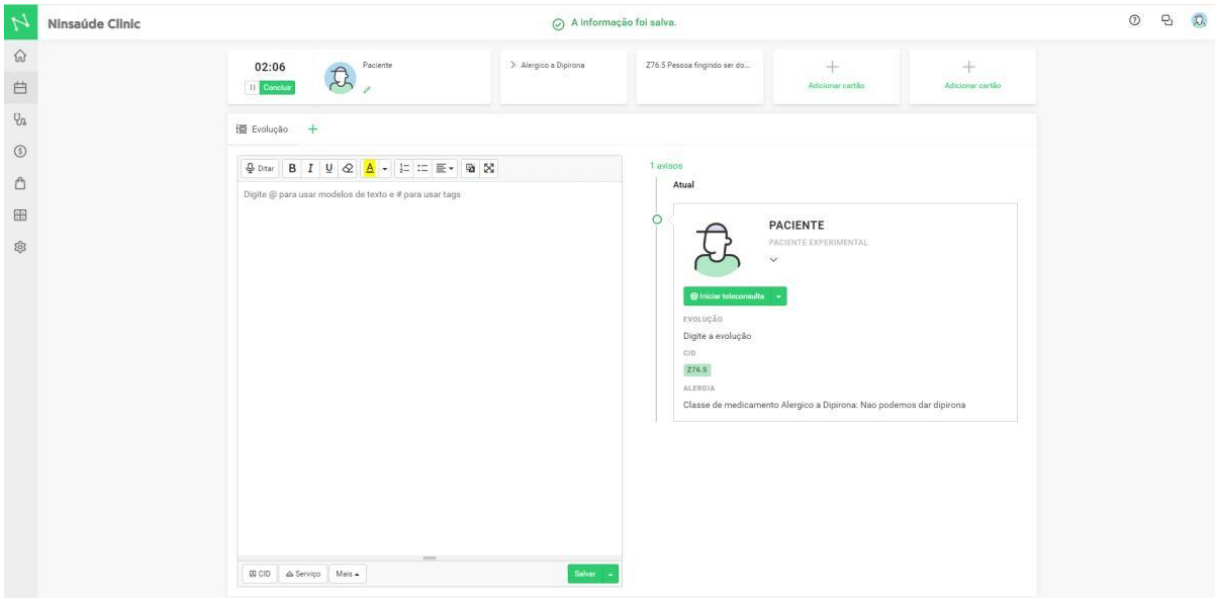
Fonte: Meditech (2025).

2.3.2 Ninsaúde Clinic

O Ninsaúde é um *software* médico desenvolvido para clínicas e consultórios que oferece um conjunto abrangente de ferramentas para facilitar o gerenciamento das atividades clínicas e administrativas. Ele inclui recursos como prontuário eletrônico do paciente, agendamento de consultas, prescrição eletrônica, gestão financeira e relatórios de desempenho. Como uma solução baseada na nuvem, o Ninsaúde Apolo permite acesso seguro aos dados de qualquer lugar com uma conexão à internet, o que aumenta a flexibilidade e a eficiência no atendimento ao paciente. Além disso, o *software* é projetado para ser intuitivo e fácil de usar, visando melhorar a produtividade dos profissionais de saúde e proporcionar uma experiência positiva para os pacientes (NINSAUDE, 2025).

A Figura 2 mostra a interface inicial do portal *web* da aplicação Ninsaúde. Com recursos voltados a uma experiencia de uso mais intuitiva, observa-se a simulação de um atendimento a um paciente, bem como a atribuição de possíveis doenças e alergias que ele possui.

Figura 2 - Tela atendimento *web* Ninsaúde.



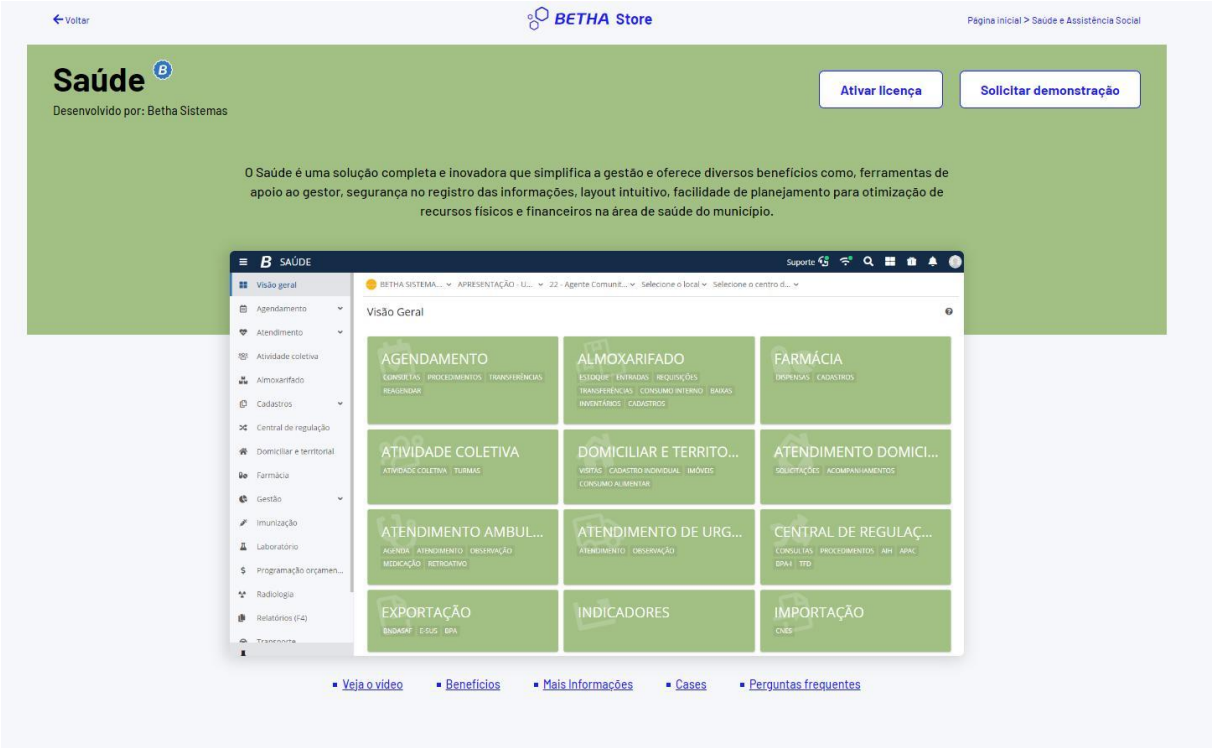
Fonte: Ninsaúde Clinic (2025).

2.3.3 Betha Saúde

A Betha é uma empresa brasileira que oferece sistemas de gestão para diversas áreas, incluindo saúde. Seu sistema de prontuário médico eletrônico oferece funcionalidades para o registro e o acompanhamento do histórico clínico dos pacientes, prescrição de medicamentos, agendamento de consultas e gestão de recursos hospitalares (BETHA, 2025).

A Figura 3 mostra o portal e a interface inicial do sistema de gestão de saúde da Betha Saúde. Podemos observar os diversos módulos disponíveis no sistema.

Figura 3 - Portal e Pagina Inicial do Betha Saúde.



Fonte: Betha Saúde (2025).

2.3.4 Diferenças dos *softwares* citados com o proposto neste projeto

A escolha dos sistemas Meditech, Betha Saúde e Ninsaúde para esta análise comparativa se justifica pelo fato de todos eles possuírem módulos consolidados de prontuário eletrônico do paciente, o que os torna relevantes para o escopo deste trabalho. No entanto, embora representem soluções reconhecidas no mercado e utilizadas em diferentes contextos da saúde, tais sistemas foram concebidos principalmente para hospitais e clínicas privadas, não sendo necessariamente adequados para instituições de ensino.

Esses *softwares* priorizam integrações administrativas e processos financeiros complexos, além de apresentarem custos de implantação e manutenção elevados, aspectos que não se alinham às necessidades de setores de saúde universitários, cujo foco está no atendimento básico à comunidade acadêmica. Dessa forma, ainda que sejam exemplos de referência na informatização de serviços de saúde, evidenciam uma lacuna: a ausência de soluções desenvolvidas especificamente para a realidade de instituições de ensino públicas.

O sistema proposto neste trabalho busca justamente suprir essa carência, oferecendo uma ferramenta de baixo custo, simples, acessível e direcionada ao gerenciamento de prontuários médicos em ambiente acadêmico, sem sobrecarga de funcionalidades administrativas que não fazem parte das rotinas do SASAS/UFES.

A Tabela 1 apresenta um comparativo entre as funcionalidades do prontuário eletrônico do sistema Ninsaúde Clinic e do sistema proposto neste trabalho. A análise foca exclusivamente nas funcionalidades relacionadas ao prontuário médico.

Tabela 1 – Comparação entre o Sistema Ninsaúde e o Sistema Proposto.

Funcionalidade	**Ninsaúde Clinic**	**Sistema Proposto**
Customização por Especialidade	Sim	Não
Gestão de Imagens/Exames	Sim	Sim
Telemedicina	Sim	Não
Gestão de Alergias	Sim	Sim
Gestão de Pacientes	Sim	Sim
Acesso Multidispositivo	Sim	Sim
Gestão de Agendamentos	Sim	Sim
Gestão de CID/Procedimentos	Sim	Sim
Integração com Inteligência Artificial	Sim	Não
Gestão de Médicos	Sim	Sim
Gestão de Medicamentos	Sim	Sim
Gratuito	Não	Sim

Fonte: o autor (2025).

2.4 TENCOLOGIAS EMPREGADAS

É crucial selecionar adequadamente as tecnologias empregadas em um projeto para garantir resultados favoráveis. Nesse sentido, foram adotadas tecnologias que se complementam e são familiares ao autor do projeto. Nas seções seguintes, serão apresentadas as tecnologias selecionadas, acompanhadas de suas respectivas características.

2.4.1 Linguagens e recursos de programação

a) JAVA - criado pela Sun Microsystems em 1995, consolidou-se como uma linguagem de programação e plataforma de computação amplamente utilizada, servindo como base para inúmeros serviços e aplicativos. (JAVA, 2025).
b) SQL - A Linguagem de consulta estruturada (SQL), surgiu em 1970 fundamentada no modelo de dados relacionais, originalmente batizada de *Structured English Query Language* (SEQUEL). Posteriormente, foi simplificada para SQL. A Oracle, anteriormente denominada *Relational Software*, foi pioneira ao apresentar o primeiro sistema comercial de gerenciamento de banco de dados relacional baseado em SQL. A SQL é uma linguagem de programação utilizada para gerenciar informações em bancos de dados relacionais, que organizam dados em tabelas com linhas e colunas. Com instruções SQL, é possível armazenar, atualizar, remover, pesquisar e recuperar dados do banco de dados, além de otimizar sua performance (AWS, 2025).
c) HTML - O HTML (*HyperText Markup Language*) é a linguagem de marcação padrão para a criação de páginas da web, responsável por estruturar conteúdos como textos, imagens, links e multimídia. Por meio de elementos (*tags*) e atributos, permite organizar e definir a hierarquia da informação, sendo a base para o funcionamento de qualquer site. Atualmente, sua versão mais recente, o HTML5, amplia recursos ao oferecer suporte nativo a áudio, vídeo e novos elementos semânticos, consolidando-se como tecnologia essencial no desenvolvimento web (MDN Web Docs, 2025).
d) CSS - O *Cascading Style Sheets* (CSS) criado em 1994 por Hakon Lie, com o objetivo de simplificar a programação de *web* sites, que na época era bastante complexa. Antes do CSS, os desenvolvedores precisavam usar uma quantidade considerável de código HTML para alcançar resultados simples, como a formatação de tabelas. O CSS é uma linguagem que determina a aparência e o layout das páginas da *web*, permitindo que os usuários criem páginas com códigos mais simples e organizados em combinação com o HTML.

e) *JavaScript* - É uma linguagem de programação usada para adicionar funcionalidades avançadas a uma página *web*. Junto com HTML e CSS, formam a base das tecnologias padrões da *web*. Com *JavaScript*, é possível criar um conteúdo dinâmico que se atualiza automaticamente, controlar multimídia, imagens animadas e uma variedade de outras interações das quais o usuário necessita (MDN, 2025).

f) *TypeScript* - É uma linguagem de programação que se baseia no *JavaScript*, porém com tipagem forte, fornecendo vantagens em todos os tamanhos de projetos. Ela introduz uma sintaxe adicional para permitir uma integração mais eficiente com seu editor, detectando erros logo no início do processo de desenvolvimento. Além disso, o *TypeScript* compreende o *JavaScript* e utiliza a inferência de tipos para oferecer excelentes ferramentas sem exigir código adicional (TYPESCRIPT, 2025).

2.4.2 Frameworks

Um *framework* em engenharia de *software* é uma coleção de componentes reutilizáveis que agilizam o desenvolvimento de novas aplicações. Os *frameworks* de *software* contêm módulos de código reutilizáveis baseados em padrões específicos, permitindo o desenvolvimento padronizado de novas aplicações e a aplicação de regras de arquitetura de *software* ou processos de negócios (AWS, 2025).

2.4.2.1 Spring *Framework*

O *Spring Framework* é uma plataforma Java que simplifica o desenvolvimento de aplicativos ao oferecer suporte abrangente à infraestrutura. Ele gerencia a infraestrutura para que os desenvolvedores possam focar na lógica de suas aplicações (SPRING, 2025).

O *Spring Boot* é uma ferramenta que simplifica e acelera o desenvolvimento de aplicativos da *web* e micros serviços com o *Spring Framework*. Ele oferece três recursos principais: autoconfiguração, uma abordagem opinativa à configuração e a capacidade de criar aplicativos autônomos (IBM, 2025).

O *Spring Data* JPA (*Java Persistence API*) aprimora substancialmente a implementação das camadas de acesso a dados, visando reduzir o esforço necessário ao mínimo indispensável. Sob essa estrutura, os desenvolvedores elaboram suas interfaces de repositório empregando uma variedade de técnicas, nas quais o Spring se encarrega de conectá-las de forma automática. Além disso, é possível utilizar localizadores personalizados ou consultas, por exemplo, cabendo ao Spring a responsabilidade de redigir a consulta correspondente (SPRING, 2025).

2.4.2.2 Angular

O Angular é um *framework* de desenvolvimento construído com base no TypeScript. Ele oferece uma estrutura baseada em componentes para criar aplicações *web* escaláveis, uma variedade de bibliotecas integradas, incluindo recursos como roteamento, gerenciamento de formulários e comunicação cliente-servidor e ferramentas de desenvolvedor para ajudar na construção, teste e atualização do código. Angular é flexível o suficiente para projetos de um desenvolvedor único ou para aplicativos de grande escala. Além disso, possui um ecossistema ativo com uma comunidade diversificada de desenvolvedores, autores de bibliotecas e criadores de conteúdo (ANGULAR, 2025).

2.4.2.3 Bootstrap

O Bootstrap é um *framework* de desenvolvimento de interfaces gratuito e de código aberto, oferecendo um conjunto de componentes pré-fabricados que facilitam a criação de sites responsivos em diversos dispositivos. Ele inclui uma ampla variedade de elementos de *design* pré-configurados e recursos adicionais através de extensões do *JavaScript*. O Bootstrap é viabilizado pela integração das estruturas de HTML, CSS e JavaScript, exigindo apenas a inclusão das dependências relevantes pelo desenvolvedor em seu projeto (BOOTSTRAP, 2025).

2.4.3 Sistema Gerenciador de Banco de Dados (SGDB)

De acordo com Elmasri et al. (2005), um Sistema Gerenciador de Banco de Dados (SGBD) são aplicações desenvolvidas para administrar bases de dados. Essencialmente, é um sistema de *software* versátil que simplifica as etapas de criação, manipulação e

compartilhamento de bancos de dados entre diferentes usuários e aplicativos. Os SGBDs abrangem a preservação e a segurança contínua do banco de dados ao longo de períodos prolongados. A proteção envolve resguardar o sistema contra falhas de *hardware* ou *software*, bem como garantir a segurança contra acessos não autorizados ou maliciosos. Devem ser capazes de manter um ambiente de banco de dados flexível e capaz de se adaptar aos requisitos em constante evolução ao longo do tempo.

O PostgreSQL é uma solução essencial de banco de dados que auxilia os desenvolvedores na preservação da integridade dos dados, na gestão eficiente de cargas de trabalho de diferentes magnitudes e na capacidade de escalabilidade conforme necessário. É reconhecido pela sua reputação sólida e oferece vantagens significativas para aplicativos que envolvem geolocalização, manipulação de séries temporais e dados relacionais. Além da sua alta confiabilidade na proteção de dados, por ser robusto, seguro e altamente personalizável, juntamente com um ecossistema diversificado de ferramentas, o PostgreSQL é utilizado por desenvolvedores em uma variedade de contextos (MICROSOFT AZURE, 2025).

2.4.4 Ferramentas CASE

Ferramentas CASE, vêm do inglês *Computer-Aided Software Engineering* e é utilizada para classificar ferramentas baseadas no ramo computacional e que ajudam em atividades da engenharia de *software*. As ferramentas CASE dão um importante suporte na modelagem de dados e na organização dos processos de desenvolvimento desde suas fases iniciais até as fases finais de um projeto (DATASUS, 2025).

2.4.4.1 Astah UML

O Astah UML é uma ferramenta de modelagem que permite a criação de diagramas utilizando a linguagem UML (*Unified Modeling Language*), amplamente empregada para especificação e visualização de sistemas de *software*. A ferramenta oferece suporte a vários tipos de diagramas, como os de classes, casos de uso, sequência e componentes, sendo especialmente útil no desenvolvimento de sistemas orientados a objetos. Com uma interface intuitiva, o Astah UML facilita a criação e o compartilhamento de diagramas detalhados, promovendo a colaboração entre desenvolvedores e analistas. Essa colaboração melhora a

documentação e o planejamento dos projetos de *software*, contribuindo para a comunicação eficaz entre as equipes envolvidas (ASTAH, 2025).

2.4.4.2 Trello

O Trello é um instrumento de gerenciamento de projetos que adota a metodologia Kanban para organizar e visualizar tarefas de forma simples e eficiente. Ele utiliza quadros, listas e cartões para representar as atividades e estágios dos projetos, facilitando a visão do progresso e a colaboração entre os membros da equipe. Ele permite ajustes no fluxo de trabalho conforme as necessidades do projeto, além de ser útil no controle de prazos, responsabilidades e prioridades. Sua capacidade de atualizar e compartilhar informações em tempo real entre os usuários é uma característica que potencializa a produtividade e o acompanhamento das tarefas (TRELLO, 2025).

2.5 ARQUITETURA DO SISTEMA

A arquitetura de um sistema, refere-se à sua estrutura e organização em vários componentes computacionais que se relacionam, sendo fundamental para a solução técnica dos problemas de negócio. A definição dessa arquitetura envolve a consideração de todos os requisitos, funcionais e não funcionais, além de sua integração com componentes do sistema em si e de outros subsistemas. Isso implica considerar nas decisões arquitetônicas, diversos fatores além das questões tecnológicas e de infraestrutura (VAROTO, 2002).

2.5.1 API

As *Application Programming Interface* (API), são interfaces que facilitam a comunicação entre dois componentes de *software*, seguindo um conjunto de definições e protocolos estabelecidos. Dentro do contexto das APIs, o termo aplicação se refere a qualquer *software* com uma função específica. A interface pode ser considerada como um contrato de serviço estabelecido entre duas aplicações, denotando como elas se comunicam por meio de

solicitações e respostas. A documentação das APIs de ambas as aplicações fornece instruções detalhadas aos desenvolvedores sobre como estruturar essas solicitações e respostas de acordo com o contrato estabelecido. (AWS, 2025).

A estrutura da API é frequentemente descrita em relação aos papéis de cliente e servidor. O cliente é a aplicação que faz a solicitação, enquanto o servidor é a aplicação que envia a resposta.

2.5.2 REST

A arquitetura *Representational State Transfer* (REST) é um conjunto de diretrizes de *design* essenciais para criar sistemas distribuídos eficientes, confiáveis e escaláveis na arquitetura de *software*.

Como qualquer outra arquitetura, o REST precisa satisfazer alguns princípios para que sua aplicabilidade esteja de acordo com o que prega a arquitetura REST. Sendo que, a conformidade com esses princípios denota a construção do que chamamos de API REST ou API RESTFul (RESTFul API, 2025).

Tendo em vista a necessidade de seguir certas orientações para que a arquitetura REST seja aplicada, temos os princípios básicos como diretriz para a estruturação da arquitetura, sendo:

a) Interface Uniforme - Define uma interface coerente e regular para a comunicação entre cliente e servidor. Sendo, um exemplo utilizado por APIs REST baseadas em HTTP é o uso de métodos padrões como o GET, POST, PUT, DELETE, entre outros.
b) Cliente-servidor - Propõe a evolução de forma independente dos componentes de clientes e servidores, melhorando questões relacionadas ao armazenamento de dados, escalabilidade, portabilidade de interfaces com os usuários. Para isso, é necessário que a comunicação entre o cliente e o servidor não seja quebrada conforme o tópico anterior.
c) *Statelessness* - Com tradução livre para “sem estado”, esse princípio exige que toda comunicação entre o cliente-servidor possua todas as informações necessárias para concluir sua requisição. Uma vez que o servidor não armazena nenhuma informação de contexto de requisições anteriores.

d) Armazenamento em cache - As restrições de cache estipulam que os dados em uma resposta a uma solicitação devem ser identificados como passíveis ou não de armazenamento em cache, seja de forma implícita ou explícita. Quando uma resposta pode ser armazenada em cache, o cache do cliente tem permissão para reutilizar os dados em respostas para futuras solicitações equivalentes.

e) Sistemas em camadas - Este princípio faz com que o comportamento dos componentes seja restringido por camadas hierárquicas, onde, em um sistema com essa estrutura, seus componentes podem se comunicar apenas com os componentes imediatamente relacionados a ele.

2.5.3 Desenho da Arquitetura

Na Figura 4 apresentamos a estrutura arquitetural do sistema desenvolvido. Seguindo um conceito onde as aplicações são subdivididas entre *back-end* e *front-end*. Sendo o *back-end* uma API REST desenvolvida em Java por meio do *Spring framework*, que faz a comunicação com o banco de dados e com o *front-end*, que por sua vez foi desenvolvido com o auxílio do *framework Angular*. Observa-se que o cliente *(front-end*), faz requisições ao *back-end* representado pela API REST, que por sua vez solicita e manipula informações nas bases de dados, para enfim retornar as respostas das requisições ao cliente e finalizar o processo de uma operação no sistema.

Figura 4 – Arquitetura do Sistema.

Fonte: o autor (2025).

2.5.4 MVC

O padrão MVC (*Model-View-Controller*) é comumente adotado no desenvolvimento de sistemas, especialmente em APIs, com o objetivo de separar as responsabilidades em três camadas: *Model*, *View* e *Controller*. Em uma API, o *Model* é encarregado de gerenciar a lógica de negócios e manipulação dos dados, interagindo com o banco de dados e validando as informações. A *View*, geralmente menos destacada, representa a estrutura da resposta que foi consumida pelo cliente, tipicamente nos formatos JSON ou XML. Já o *Controller* é responsável por processar as requisições HTTP, delegar as operações ao *Model* e retornar os resultados para o cliente.

Esse padrão de arquitetura traz diversas vantagens para o desenvolvimento de APIs, como a facilidade de manutenção, pois as mudanças na lógica de negócios não afetam diretamente a forma como os dados são apresentados, e a escalabilidade, permitindo que diferentes equipes trabalhem de forma independente nas camadas de negócios e controle. Além disso, o MVC facilita a reusabilidade de componentes, pois o *Controller* pode ser reutilizado para diferentes *endpoints* e a camada de *Model* pode ser compartilhada entre diversas APIs. Em um ambiente de APIs *RESTful*, por exemplo, o uso do padrão MVC é fundamental para garantir que o código seja modular, bem-organizado e fácil de testar, o que é essencial para a criação de sistemas robustos e escaláveis (GAMMA et al., 1995; FOWLER, 2002; SOMMERVILLE, 2011).

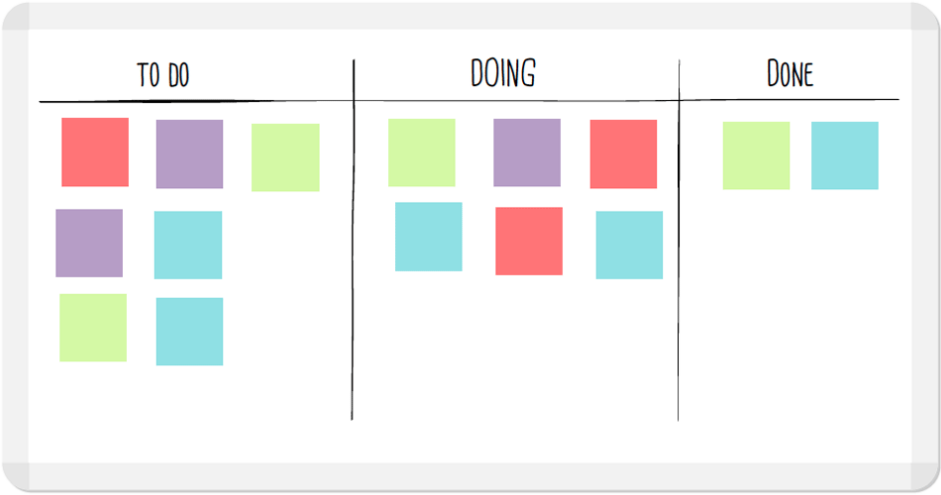
2.6 KANBAN

A metodologia Kanban é uma abordagem visual para o gerenciamento de fluxos de trabalho que prioriza a eficiência e a organização. Ela foi desenvolvida no contexto da manufatura na Toyota nos anos 1940, mas se expandiu para outros setores, incluindo o desenvolvimento de *software* e a gestão de projetos. O sistema baseia-se no uso de quadros divididos em colunas, que representam as etapas do processo, e cartões, que correspondem a tarefas específicas. À medida que uma tarefa avança, o cartão é movido entre as colunas, proporcionando uma visão clara do progresso do trabalho.

O Kanban incentiva a limitação do trabalho em andamento (*WIP,* do inglês *Work In Progress*) para evitar sobrecargas e promover um fluxo contínuo de produção. Ele também é centrado na melhoria contínua, permitindo ajustes nas etapas conforme as necessidades surgem. Por ser flexível e visual, a metodologia é amplamente adotada em contextos ágeis, onde a

colaboração e a adaptabilidade são essenciais (ANDERSON, 2010). A Figura 5 ilustra um exemplo de quadro Kanban.

Figura 5 – Representação do Kanban.



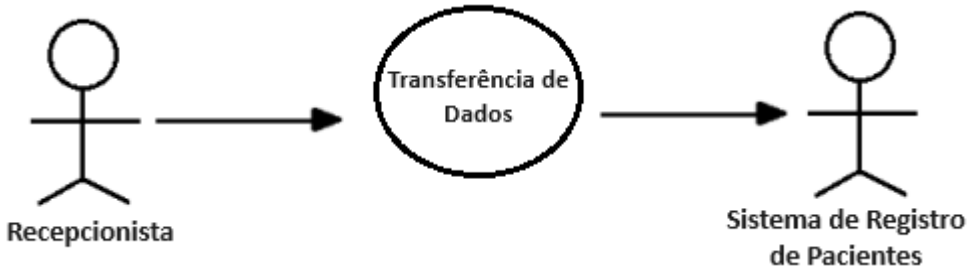
Fonte: BAKULINA, Olga (2023).

2.7 CASOS DE USO

Casos de uso podem ser definidos como um cenário simples que descreve o que se espera de um sistema na visão dos usuários. Essa abordagem de modelagem foi inicialmente desenvolvida por Jacobson et al. (1993) na década de 1990 e posteriormente incorporada na Linguagem de Modelagem Unificada (UML). Para Sommerville (2011), os casos de uso são amplamente utilizados para obtenção dos requisitos.

De modo geral, cada caso de uso pode representar um processo que envolve interação externa com um sistema, sendo em sua forma mais simples elipses que indicam ações sendo manipuladas por atores, que são identificados como atores do sistema, representados em forma de bonecos palitos. Um exemplo de Diagrama de Caso de Uso pode ser visto na Figura 6.

Figura 6 – Representação de um caso de uso.



Fonte: o autor (2025).

A Figura 6 representa um caso de uso que indica a transferência de dados para um outro sistema. Sendo, o médico recepcionista o ator que executa a tarefa de transferência de dados e o armazenamento é realizado para um sistema externo representado por Sistema de Registros de Pacientes. Sommerville (2011) destaca que, outros sistemas também podem ser identificados como atores, não somente pessoas.

3 METODOLOGIA

Para a realização do presente trabalho, foram utilizadas técnicas de análises de requisitos, a fim de solucionar os problemas descritos na Seção 1.1. Para tanto, o método de desenvolvimento de projetos de *softwares* Kanban foi escolhido para gerenciamento do desenvolvimento da aplicação. Trata-se de um modelo de desenvolvimento de *software* ágil centrado na melhoria contínua, permitindo ajustes nas etapas conforme as necessidades surgem, conforme detalhado na Seção 2.6.

O levantamento de requisitos foi realizado com base em entrevistas, realizadas diretamente com o médico do setor, e explorações em sistemas semelhantes existentes, bem como as necessidades identificadas para que um *software* deste segmento seja de fato útil para seus usuários.

O sistema foi modelado com base na Linguagem de Modelagem Unificada (UML), utilizando por tanto, técnicas de estruturação de projetos de *software* seguindo os padrões consolidados pela engenharia de *software*.

Ao final do ciclo de desenvolvimento, técnicas e ferramentas para validações e testes no sistema foram aplicadas, com a finalidade de garantir maior robustez e confiança para a aplicação, não deixando de considerar os requisitos propostos.

3.1 LEVANTAMENTO E ANÁLISE DE REQUISITOS

O levantamento de requisitos teve como finalidade compreender as necessidades dos usuários do SASAS/UFES e identificar os problemas enfrentados no processo atual de registro de informações médicas, de modo a propor soluções que atendessem de forma eficiente ao contexto da instituição.

Para isso, realizou-se uma entrevista exploratória, de caráter não estruturado, com o médico responsável pelo setor de atendimentos do SASAS/UFES. Essa escolha ocorreu pelo fato de o médico ser o principal usuário do sistema e possuir conhecimento detalhado sobre as demandas práticas do serviço, permitindo levantar informações valiosas sobre os problemas enfrentados e as funcionalidades mais necessárias em um prontuário eletrônico. Além da

entrevista, foi conduzida também uma análise de sistemas semelhantes disponíveis no mercado (Meditech, Betha Saúde e Ninsaúde), de modo a identificar recursos relevantes que poderiam ser adaptados ao contexto acadêmico, bem como limitações que justificassem o desenvolvimento de uma solução específica para a instituição.

As informações coletadas foram analisadas e organizadas em requisitos funcionais e não funcionais, conforme a literatura em engenharia de *software* recomenda. Os requisitos funcionais representaram os serviços e funcionalidades que o sistema deveria disponibilizar ao usuário, como cadastro de pacientes, registro de consultas e geração de receituários. Já os requisitos não funcionais abrangeram características relacionadas ao desempenho, segurança, acessibilidade e usabilidade da aplicação.

Essa etapa foi essencial para estabelecer a base do projeto, garantindo que o sistema desenvolvido atendesse às reais necessidades do setor de saúde universitário, ao mesmo tempo em que superasse as limitações observadas nos sistemas existentes, sobretudo em relação à simplicidade, custo e adequação ao contexto acadêmico.

3.1.1 Requisitos funcionais

De acordo com Sommerville (2011), os requisitos funcionais de um sistema descrevem o que a aplicação deve fazer, ou seja, como será o comportamento do que está sendo desenvolvido. Para o desenvolvimento deste trabalho, foram levantados os seguintes requisitos:

RF01 - Cadastrar Paciente - O sistema deverá permitir o cadastro de um paciente utilizando as seguintes informações: nome, CPF, contato, data de nascimento, peso, altura, sexo, e-mail e endereço.

RF02 - Editar Paciente - O sistema deverá permitir a edição das informações de um paciente.

RF03 - Consultar Paciente - O sistema deverá permitir a consulta das informações de pacientes cadastrados.

RF04 - Cadastrar Médico - O sistema deverá permitir o cadastro de um médico utilizando as seguintes informações: nome, CPF, contato, data de nascimento e CRM.

RF05 - Editar Médico - O sistema deverá permitir a edição das informações de um médico cadastrado.

RF06 - Consultar Médico - O sistema deverá permitir a consulta das informações de médicos cadastrados.

RF07 - Cadastrar Usuário - O sistema deverá permitir o cadastro de usuários com e-mail e senha para autenticação.

RF08 - Editar Usuário - O sistema deverá permitir a edição de informações de usuários cadastrados.

RF09 - Consultar Usuário - O sistema deverá permitir a consulta de informações de usuários cadastrados.

RF10 - Inativar Usuário - O sistema deverá permitir a inativação de usuários cadastrados.

RF11 - Autenticar Usuário - O sistema deverá permitir a autenticação de usuários para acesso às funcionalidades, utilizando login e senha cadastrados.

RF12 - Cadastrar Consulta - O sistema deverá permitir o cadastro de consultas incluindo médico responsável, paciente, data, hora, motivo, diagnóstico, CID, alergias, prescrições, medicamentos, tratamentos, solicitações de exame, resultados de exame e arquivos associados.

RF13 - Editar Consulta - O sistema deverá permitir a edição das informações de uma consulta cadastrada.

RF14 - Consultar Consulta - O sistema deverá permitir a consulta das informações de consultas cadastradas.

RF15 - Cadastrar Diagnóstico - O sistema deverá permitir o cadastro de diagnósticos para uma consulta.

RF16 - Editar Diagnóstico - O sistema deverá permitir a edição de diagnósticos cadastrados para uma consulta.

RF17 - Cadastrar CID - O sistema deverá permitir o cadastro de código e descrição de CID em diagnósticos de consultas.

RF18 - Excluir CID - O sistema deverá permitir a exclusão de códigos CID cadastrados em diagnósticos de consultas.

RF19 - Cadastrar Alergia - O sistema deverá permitir o cadastro de alergias associadas a uma consulta.

RF20 - Remover Alergia - O sistema deverá permitir a remoção de alergias associadas a uma consulta.

RF21 - Cadastrar Prescrição - O sistema deverá permitir o cadastro de prescrições contendo medicamentos, dosagens e instruções de uso.

RF22 - Editar Prescrição - O sistema deverá permitir a edição de prescrições cadastradas.

RF23 - Cadastrar Medicamento na Consulta - O sistema deverá permitir o cadastro de medicamentos em uma consulta.

RF24 - Remover Medicamento da Consulta - O sistema deverá permitir a remoção de medicamentos cadastrados em uma consulta.

RF25 - Cadastrar Tratamento - O sistema deverá permitir o cadastro de tratamentos em uma consulta.

RF26 - Editar Tratamento - O sistema deverá permitir a edição de tratamentos cadastrados em uma consulta.

RF27 - Cadastrar Solicitação de Exame - O sistema deverá permitir o cadastro de solicitações de exames em uma consulta.

RF28 - Excluir Solicitação de Exame - O sistema deverá permitir a exclusão de solicitações de exames cadastradas em uma consulta.

RF29 - Cadastrar Arquivo da Consulta - O sistema deverá permitir o cadastro de arquivos anexados a uma consulta.

RF30 - Remover Arquivo da Consulta - O sistema deverá permitir a remoção de arquivos anexados a uma consulta.

RF31 - Consultar Agendamento de Consulta - O sistema deverá permitir a consulta dos agendamentos de consultas cadastrados.

RF32 - Cadastrar Agendamento de Consulta - O sistema deverá permitir o cadastro de agendamentos de consultas.

RF33 - Editar Agendamento de Consulta - O sistema deverá permitir a edição de agendamentos de consultas cadastrados.

RF34 - Gerar Consulta a partir do Agendamento - O sistema deverá permitir a geração de uma consulta a partir de um agendamento existente.

3.1.2 Requisitos não funcionais

Requisitos não funcionais (RNF) correspondem a restrições e critérios que delimitam como o sistema deve operar, indo além das funcionalidades específicas que ele oferece. Esses

requisitos geralmente tratam de características como desempenho, segurança, escalabilidade, disponibilidade e conformidade com normas ou padrões. Eles também podem incluir restrições relacionadas ao ambiente de desenvolvimento, tecnologias utilizadas, tempo de resposta, ou até mesmo a forma como os dados devem ser armazenados e protegidos. Em essência, os RNFs descrevem atributos qualitativos do sistema que impactam diretamente na sua usabilidade e eficácia (SOMMERVILLE, 2011). Os requisitos não funcionais para o sistema proposto são:

RFN01 – O sistema não deverá expor dados de pacientes e médicos a usuários não autenticados.

RFN02 – Responsividade e compatibilidade – O sistema deverá ser responsivo e funcionar nos principais navegadores .

RFN03 – Usabilidade – O sistema deverá ser simples e intuitivo, seguindo padrões consistentes de navegação e rotulagem, minimizando passos para tarefas frequentes.

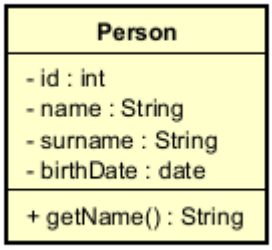
RFN04 – Autenticação e autorização – O sistema deverá exigir autenticação com política de senhas, expiração de sessão inativa.

RFN05 – Auditoria – O sistema deverá registrar trilhas de auditoria para operações sensíveis.

3.1.3 Diagrama de classes

O Diagrama de Classes é um artefato da UML que representa a estrutura estática do sistema, descrevendo as classes, atributos, operações e relacionamentos. Esse diagrama é fundamental para compreender a arquitetura do sistema e orientar o desenvolvimento (SOMMERVILLE, 2011). A Figura 7 exemplifica a representação desse modelo.

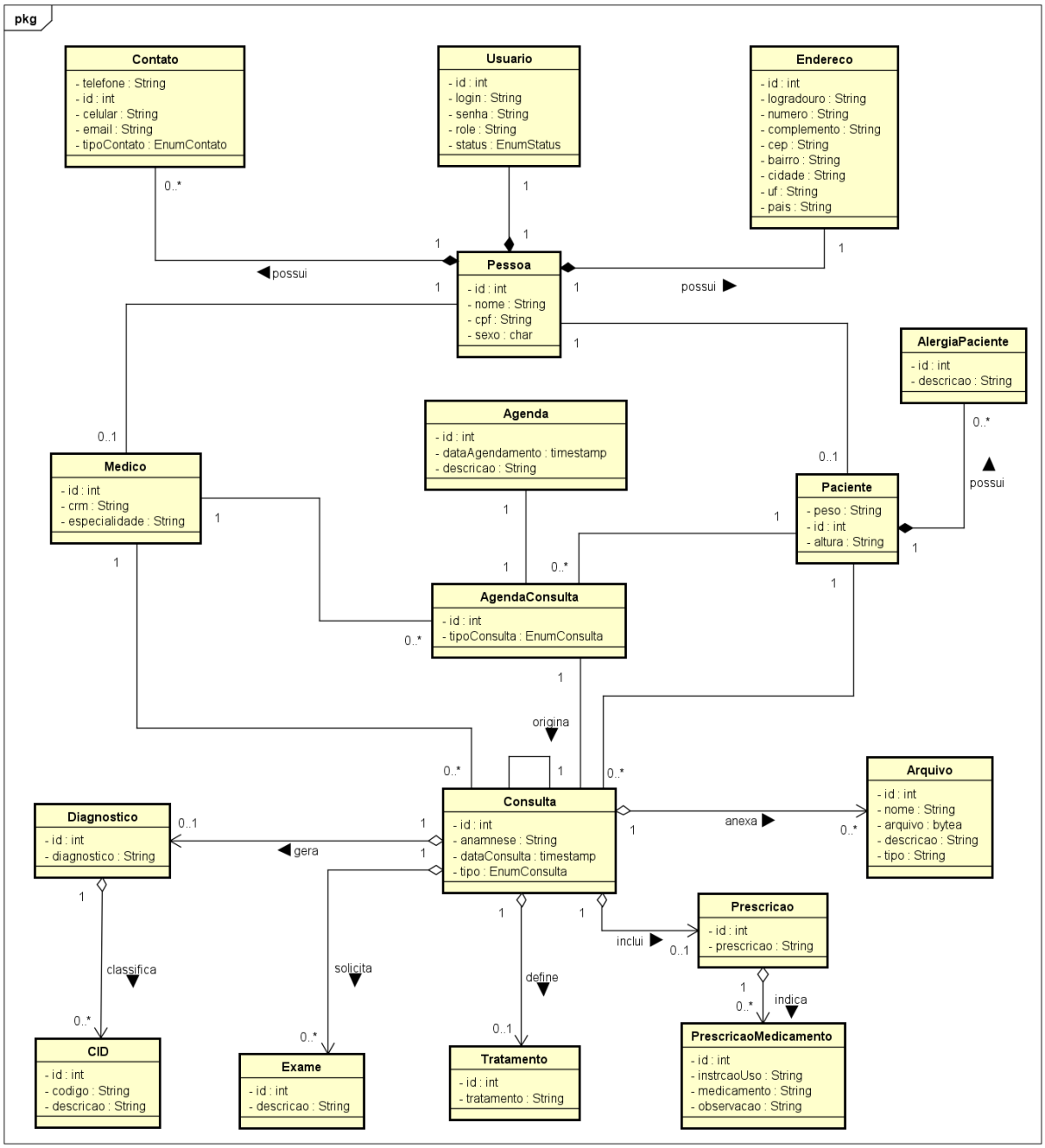
Figura 7 – Representação de um modelo em um diagrama de classes UML.



Fonte: o autor (2025).

No contexto deste projeto, o diagrama foi elaborado com o intuito de fornecer uma visão geral da estrutura e apoiar a implementação das funcionalidades derivadas dos casos de uso. O diagrama de classes encontra-se na Figura 8, a seguir.

Figura 8 - Diagrama de classes do sistema.

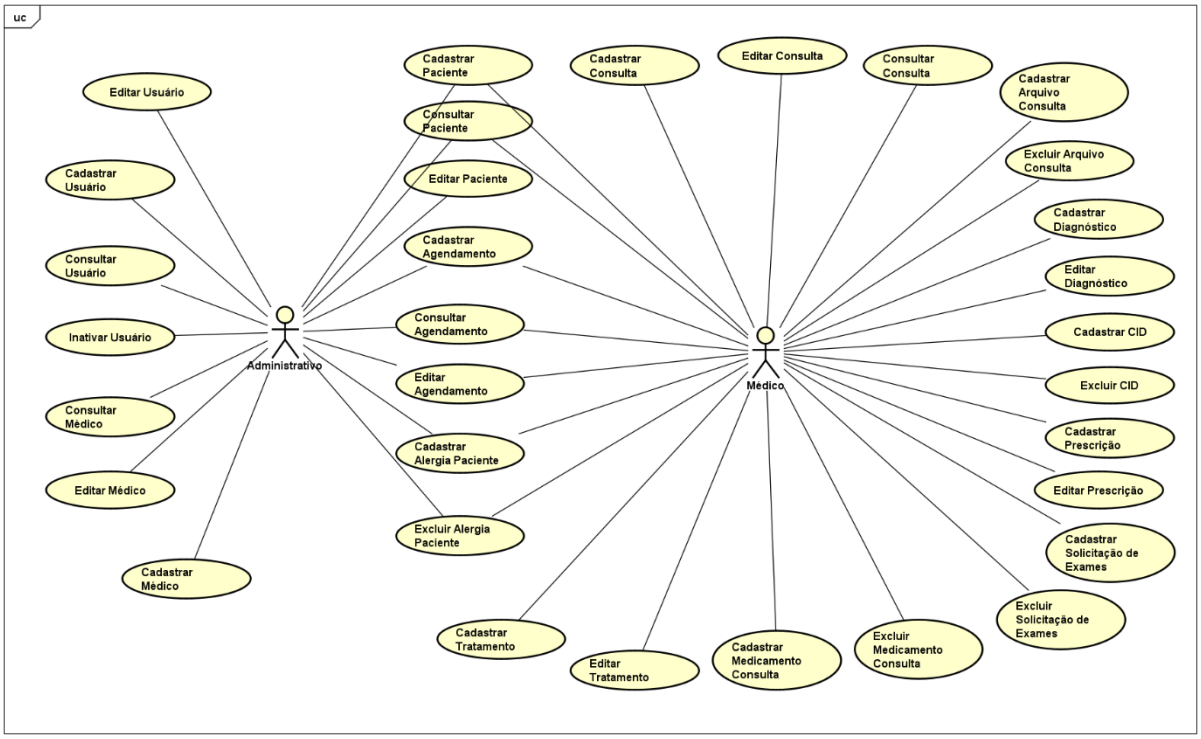


Fonte: o autor (2025).

3.1.4 Diagrama de caso de uso

Para o desenvolvimento deste projeto, adotou-se a abordagem de especificação de requisitos baseada em Casos de Uso, conforme descrito na Seção 2.7. Nessa perspectiva, o Diagrama de Casos de Uso foi elaborado como instrumento para identificar, organizar e representar, de forma clara, as interações entre os atores e as funcionalidades previstas para o sistema. O diagrama definido, que serviu de base para a construção e detalhamento dos requisitos funcionais apresentados, encontra-se ilustrado na Figura 9.

Figura 9 - Diagrama de casos de uso.



Fonte: o autor (2025).

3.1.5 Narrativas de casos de uso

Uma narrativa de casos de uso é uma descrição detalhada de como um sistema será utilizado em situações do mundo real. Ela geralmente é escrita na forma de texto narrativo, descrevendo passo a passo as interações entre os usuários e o sistema para atingir um objetivo específico.

Para Cockburn (2005), as narrativas de casos de uso ajudam a prever o sistema em uso. Sua descrição ajuda a compreender detalhes de algum processo que será executado pelo sistema, sendo ela, responsável por narrar a interação dos atores do sistema com as funcionalidades propostas.

A seguir são apresentados, como exemplo, as narrativas de caso de uso do UC001 e UC002, os demais serão encontrados no Apêndice A.

UC001 - Cadastrar Paciente

Descrição: Este caso de uso descreve como se dá a inserção de um novo paciente no sistema.

Atores: Usuário (Médico/Administrativo).

Pré-condições: O usuário deve estar logado no sistema.

Pós-condições: Um novo paciente é registrado no sistema.

Fluxo Principal:

[FP01] - Cadastro de Paciente.

01 - O sistema exibe a tela de cadastro de paciente.

02 - O usuário preenche o CPF do paciente.

03 - O usuário preenche o nome do paciente.

04 - O usuário preenche a data de nascimento do paciente.

05 - O usuário preenche o sexo do paciente.

06 - O usuário preenche a altura do paciente.

07 - O usuário preenche o peso do paciente.

08 - O usuário clica no botão salvar. [FE01][FA01].

09 - O sistema cadastra o paciente no banco de dados.

10 - O sistema exibe a MSG001.

Fluxos de alternativos:

[FA01] - Cancelamento do cadastro do paciente.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de pacientes.

Fluxos exceção:

[FE01] - Campo obrigatório não preenchido.

01 - O sistema exibe a mensagem de erro abaixo do campo obrigatório.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP01] no passo 08.

Mensagens do Sistema:

MSG001: “Paciente cadastrado com sucesso.”

MSG002: “Preencha os campos obrigatórios do paciente.”

UC002 - Cadastrar Consulta

Descrição: Este caso de uso descreve como se dá a inserção de uma nova consulta no sistema.

Atores: Médico.

Pré-condições: O médico deve estar logado no sistema.

Pós-condições: Uma nova consulta é registrada no sistema.

Fluxo Principal:

[FP02] - Cadastro de consulta.

01 - O Sistema exibe a tela de cadastrar uma nova consulta.

02 - O médico seleciona o tipo de consulta.

03 – O médico seleciona o paciente [FA02].

04 – O sistema exibe as informações do paciente selecionado.

05 – O médico clica no botão salvar [FA3][FE2].

06 – O sistema exige a confirmação MSG003 para salvar a consulta [FA4].

07 – O médico confirma as informações e salva a consulta.

08 – O sistema exibe a mensagem MSG004.

Fluxos de alternativos:

[FA02] - Paciente não cadastrado no sistema.

01 - O médico cadastra o paciente de acordo com o [FP01 - UC001].

02 - Retorna ao [FP02] no passo 02.

[FA03] – Cancelamento do cadastro da consulta.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de consultas.

[FA04] – Médico não confirma o cadastro da consulta.

01 - O usuário clica no botão NÃO.

02 – Retorna ao [FP02] no passo 04.

Fluxos exceção:

[FE02] – Médico não seleciona um campo obrigatório.

01 - O sistema exibe a MSG005.

02 - O sistema finaliza a operação sem salvar a consulta.

Mensagens do Sistema:

MSG004: “Consulta salva com sucesso.”

MSG005: “Selecione um paciente e um tipo de consulta.”

3.2 MODELAGEM DE BANCO DE DADOS

A modelagem de banco de dados diz respeito ao processo de levantamento, categorização e exploração de todos os tipos de dados que irão compor o sistema a ser desenvolvido. Esta etapa é essencial para que a implementação do *software* ocorra de maneira coerente com as necessidades dos usuários. Quando mal elaborada, a complexidade para o desenvolvimento e posteriormente manutenções, se tornam um grande desafio e até mesmo podem inutilizar todo um sistema.

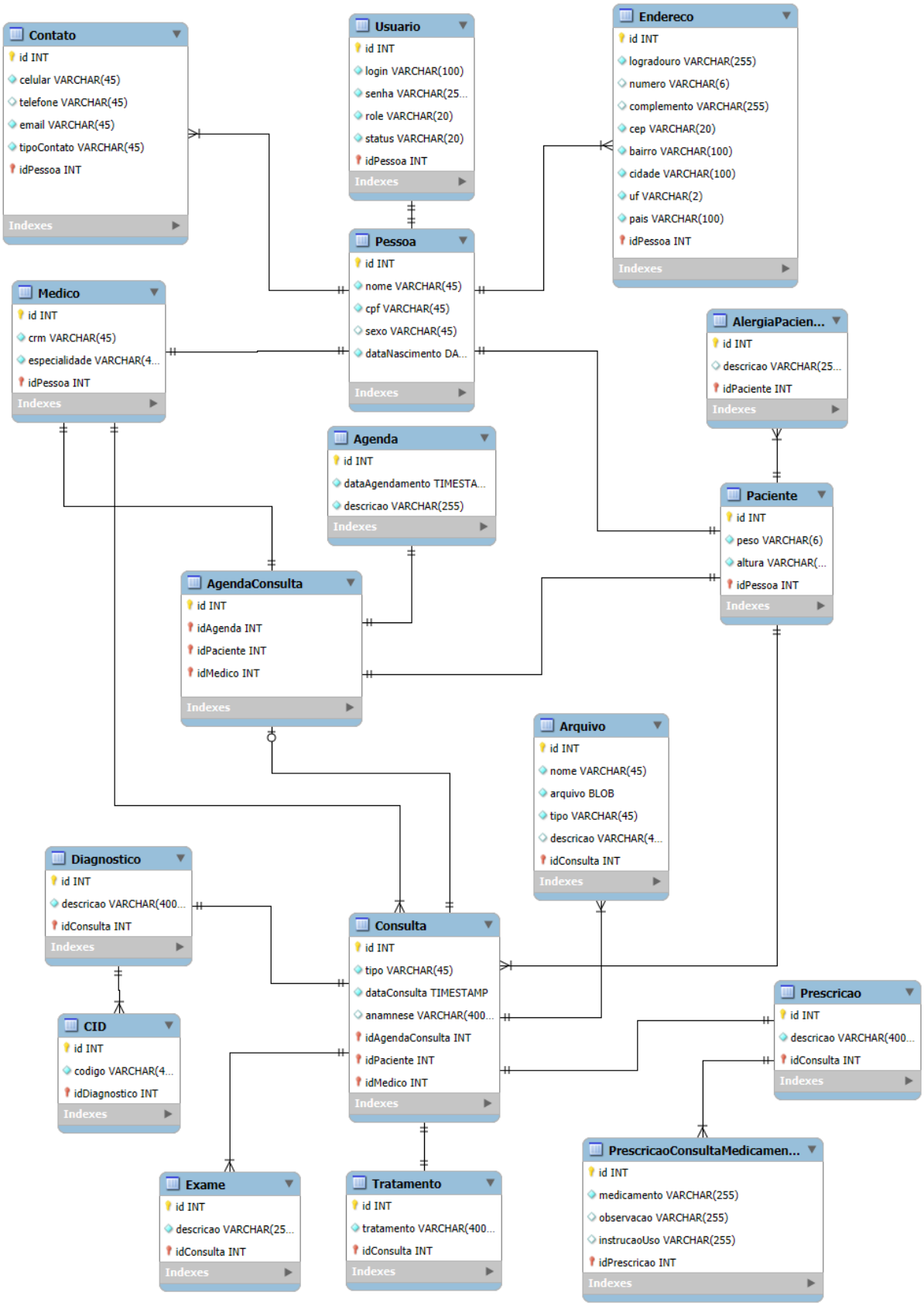
Neste trabalho, optou-se pela modelagem física de dados por possibilitar um detalhamento completo da estrutura implementada, incluindo definições precisas de tipos de dados, chaves, índices e relacionamentos. Essa abordagem garante que todos os requisitos técnicos sejam contemplados desde o início, reduzindo riscos de incompatibilidade e retrabalho na fase de desenvolvimento e facilitando a implementação no SGBD escolhido.

3.2.1 Modelo físico de dados

Segundo Lehmkuhl e Eger (2013), o modelo físico de dados representa as informações do sistema em uma estrutura de armazenamento de dados. Tem como objetivo traduzir os conceitos do negócio para um banco de dados. A partir desse momento, as entidades são definidas mais detalhadamente, considerando seus atributos, tipos e relações. Este modelo é iniciado após a estruturação do modelo conceitual, fazendo com que as abstrações obtidas no modelo anterior, ganhe um aspecto mais robusto e detalhado.

A Figura 10 apresenta o Modelo Entidade Relacionamento Físico proposto para o sistema.

Figura 10 - Modelo físico de banco de dados.



Fonte: o autor (2025).

3.2.2 Dicionário de dados

A Tabela 2 apresenta a estrutura da tabela Pessoa do banco de dados, as estruturas das demais tabelas estão disponíveis no Apêndice B.

Tabela 2 - Tabela Pessoa.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
nome	VARCHAR(255)	Nome completo da pessoa
cpf	VARCHAR(14)	CPF da pessoa
sexo	CHAR(1)	Sexo da pessoa
data_nascimento	DATE	Data nascimento da pessoa

Fonte: O autor (2025).

.

4 DESENVOLVIMENTO DA APLICAÇÃO

Visando facilitar e automatizar os processos de atendimento aos pacientes em hospitais disponíveis em instituições de ensino, foi desenvolvido um Sistema de Prontuário Médico Eletrônico. Este sistema *web* oferece uma ampla gama de funcionalidades, como o gerenciamento de dados clínicos, histórico de consultas, controle de exames, medicamentos e agendamentos, sempre visando o fluxo de trabalho em ambientes médicos com o intuito de promover um serviço de qualidade de forma mais eficiente possível.

Para tanto, a metodologia Kanban foi adotada durante todo o processo de desenvolvimento, garantindo um acompanhamento visual das etapas propostas. A ferramenta Trello foi empregada como suporte para a organização dos processos.

O desenvolvimento do sistema foi baseado nos requisitos e modelos definidos na Seção 3 deste documento, assegurando que as funcionalidades implementadas atendam às necessidades específicas do público-alvo e estejam alinhadas com as boas práticas de desenvolvimento de *software* para aplicações médicas.

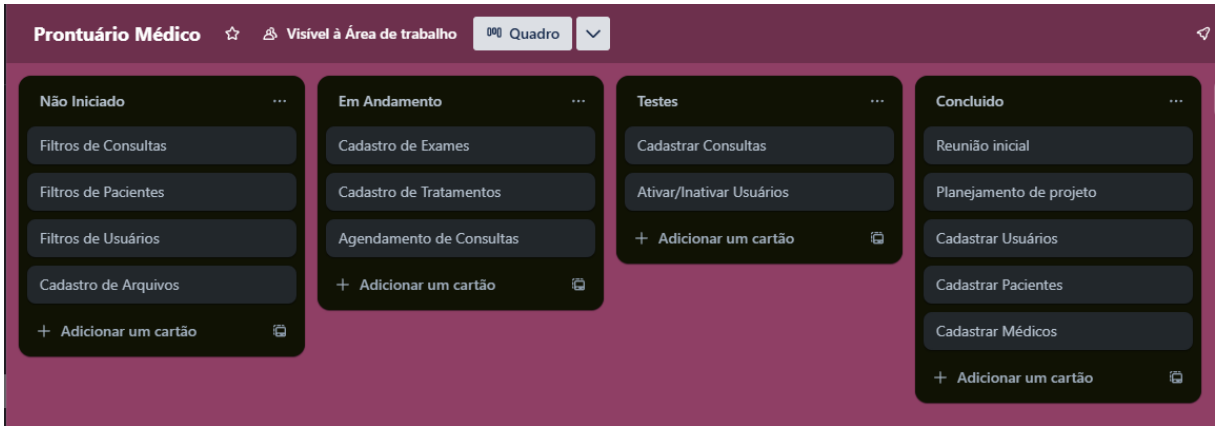
4.1 ORGANIZAÇÃO DOS PROCESSOS

Dada a utilização do Kanban no Trello, foram projetados quadros com colunas distintas para refletir as fases do desenvolvimento. As colunas escolhidas para representarem as etapas do projeto foram: “Não iniciado”, “Em andamento”, “Testes” e “Concluído”.

Com base nos requisitos funcionais, as atividades do projeto são detalhadas em cards, contendo informações que identificam o que deve ser realizado na tarefa. Esses cards são utilizados no Trello para organizar e acompanhar o progresso das tarefas, sendo movimentados entre as colunas do quadro conforme o desenvolvimento avança, o que assegura um gerenciamento eficiente e visual do fluxo de trabalho.

A Figura 11 exemplifica o quadro do projeto no Trello, demonstrando sua interface.

Figura 11 - Representação quadro de tarefas do Trello



Fonte: O autor (2025)

Desta forma podemos observar de maneira eficiente o desenvolvimento do sistema de acordo com as tarefas devidamente registradas em suas respectivas etapas.

4.2 A APLICAÇÃO

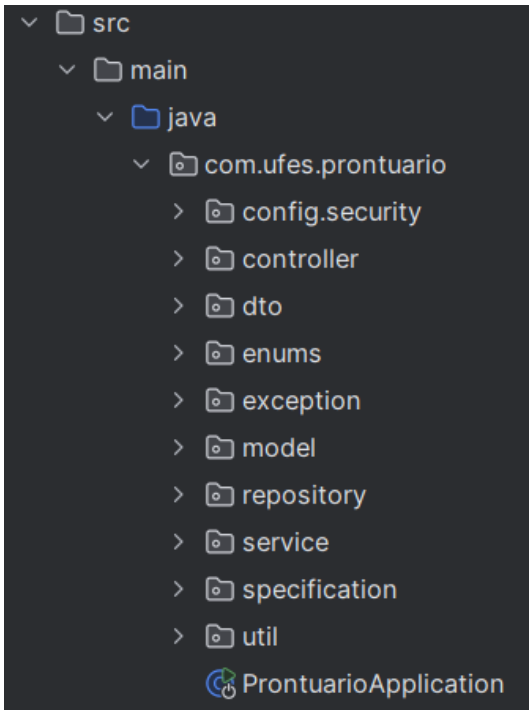
O sistema foi desenvolvido com uma arquitetura desacoplada, utilizando o *framework* Spring Boot para o *back-end* e Angular para o *front-end*, permitindo uma comunicação eficiente entre as camadas por meio de APIs RESTful. O *front-end* foi implementado com tecnologias como *TypeScript*, *JavaScript*, CSS e HTML5, proporcionando interfaces dinâmicas e responsivas. A autenticação de usuários foi realizada com o uso de *tokens* JWT (*JSON Web Token*), garantindo funcionalidades como *login*, registro e gerenciamento seguro de sessões. O PostgreSQL foi adotado como sistema de gerenciamento de banco de dados, assegurando confiabilidade e desempenho no armazenamento de informações. Nas próximas seções, serão apresentados aspectos relacionados à aplicação dessas tecnologias, bem como outros elementos relevantes para o desenvolvimento do sistema.

4.2.1 Estrutura do projeto

O padrão de arquitetura utilizado no sistema foi o MVC conforme explicado na Seção 2.5.4. O MVC proporciona uma divisão bem definida e estruturada das funções do sistema, favorecendo a modularidade e a reutilização do código.

A Figura 12 ilustra a estrutura do projeto do *back-end*, destacando a organização das pastas, cada uma com uma função específica na hierarquia.

Figura 12 - Representação da estrutura de pacotes do projeto *back-end.*



Fonte: O autor (2025).

O Quadro 1 explica a função de cada uma destas pastas, relacionadas a arquitetura do *software* projetado.

Quadro 1 - Descrição funcionalidade dos pacotes do projeto *back-end.*

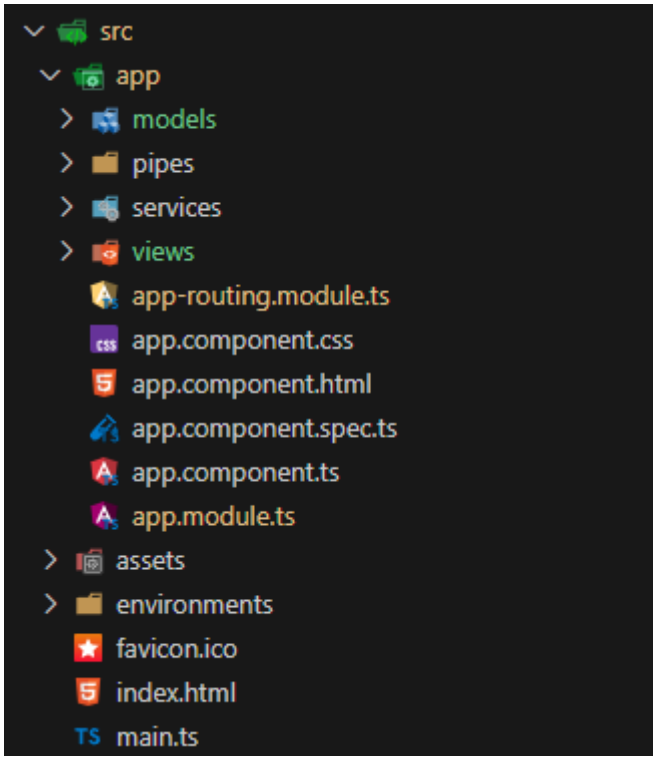
Pasta	**Função**
config	Armazena as configurações de inicialização do sistema, como configurações de banco de dados, segurança e dependências.
controller	Contém as classes responsáveis por gerenciar as requisições HTTP e direcioná-las para os serviços adequados.

dto	Utilizado para transferir dados entre camadas do sistema, frequentemente em formato simplificado ou específico para comunicação.
enums	Contém enumeradores (tipos enumerados) usados para representar um conjunto fixo de constantes.
exception	Define as exceções personalizadas para tratar erros específicos do sistema.
model	Representa as entidades de dados, geralmente mapeando tabelas de banco de dados ou objetos do domínio.
repository	Contém as classes responsáveis pela interação com o banco de dados.
service	Implementa a lógica de negócios do sistema, orquestrando as operações entre controladores, repositórios e outros componentes.
specification	Define critérios complexos de consulta ou regras de negócio para consultas dinâmicas.
util	Contém classes utilitárias com funções comuns e auxiliares usadas em várias partes do sistema.

Fonte: O autor (2025).

A Figura 13 ilustra a estrutura do projeto do *front-end*, destacando a organização e sua função.

Figura 13 - Representação da estrutura de pacotes do projeto *front-end.*



Fonte: O autor (2025).

O Quadro 2 explica a função de cada uma destas pastas e/ou arquivos, relacionadas a arquitetura do *software* projetado.

Quadro 2 - Descrição funcionalidade dos pacotes do projeto *front-end.*

Pasta	**Função**
model	Representa as entidades de dados, geralmente mapeando objetos que são utilizados no *back-end*.
pipes	Representa os validadores e tratadores de dados manipulados no *front-end.*
service	Implementa a lógica de negócios do sistema, orquestrando as operações entre o usuário e o *back-end.*
views	Responsável por armazenar os arquivos que definem a estrutura e o layout das páginas, essa camada apresenta de forma organizada os dados do modelo e permite a interação do usuário com o sistema.

assets	Define arquivos auxiliares que não são código executável, mas são essenciais para a aparência, funcionamento e identidade visual da aplicação.
environments	Utilizada na configuração que definem variáveis de ambiente específicas para cada estágio da aplicação, como desenvolvimento, testes ou produção.

Fonte: O autor (2024).

Com a adoção do padrão MVC e uma estrutura de pacotes organizada, o desenvolvimento do sistema foi otimizado, proporcionando uma base robusta para a implementação de funcionalidades e facilitando a manutenção do código de maneira eficiente.

4.2.2 Conexão com o banco de dados

A configuração de conexão com o banco de dados PostgreSQL é feita através do arquivo *application.properties*, localizado na raiz dos *resources* projeto. Neste arquivo, são definidos detalhes importantes como o nome do banco de dados, o usuário, a senha e o endereço do servidor do banco de dados. Essas configurações permitem que o *Spring Boot* estabeleça a conexão com o banco de dados de forma automática ao inicializar a aplicação.

A manipulação de registros no banco de dados é facilitada pelo *Spring Data JPA,* que integra a especificação *Java Persistence API* (JPA) com o *Hibernate*, um framework de mapeamento objeto-relacional (ORM). A JPA permite que você trabalhe com objetos Java, enquanto o *Hibernate* cuida da conversão desses objetos em registros no banco de dados, além de fornecer diversas funcionalidades para a interação com o banco, como consultas e transações.

Com a conexão estabelecida, o aplicativo *Spring Boot* utiliza o PostgreSQL para armazenar e gerenciar dados de forma eficiente, com o suporte do Spring Data JPA e do *Hibernate* para o mapeamento das entidades. Isso garante uma interação simples com o banco de dados e uma base sólida para a persistência de informações, permitindo operações CRUD eficazes.

Na Figura 14 é apresentado o *resources/application.properties* do projeto, juntamente com comentários em cada linha que explica cada uma delas.

Figura 14 - Conexão com o banco de dados.

```
spring.application.name=prontuario

##PRD

#Especifica a URL da conexão com o banco de dados
#Composta pelo Host, Port e Database
spring.datasource.url=jdbc:postgresql://${PGHOST}:${PGPORT}/${PGDATABASE}

#Define o nome e a senha do usuário utilizado para autenticação ao banco de dados
spring.datasource.username=${PGUSER}
spring.datasource.password=${PGPASSWORD}

#Configura o comportamento do Hibernate na inicialização em relação è criação ou atualização do esquema do banco de dados
spring.jpa.hibernate.ddl-auto=update

#Definição de exibição de logs das execuções geradas no banco de dados
spring.jpa.show-sql=false

#Define a comunicação da aplicação com o Banco de Dados.
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Fonte: O autor (2025).

4.2.3 Segurança do sistema e autenticação dos usuários

O *Spring Security* é um módulo do *framework Spring* que fornece recursos de autenticação e autorização para aplicações Java. No sistema desenvolvido, ele é responsável por controlar o acesso às rotas da API, permitindo ou negando requisições com base nas regras definidas. Utiliza-se uma abordagem sem estado (*stateless*), onde cada requisição deve conter informações de autenticação válidas, dispensando o uso de sessões no servidor (SPRING.IO, 2024).

A Figura 15 ilustra a configuração da segurança no *Spring Security*. São definidos *endpoints* públicos, como autenticação e monitoramento, e outros protegidos por permissões específicas de acordo com o papel do usuário. As requisições não mapeadas exigem autenticação.

Figura 15 - Configuração de Segurança da API no *Spring Security.*

```
@Bean  papa +1 *
public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {
    return httpSecurity
            .csrf(AbstractHttpConfigurer::disable)
            .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .authorizeHttpRequests(authorize -> authorize
                    .requestMatchers("/actuator/health").permitAll() AuthorizationManagerRequestMat...
                    .requestMatchers(HttpMethod.POST, "/api/auth/autenticar", "/api/auth/registrar", "/api/auth/recuperar-senha").permitAll()

                    .requestMatchers(HttpMethod.PATCH, "/api/auth/alterar-status/**").hasRole("ADMINISTRATIVO")

                    .requestMatchers(HttpMethod.PATCH, "/api/auth/alterar-senha").hasAnyRole( ...roles: "ADMINISTRATIVO", "MEDICO")

                    .requestMatchers(HttpMethod.GET, "/pessoas/**", "/pacientes/**", "/contatos/**", "/medicos/**",
                            "/consultas/**", "/alergias-paciente/**", "/tratamentos/**", "/diagnosticos/**", "/agenda-consulta/**").hasAnyRole( ...roles: "MEDICO", "ADMINISTRATIVO")

                    .requestMatchers(HttpMethod.POST, "/pessoas/**", "/contatos/**", "/pacientes/**", "/medicos/**",
                            "/consultas/**", "/alergias-paciente/**", "/tratamentos/**", "/diagnosticos/**", "/agenda-consulta/**").hasAnyRole( ...roles: "MEDICO", "ADMINISTRATIVO")

                    .requestMatchers(HttpMethod.PUT, "/pessoas/**", "/pacientes/**", "/contatos/**", "/medicos/**",
                            "/consultas/**", "/alergias-paciente/**", "/tratamentos/**", "/diagnosticos/**", "/agenda-consulta/**").hasAnyRole( ...roles: "MEDICO", "ADMINISTRATIVO")

                    .requestMatchers(HttpMethod.DELETE, "/pessoas/**", "/pacientes/**", "/contatos/**", "/medicos/**",
                            "/consultas/**", "/alergias-paciente/**", "/tratamentos/**", "/diagnosticos/**", "/agenda-consulta/**").hasAnyRole( ...roles: "MEDICO", "ADMINISTRATIVO")

                    .anyRequest() AuthorizedUrl
                    .authenticated()
            )
            .addFilterBefore(securityFilter, UsernamePasswordAuthenticationFilter.class)
            .build();
}
```

Fonte: o autor (2025).

Para autenticação, o sistema utiliza *JSON Web Token* (JWT), que permite validar a identidade do usuário de forma segura por meio de um *token* assinado. Após o *login*, o *back-end* gera um token contendo informações do usuário, que é enviado ao *front-end* e incluído nas requisições subsequentes. O *Spring Security* valida esse *token* antes de permitir o acesso aos recursos protegidos (SPRING.IO, 2024).

Para garantir a segurança no armazenamento de senhas dos usuários cadastrados, o sistema utiliza o algoritmo *BCrypt*, uma função de *hash* robusta e amplamente recomendada para esse propósito. No momento do cadastro de um novo usuário, a senha informada é automaticamente criptografada antes de ser persistida no banco de dados, evitando que senhas em texto simples sejam armazenadas. Essa abordagem está alinhada às boas práticas de segurança da informação, contribuindo para a proteção dos dados sensíveis dos usuários do sistema (ESCUDÉLARIO, 2023). A Figura 16 ilustra como são persistidas as senhas criptografadas.

Figura 16 - Senhas criptografadas de usuários.

id	login	senha
10	admin@ufes.br	$2a$10$k5QN6IOdhHKdAtBVUYCxsOOWDnwZV.TeBswElPwy.HQnDsfxVMwuS
1	medico@ufes.br	$2a$10$ZxQu5FK.Luk4Iabb.DcLoe8vt4J9WdLS6EF78nmwgWiTYiBK46F7.

Fonte: O autor (2025).

4.2.4 Rotas de navegação

O roteamento no Angular estabelece o mapeamento entre URLs e componentes por meio de um arranjo *Routes*, registrado no módulo de roteamento. Cada entrada de rota define um *path* e o *component* correspondente, e, em cenários com *layout* persistente, uma rota pai hospeda rotas filhas que são renderizadas no *RouterOutlet* do componente pai, o que permite atualizar apenas o conteúdo interno sem recriar a estrutura de navegação. Essa arquitetura favorece a organização por áreas funcionais e a reutilização de *layouts*.

Além do mapeamento básico, as rotas podem declarar redirecionamentos e rotas curinga para lidar com caminhos não mapeados. Rotas com parâmetros dinâmicos (por exemplo, :id) permitem identificar recursos específicos e são combinadas na ordem definida, por isso, recomenda-se declarar primeiro as rotas mais específicas e deixar o curinga por último.

O roteamento também suporta metadados e mecanismos de controle diretamente na configuração: *guards* para controle de acesso, *resolvers* para pré-carregar dados antes da ativação do componente. Esses recursos tornam a navegação previsível, segura e coesa, mantendo as políticas de acesso e o carregamento de dados declarativos e próximos à própria definição das rotas. A Figura 17 demonstra a configuração realizada para o contexto dessa aplicação.

Figura 17 - Configuração de rotas e *guards* do Angular.

```
const routes: Routes = [
  { path: 'login', component: LoginComponent },
  { path: 'recuperar-senha', component: RecuperarSenhaRequestComponent },
  { path: '', redirectTo: 'login', pathMatch: 'full' },
  {
    path: 'prontuario', component: DashboardComponent, children:
      [
        { path: 'home', component: HomeComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },
        { path: 'usuarios', component: UsuarioComponent, canActivate: [roleGuard], data: { roles: ['ROLE_ADMINISTRATIVO'] } },
        { path: 'usuarios/editar/:id', component: EditarUsuariosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_ADMINISTRATIVO'] } },
        { path: 'usuarios/cadastrar', component: CadastrarUsuariosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_ADMINISTRATIVO'] } },
        { path: 'alterar-senha', component: AlteracaoSenhaComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },


        { path: 'pacientes', component: PacientesComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },
        { path: 'pacientes/editar/:id', component: EditarPacienteComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },
        { path: 'pacientes/cadastrar', component: CadastraPacientesComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },

        { path: 'medicos', component: MedicosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_ADMINISTRATIVO'] } },
        { path: 'medicos/editar/:id', component: EditarMedicoComponent, canActivate: [roleGuard], data: { roles: ['ROLE_ADMINISTRATIVO'] } },
        { path: 'medicos/cadastrar', component: CadastraMedicosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_ADMINISTRATIVO'] } },

        { path: 'agendamentos', component: AgendamentosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },
        { path: 'agendamentos/editar/:id', component: EditarAgendamentosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },
        { path: 'agendamentos/cadastrar', component: CadastrarAgendamentosComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO', 'ROLE_ADMINISTRATIVO'] } },

        { path: 'consultas', component: ConsultasComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO',] } },
        { path: 'consultas/cadastrar', component: ManterConsultasComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO',] } },
        { path: 'consultas/editar/:id', component: ManterConsultasComponent, canActivate: [roleGuard], data: { roles: ['ROLE_MEDICO',] } },

        { path: '', redirectTo: 'home', pathMatch: 'full' }
      ]
  },
  { path: '**', redirectTo: '/login' }

];
```

5 GARANTIA DE QUALIDADE E PROCESSOS DE TESTES

Ao longo da construção do projeto, foram conduzidos procedimentos e cenários de verificação, apresentados nas próximas seções, com a finalidade de confirmar a conformidade do sistema em relação aos objetivos propostos e aos requisitos funcionais definidos.

5.1 TESTES DE DESENVOLVIMENTO

Os testes de desenvolvimento, quando aplicados de forma iterativa durante a implementação, têm papel fundamental na verificação contínua do alinhamento entre o que está sendo construído e os requisitos previamente definidos. Essa prática envolve não apenas a execução de testes unitários e de integração, mas também a revisão constante dos requisitos funcionais, possibilitando ajustes imediatos sempre que divergências são identificadas. Pressman e Maxim (2016) destacam que essa abordagem incremental e integrada aos ciclos de codificação favorece a detecção precoce de inconsistências, reduz o retrabalho e aumenta a qualidade do *software* entregue.

Neste sentido, os testes do sistema desenvolvido seguiram uma abordagem de validação contínua dos requisitos, garantindo a qualidade e a conformidade com as especificações definidas. Os problemas identificados durante o processo foram devidamente documentados, e as respectivas soluções estão registradas no Quadro 3.

Quadro 3 - Defeitos encontrados e soluções aplicadas.

Defeito	**Solução**
Mensagem exceção não condizente com a ação executada	Adequação da mensagem de erro para refletir com clareza o problema apresentado
Autenticação não expirava	Configuração no *back-end* para estabelecer um tempo de sessão de 2 horas.
Acesso a rotas diretamente pela URL sem autenticação do usuário	Implementação de guardas (Guards) nas rotas no *front-end*.

Problemas no console da aplicação durante o carregamento das telas	Tratamento das variáveis que ainda não estavam prontas para uso, evitando erros de *null pointer* ou *undefined*.
Usuários acessando funcionalidades indevidas para seu perfil	Adequação da configuração de segurança no *back-end* para restringir o acesso às rotas conforme o perfil, utilizando *Spring Security*.
Campos obrigatórios não estavam sendo validados corretamente no formulário	Implementação de validações no *front-end.*
Falha no logout não invalidando o token de sessão	Implementação de lógica para invalidar o token no *back-end* e limpar as credenciais no *front-end.*

Fonte: O autor (2025).

5.2 TESTES UNITÁRIOS

Os testes de unitários têm como objetivo verificar o comportamento isolado de pequenas partes do código, como métodos ou classes, garantindo que funcionem conforme o esperado em diferentes cenários. Para isso, fazem uso de técnicas como a criação de *mocks* e *stubs*, que permitem simular dependências externas e focar exclusivamente na lógica interna da unidade testada. Beck (2003) destaca que a aplicação sistemática de testes de unidade não apenas contribui para a detecção precoce de defeitos, mas também aumenta a confiabilidade do sistema e facilita a manutenção do código.

Visando a aplicação de testes unitários, foram desenvolvidas classes específicas para cada serviço do sistema, assegurando a consistência e a integridade de suas funcionalidades. Essa prática possibilitou a identificação de problemas à medida que alterações eram realizadas nos serviços implementados, contribuindo para a manutenção da qualidade do código e a redução de falhas em produção.

Como forma de ilustrar a aplicação dessa abordagem, apresenta-se a seguir um exemplo de algoritmo de teste unitário implementado no sistema representado na Figura 18. Esse teste foi desenvolvido para validar as principais operações do serviço de gerenciamento de

agendamentos de consultas, verificando desde a recuperação de registros até sua exclusão, de forma a assegurar que cada funcionalidade atenda ao comportamento esperado.

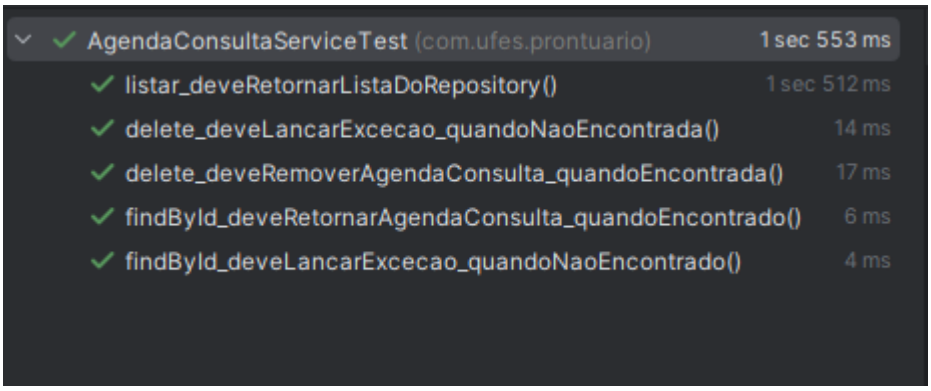
Figura 18 – Testes unitários do serviço de AgendaConsulta.

```
21  public class AgendaConsultaServiceTest {
35      private AgendaConsultaService service;
36
37      @Test  David Papa
38      void findById_deveRetornarAgendaConsulta_quandoEncontrado() {
39          AgendaConsulta agenda = new AgendaConsulta();
40          agenda.setId(1L);
41          when(repository.findById(1L)).thenReturn(Optional.of(agenda));
42          AgendaConsulta result = service.findById(1L);
43          assertNotNull(result);
44          assertEquals( expected: 1L, result.getId());
45      }
46
47      @Test  David Papa *
48      void findById_deveLancarExcecao_quandoNaoEncontrado() {
49          when(repository.findById(1L)).thenReturn( t: Optional.empty());
50          assertThrows(RecursoNaoEncontradoException.class, () -> service.findById(1L));
51      }
52
53      @Test  David Papa
54      void listar_deveRetornarListaDoRepository() {
55          when(repository.findAll()).thenReturn(List.of(new AgendaConsulta()));
56          List<AgendaConsulta> result = service.listar();
57          assertEquals( expected: 1, result.size());
58      }
59
60      @Test  David Papa *
61      void delete_deveRemoverAgendaConsulta_quandoEncontrada() {
62          AgendaConsulta agenda = new AgendaConsulta();
63          agenda.setId(1L);
64          when(repository.findById(2L)).thenReturn(Optional.of(agenda));
65          service.delete( id: 1L);
66          verify(repository).delete(agenda);
67      }
68
69      @Test  David Papa *
70      void delete_deveLancarExcecao_quandoNaoEncontrada() {
71          when(repository.findById(1L)).thenReturn( t: Optional.empty());
72          assertThrows(RecursoNaoEncontradoException.class, () -> service.delete( id: 1L));
73      }
74  }
```

Fonte: O autor (2025).

A Figura 19 exemplifica a execução dos testes, evidenciando a aprovação integral da funcionalidade avaliada por determinada classe de testes. Nesse contexto, a obtenção desse resultado indica que os testes estão plenamente alinhados à proposta funcional, validando o correto comportamento da implementação.

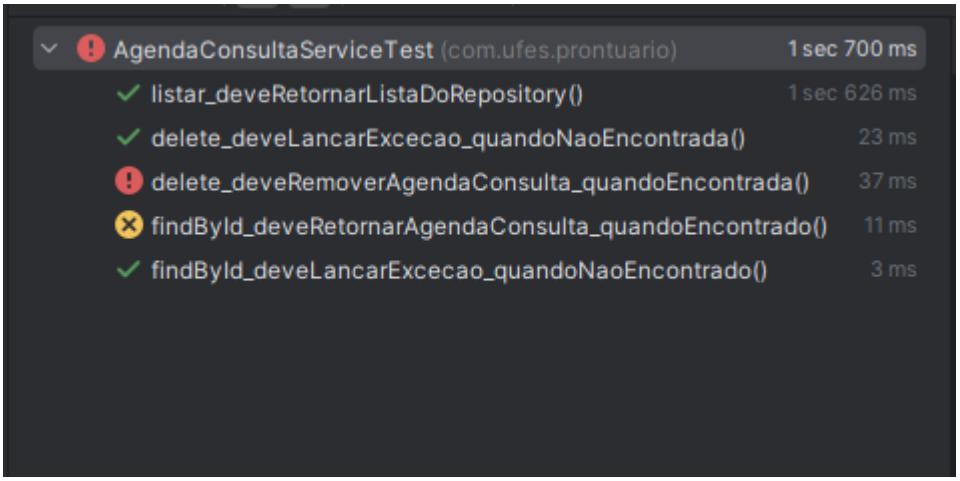
Figura 19 – Execução de testes bem-sucedida.



Fonte: O autor (2025).

Em contrapartida, quando alterações são realizadas nos serviços e alguma funcionalidade é comprometida, a execução dos testes unitários permite identificar rapidamente o problema. A Figura 20 ilustra a situação em que uma modificação impacta o funcionamento previamente correto do sistema, evidenciando falhas que necessitam de correção.

Figura 20 – Falha na execução de testes.



Fonte: O autor (2025).

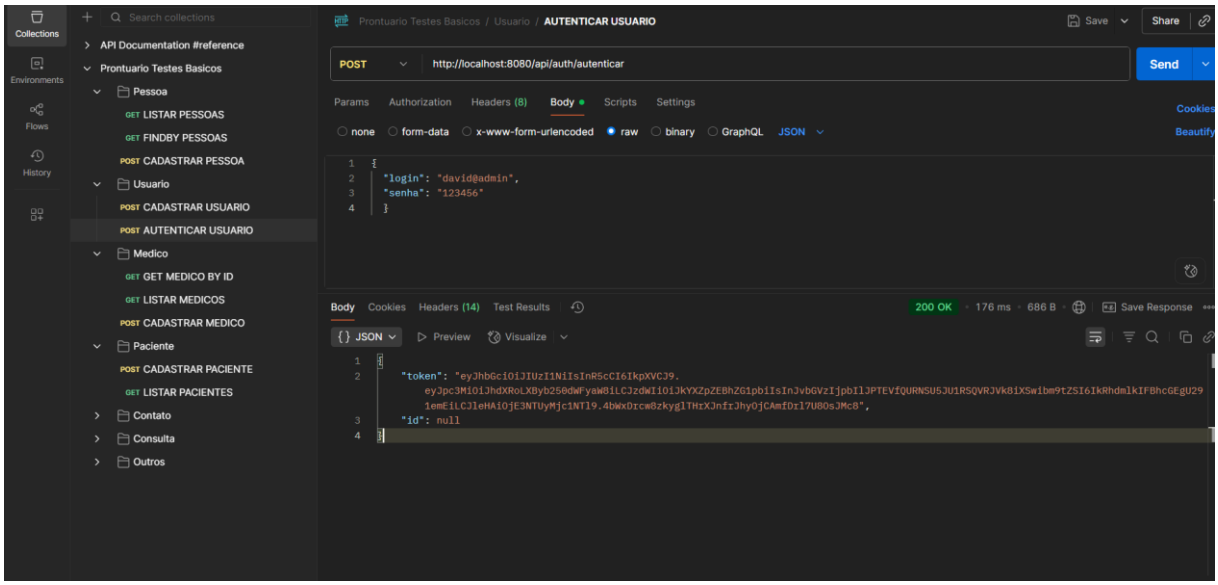
5.3 TESTES DE INTEGRAÇÃO

Os testes de integração, também denominados testes de API quando aplicados a serviços *web*, têm como finalidade validar o funcionamento conjunto dos diferentes componentes do sistema, assegurando que a comunicação entre eles ocorra de forma correta e produza os resultados esperados. Esse tipo de teste é realizado com a aplicação em execução, enviando requisições diretamente aos *endpoints* e percorrendo todas as camadas, desde a interface de entrada até a persistência no banco de dados. São verificados aspectos como códigos de status, estrutura e conteúdo da resposta, bem como a integridade e consistência dos dados armazenados. Conforme aponta Fowler (2012), essa abordagem é essencial para detectar falhas que não seriam identificadas em testes unitários, garantindo que o sistema opere de maneira integrada e atenda aos requisitos especificados.

Para a execução desses testes no presente trabalho, foi utilizada a ferramenta *Postman*, que permitiu simular requisições HTTP reais e inspecionar de forma detalhada as respostas retornadas pelo sistema.

A Figura 21 exemplifica chamadas realizadas por meio do Postman, utilizadas para a execução dos testes de integração. Nesse caso, é apresentada a requisição de autenticação do sistema, evidenciando o envio dos dados de acesso e a obtenção do token de autenticação como resposta.

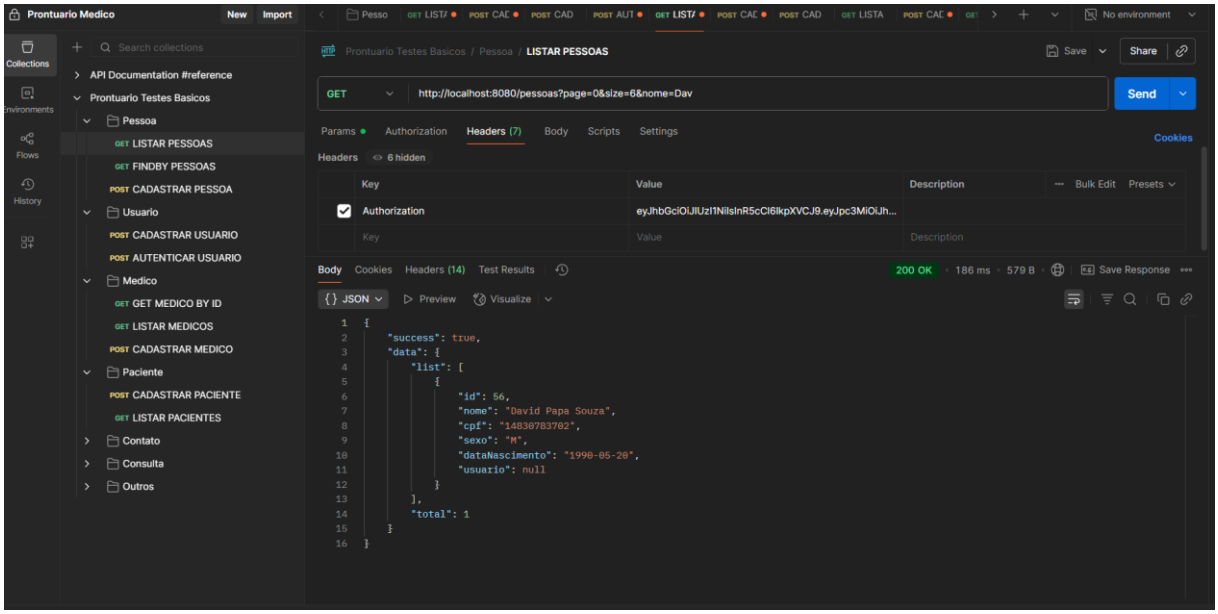
Figura 21 – Testes de integração via *postman.* Autenticação de usuários.



Fonte: O autor (2025).

Com a obtenção do *token* de autenticação da aplicação, torna-se possível executar outras operações de teste no sistema. A Figura 22 apresenta um exemplo dessa etapa, ilustrando a requisição de listagem de pacientes com a aplicação de filtro pelo nome.

Figura 22 – Execução de teste de API Consultar Pessoas.



Fonte: O autor (2025).

5.4 IMPACTO E RESULTADO DOS TESTES NO SISTEMA

A execução sistemática de testes unitários e de integração no sistema permitiu identificar e corrigir falhas de forma precoce, reduzindo retrabalhos e assegurando maior confiabilidade à aplicação. Os testes unitários, ao verificarem funcionalidades isoladas dos serviços, possibilitaram validar a lógica de negócio e garantir a consistência de cada módulo individualmente. Já os testes de integração, realizados com a aplicação em execução e utilizando dados efetivamente registrados no banco de dados, asseguraram que as camadas do sistema se comunicassem adequadamente e atendessem aos requisitos funcionais.

5.5 CRITÉRIOS DE ANÁLISE DOS RESULTADOS

A análise dos resultados obtidos com o sistema proposto concentrou-se em métricas de desempenho e robustez técnica, considerando que não foram realizados testes diretos com os usuários finais. Os principais critérios avaliados foram:

- Eficiência de processamento: tempo de resposta das requisições entre o *front-end* (Angular) e o *back-end* (Spring Boot);
- Consumo de recursos: utilização de memória e processamento durante operações de inserção, consulta e atualização de dados;
- Confiabilidade: consistência do armazenamento das informações no banco de dados PostgreSQL, garantindo a integridade dos registros;
- Segurança básica: funcionamento dos mecanismos de autenticação e autorização implementados, assegurando que apenas usuários autorizados pudessem acessar funcionalidades restritas;
- Escalabilidade inicial: capacidade do sistema em lidar com múltiplas operações simultâneas sem comprometer o desempenho.

Esses critérios permitiram verificar a conformidade do sistema com os requisitos não funcionais estabelecidos na fase de especificação, além de fornecer indicadores objetivos sobre sua viabilidade técnica. A análise detalhada dos resultados é apresentada no Capítulo 6, em conjunto com as telas e fluxos do sistema implementado.

6 RESULTADOS ALCANÇADOS

Os resultados obtidos demonstraram o atendimento ao objetivo geral deste trabalho, uma vez que foi desenvolvido e validado um sistema web de prontuário eletrônico adequado ao contexto do SASAS/UFES, capaz de organizar e disponibilizar informações clínicas de forma estruturada e acessível.

Em relação aos objetivos específicos, constatou-se que todos foram contemplados no sistema implementado. A aplicação apresenta uma interface intuitiva acessível por navegadores, garantindo o gerenciamento de dados de pacientes e profissionais de saúde de forma integrada. O registro de consultas, exames, doenças e tratamentos foi incorporado, permitindo a construção de um histórico clínico completo. Também foram implementadas funcionalidades para controle de documentos médicos, como receituários e atestados, além da gestão de perfis de acesso com autenticação.

Os requisitos não funcionais também foram atendidos, conforme demonstrado pelos testes técnicos realizados. O sistema apresentou tempo de resposta médio inferior a dois segundos para operações de cadastro, consulta e atualização, não registrou falhas em cem execuções consecutivas de operações críticas, e obteve consistência de 100% nos dados armazenados e recuperados no banco de dados PostgreSQL. Além disso, foram confirmados o funcionamento adequado dos mecanismos de autenticação e de controle de acesso por perfil, bem como a estabilidade em execuções contínuas sem interrupções inesperadas. Esses resultados evidenciam que o sistema alcançou robustez e confiabilidade satisfatórias para o ambiente ao qual se destina.

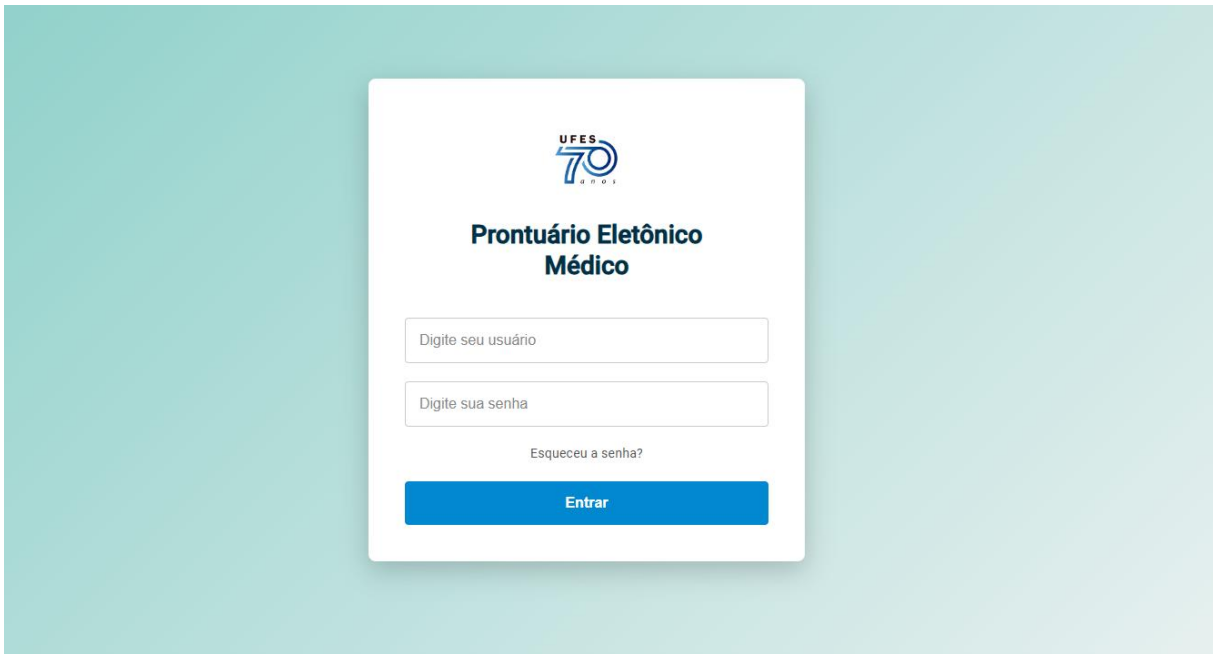
Ainda que não tenham sido conduzidos testes formais com usuários finais, a validação técnica permitiu confirmar a viabilidade prática da aplicação. Futuras avaliações em caráter piloto, realizadas com profissionais do SASAS/UFES, poderão complementar essa etapa, fornecendo indicadores adicionais de usabilidade e aceitação.

Finalmente, apresentam-se as telas do sistema desenvolvido, que ilustram as principais funcionalidades e interações implementadas, servindo como evidência prática da consolidação dos requisitos levantados e do processo de desenvolvimento conduzido.

6.1.1 Telas do sistema

Todas as funcionalidades do sistema requerem um usuário autenticado, portanto a tela de *login* é a página inicial do sistema. Nesta tela, o usuário tem a opção de realizar o login ou solicitar a recuperação das suas credenciais. A Figura 23 demonstra a tela inicial da aplicação.

Figura 23 – Página de autenticação do sistema.



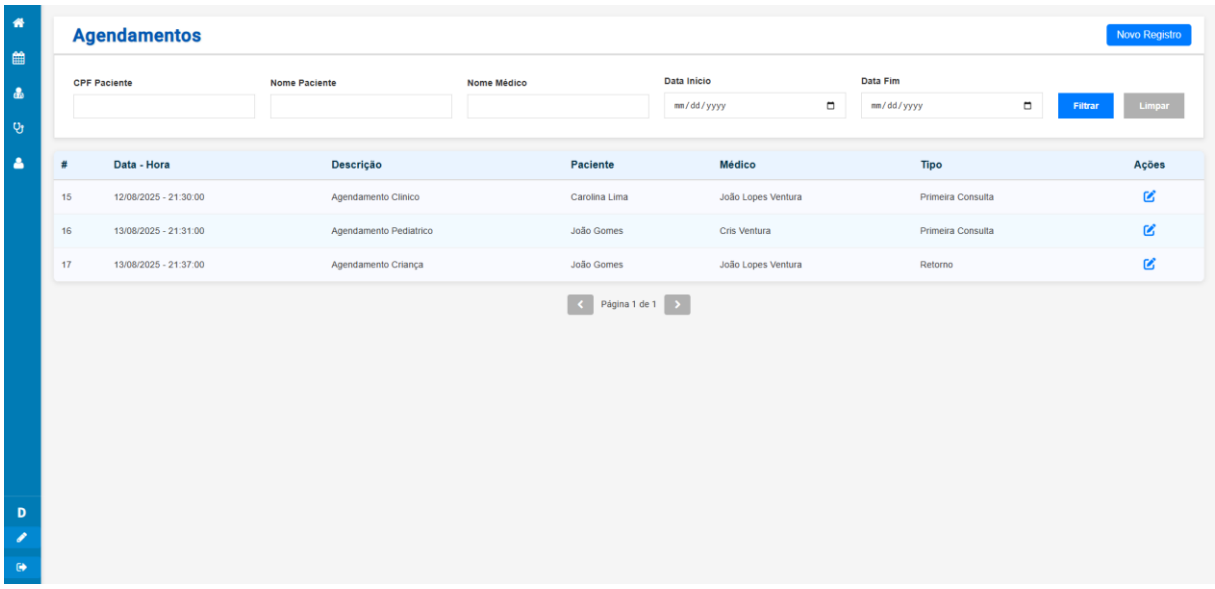
Fonte: O autor (2025).

6.1.2 Fluxo comum aos perfis de usuários

Algumas funcionalidades da aplicação são compartilhadas entre os dois perfis de usuários (Médico e Administrativo). Essas funcionalidades estão acessíveis por meio do menu principal e podem ser utilizadas tanto por usuários do perfil médico quanto do perfil administrativo. Nesta sessão detalharemos cada uma dessas funcionalidades.

No contexto do agendamento de consultas, o sistema permite que o usuário autenticado realize a pesquisa de agendamentos previamente registrados. Essa funcionalidade possibilitou a busca utilizando diferentes critérios, como o CPF do paciente, o nome do paciente, o nome do médico, a data de início e/ou a data de término do agendamento. A Figura 24 apresenta a tela de listagem dos agendamentos disponíveis no sistema.

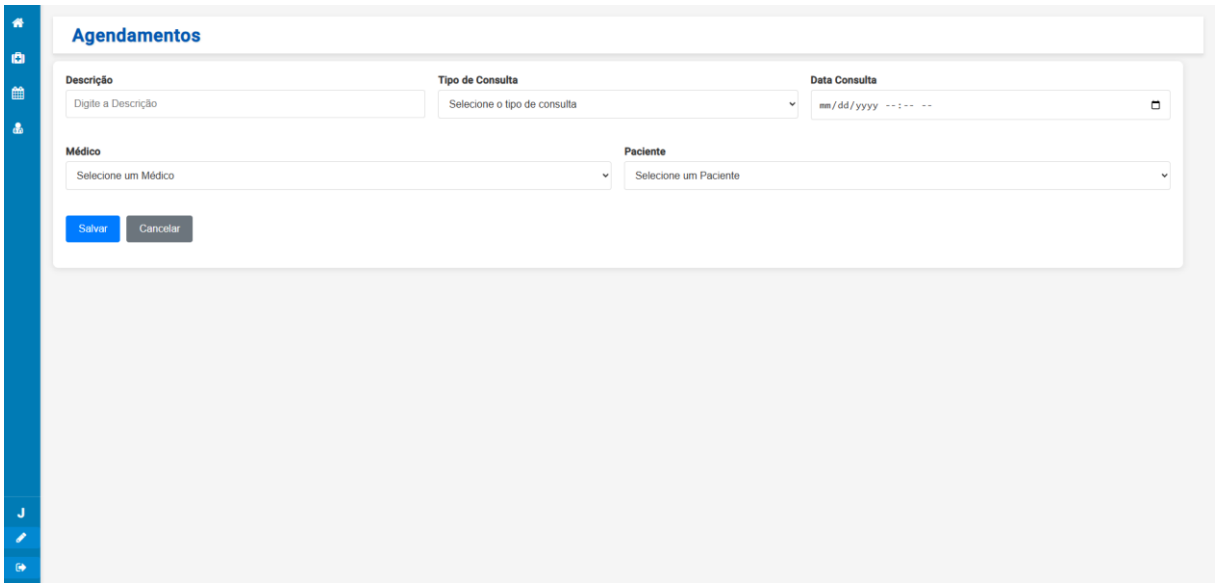
Figura 24 – Consulta de agendamentos.



Fonte: O autor (2025).

Ainda no agendamento de consultas, o sistema permite que o usuário autenticado realize o cadastro de um novo agendamento ou a edição de um agendamento já existente. Para tanto, devem ser informados dados como a descrição, o tipo da consulta, a data e o horário do atendimento, o médico responsável e o paciente vinculado. Ressalta-se que, para efetuar o cadastro, é imprescindível que tanto o médico quanto o paciente estejam previamente registrados na aplicação. A Figura 25 apresenta a tela de cadastro e edição de agendamentos disponível no sistema.

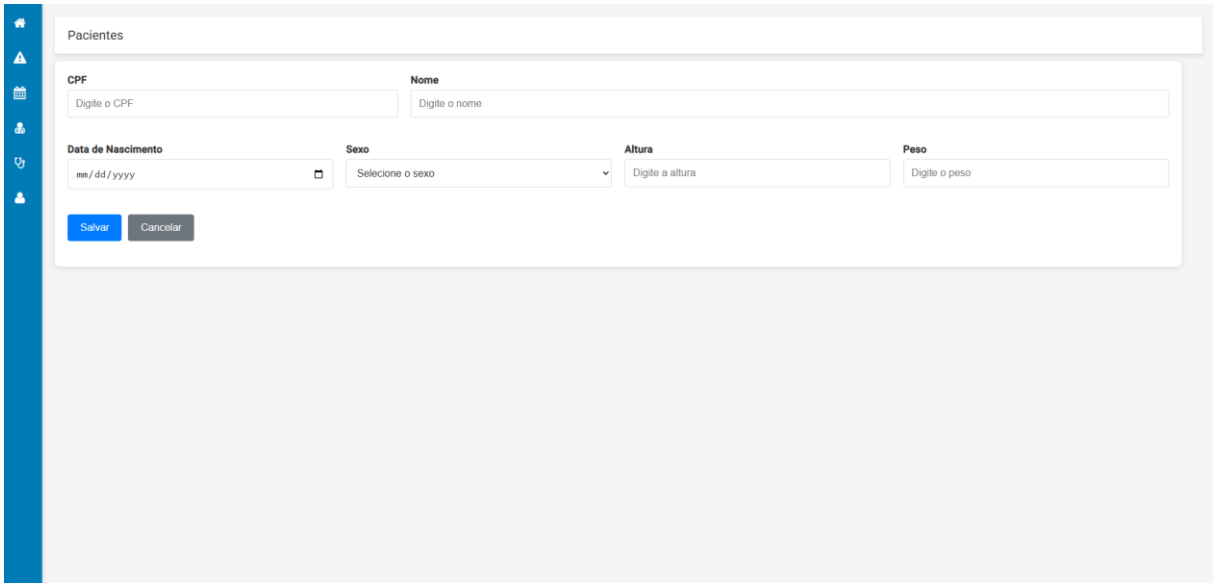
Figura 25 – Cadastro de agendamento.



Fonte: O autor (2025) .

A tela de cadastro de pacientes é subdividida em quatro etapas, sendo a principal delas dedicada ao preenchimento dos dados pessoais de identificação do paciente. A Figura 26 apresenta a interface correspondente à etapa de cadastro dessas informações. O cadastro de pacientes é detalhado conforme os casos de uso UC001 disponível na Seção 3.1.5, e UC003, UC004, UC005 disponível nos Apêndices.

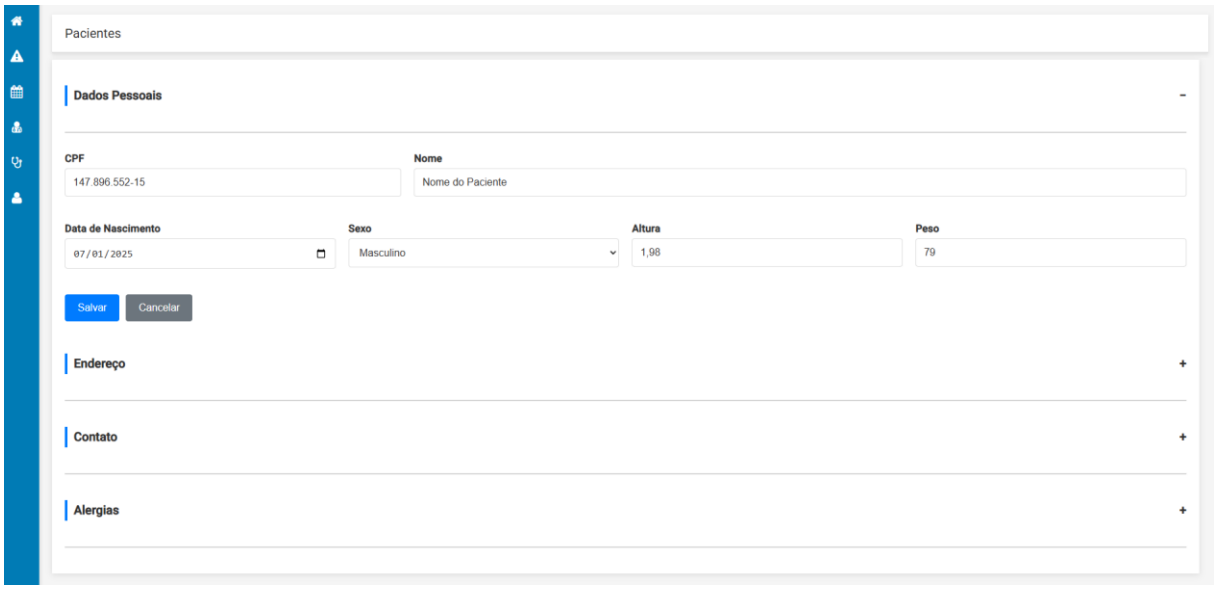
Figura 26 – Cadastro de pacientes.



Fonte: O autor (2025).

Após o salvamento das informações de identificação da pessoa, o sistema disponibiliza novas opções para complementar o registro do paciente, tais como o cadastro de endereço, contatos e alergias. A Figura 27 ilustra a interface utilizada para o preenchimento dessas informações complementares.

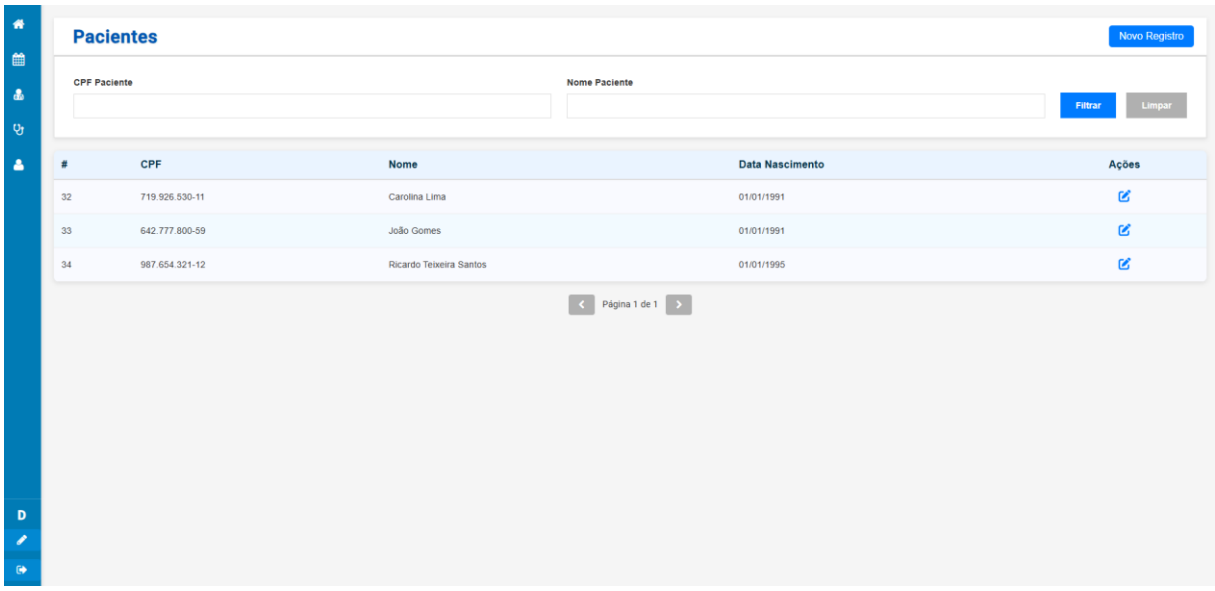
Figura 27 – Edição e complemento de cadastro do paciente.



Fonte: O autor (2025).

Ainda no que se refere aos pacientes, o sistema permite que usuários autenticados consultem os registros de pacientes cadastrados. A busca pode ser realizada por meio do CPF ou do nome do paciente, facilitando a localização das informações desejadas. A Figura 28 ilustra a tela de consulta de pacientes disponível no sistema.

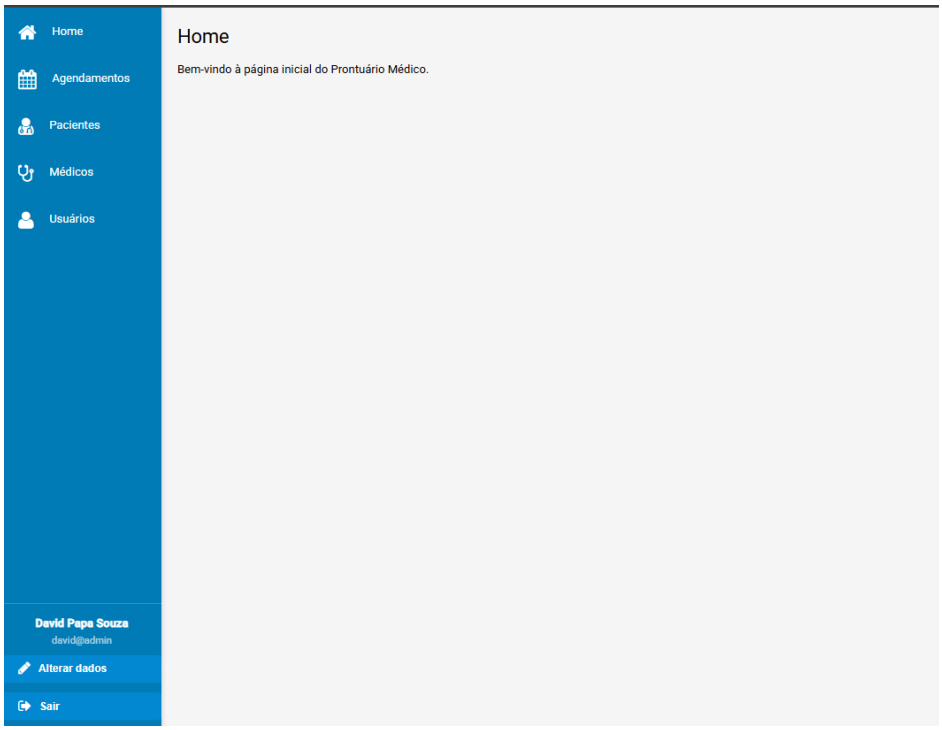
Figura 28 – Consulta de pacientes.



Fonte: O autor (2025).

As informações referentes ao usuário autenticado no sistema são exibidas no canto inferior esquerdo da interface. Nesse espaço, são apresentados o nome e o login do usuário atualmente conectado, conforme ilustrado na Figura 29.

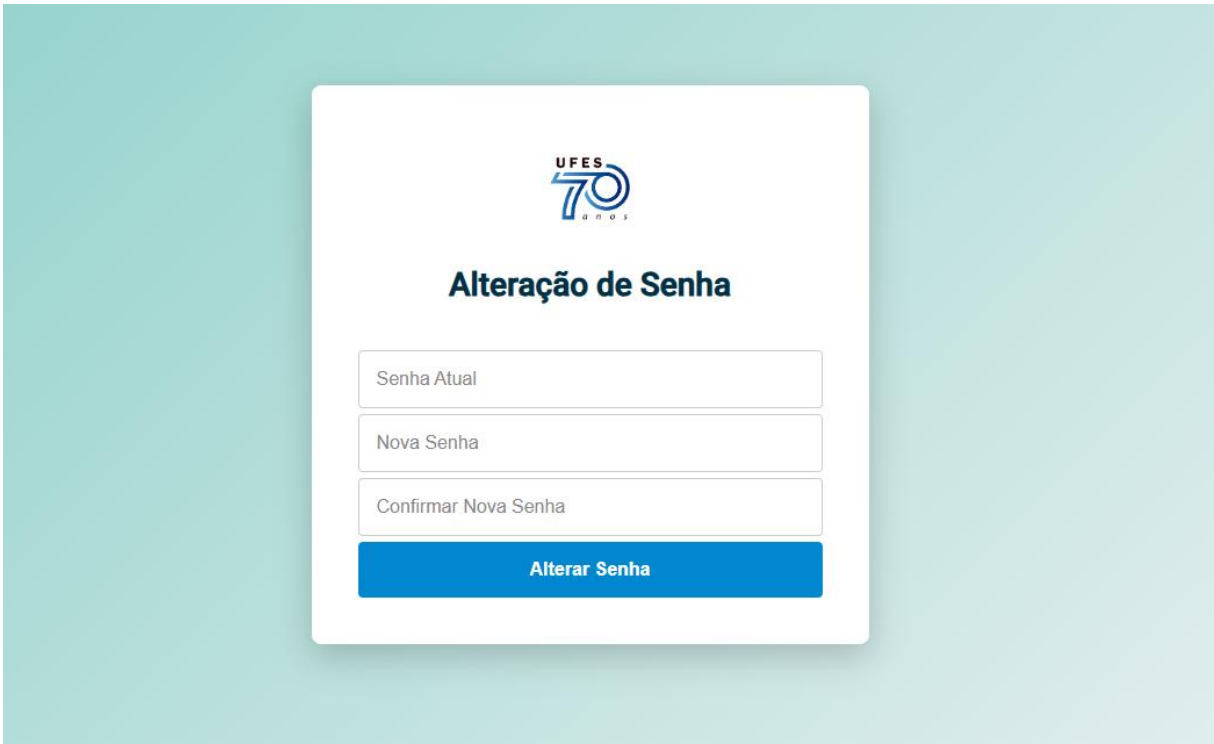
Figura 29 – Informações do usuário autenticado.



Fonte: O autor (2025).

O usuário autenticado pode encerrar sua seção na aplicação clicando no botão “Sair”, sendo imediatamente redirecionado para a tela de login. Além disso, o sistema oferece a funcionalidade “Alterar dados”, por meio da qual é possível atualizar as informações de acesso. Nessa opção, o usuário pode realizar a troca de senha, devendo informar a senha atual, a nova senha e a confirmação da nova senha. A Figura 30 ilustra a tela de alteração de credenciais do usuário no sistema.

Figura 30 – Tela de alteração de senha de um usuário autenticado.



Fonte: O autor (2025).

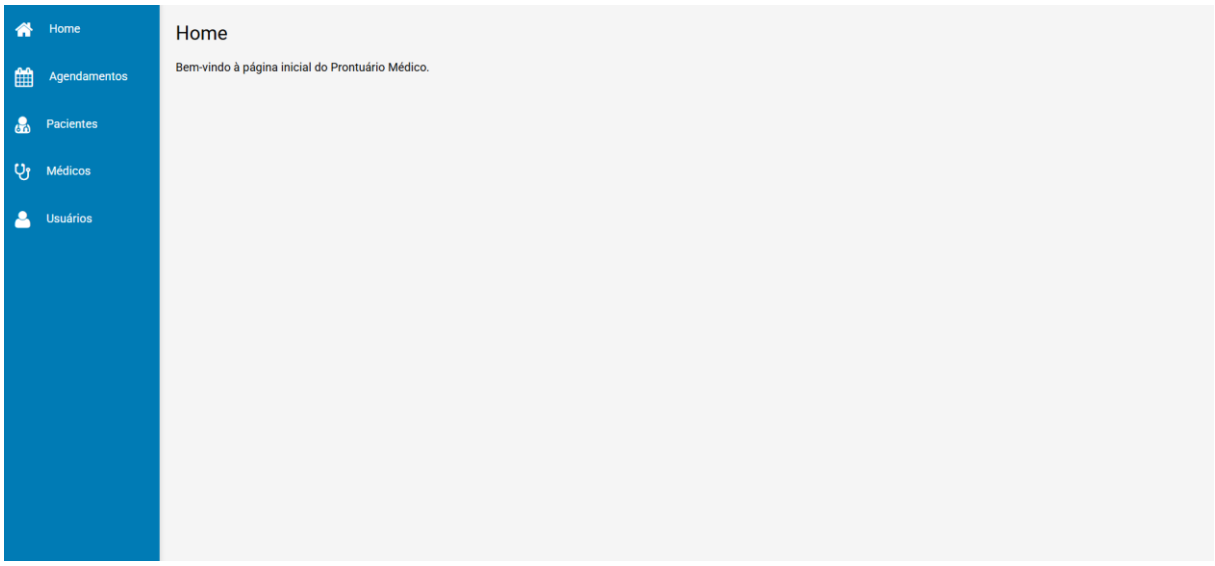
Após a alteração da senha, a sessão do usuário é encerrada automaticamente e ele é redirecionado para a tela de *login*. A partir desse momento, é necessário informar as novas credenciais para que o acesso ao sistema seja restabelecido.

Considerando que se trata de um sistema que lida com dados sensíveis, o cadastro de usuários é inicialmente realizado por meio de um usuário administrativo, previamente inserido diretamente na base de dados. A partir desse cadastro inicial, torna-se possível acessar o sistema e realizar o registro dos demais usuários.

6.1.3 Fluxo exclusivo administrativo

Após a autenticação no sistema, o menu inicial exibido para o usuário com perfil administrativo apresenta as funcionalidades disponíveis de acordo com suas permissões. A Figura 26 apresenta as opções acessíveis para a realização de operações no sistema.

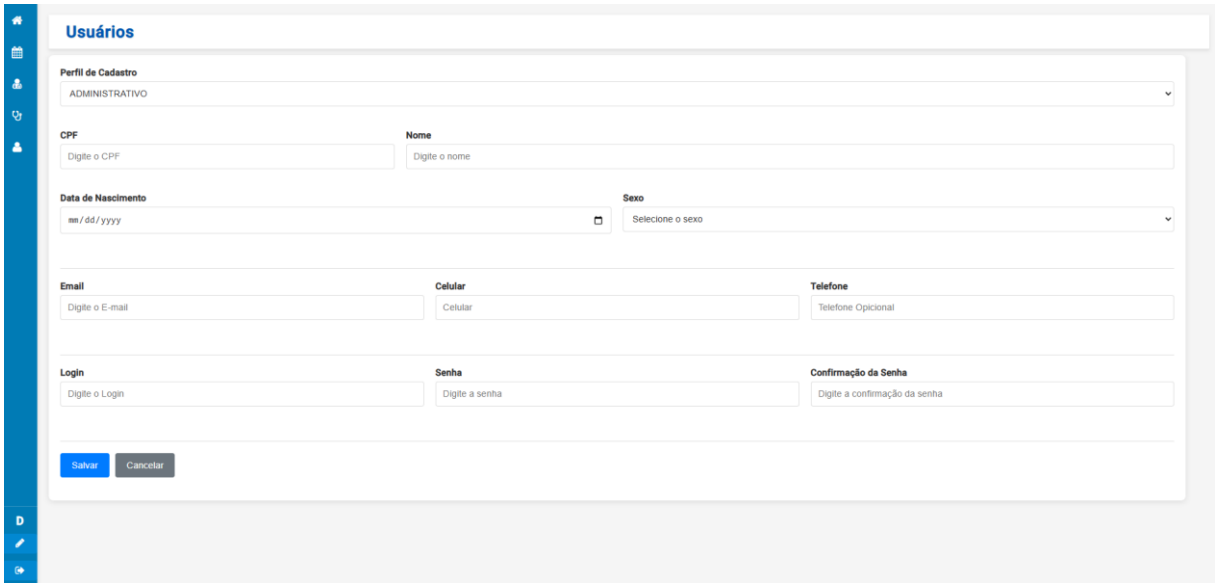
Figura 31 – Tela inicial usuário perfil administrativo.



Fonte: O autor (2025).

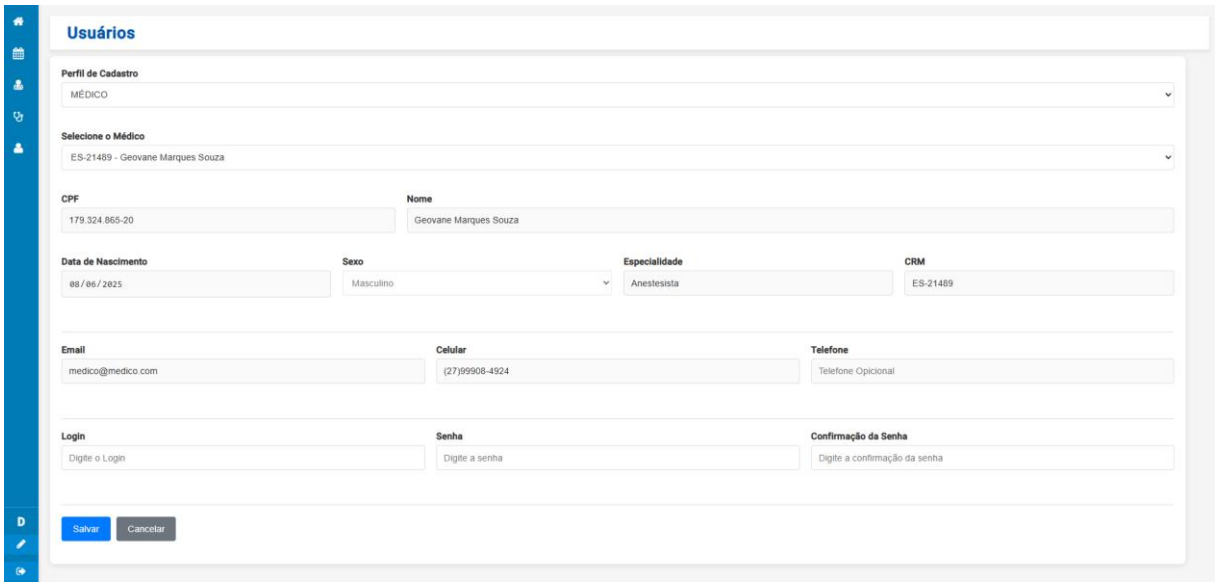
O sistema permite o cadastro de usuários em dois perfis distintos, médico e administrativo, cada um com diferentes níveis de acesso conforme as permissões atribuídas. A Figura 27 apresenta a interface de cadastro para usuários administrativos, enquanto a Figura 28 ilustra a interface destinada ao perfil médico.

Figura 32 – Tela cadastro usuário administrativo.



Fonte: O autor (2025).

Figura 33 - Tela cadastro usuário administrativo.

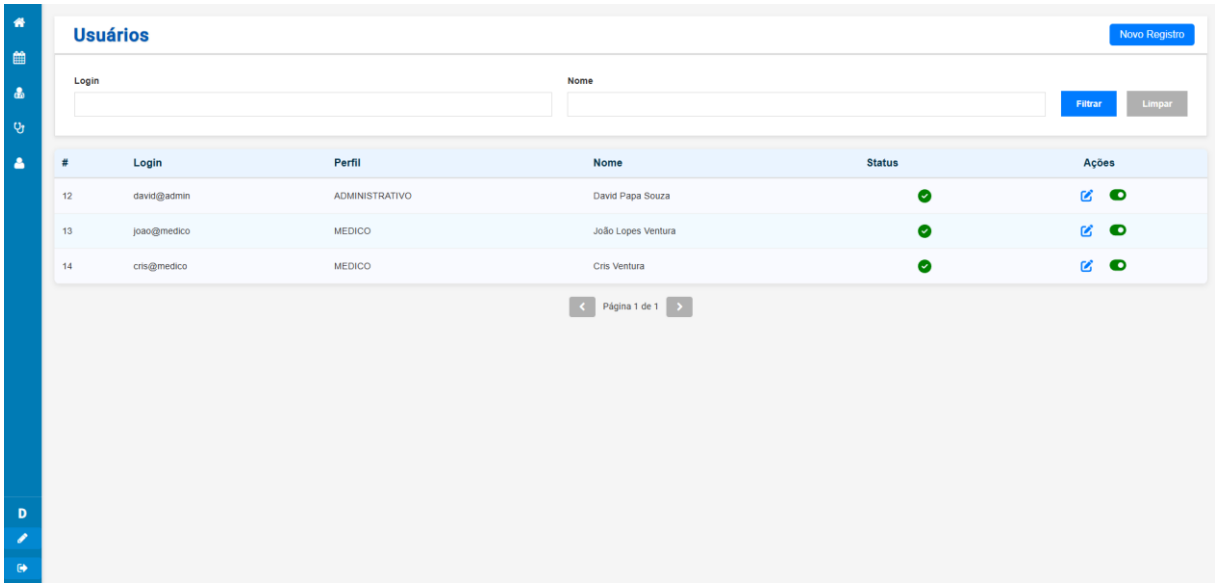


Fonte: O autor(2025).

O sistema permite a consulta de usuários previamente cadastrados, possibilitando a aplicação de filtros por *login* e/ou nome. Além disso, usuários com perfil administrativo podem

inativar um registro específico por meio da ação disponível na grade de listagem. A Figura 29 ilustra a interface dessa funcionalidade.

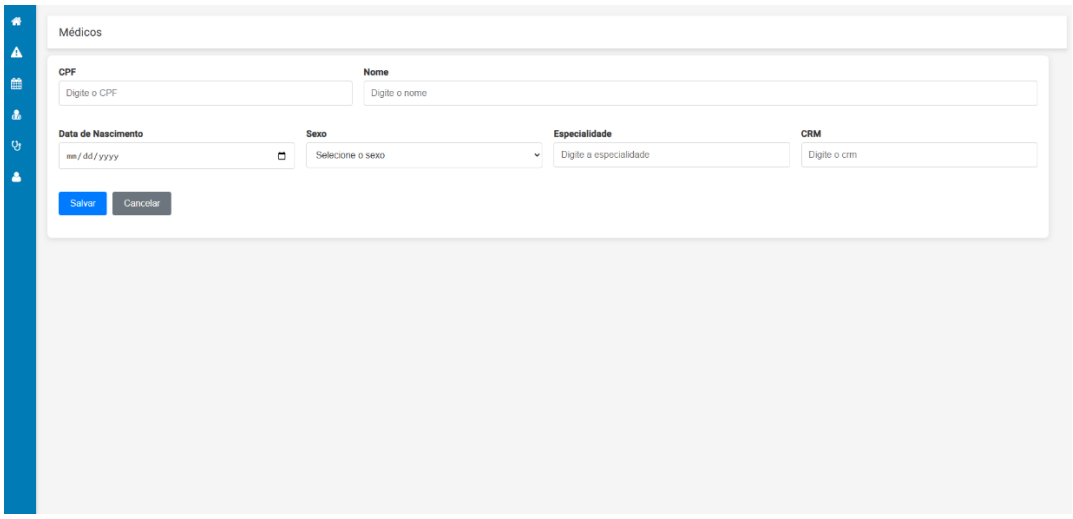
Figura 34 – Tela de consulta e inativação de usuários.



Fonte: O autor (2025).

O sistema oferece uma funcionalidade dedicada ao cadastro de médicos, permitindo o registro completo das informações necessárias para sua atuação na plataforma. Esse processo é realizado por usuários com perfil administrativo. Para o cadastro, são exigidos os dados pessoais de identificação do médico, bem como informações específicas da profissão, como o número do CRM e a especialidade exercida. É necessário que o médico esteja previamente cadastrado para que seu usuário seja devidamente criado no sistema. A Figura 30 ilustra a interface destinada a essa funcionalidade, correspondente ao caso de uso UC006, conforme detalhado no Apêndice A.

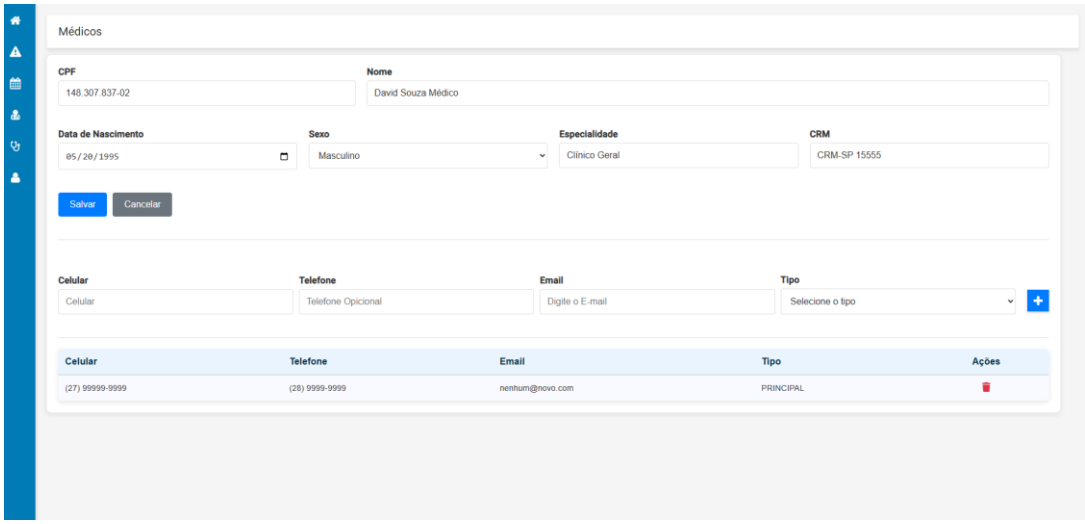
Figura 35 – Tela de cadastro de médico.



Fonte: O autor (2025).

Após o preenchimento e salvamento dos dados pessoais do médico, de forma semelhante ao processo realizado no cadastro de pacientes, o sistema permite complementar o registro com informações adicionais, como os dados de contato. A Figura 31 apresenta a interface exibida após o salvamento das informações pessoais do médico.

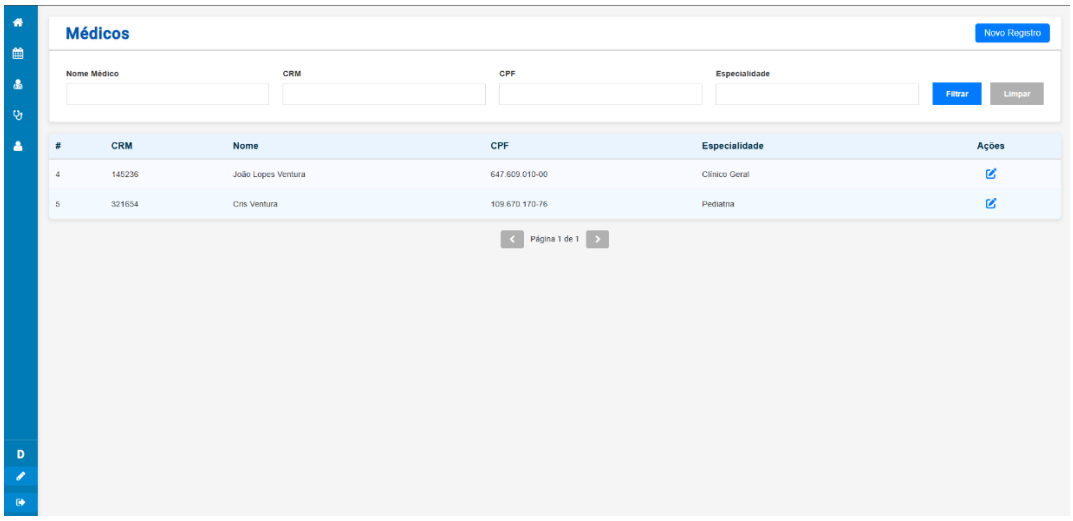
Figura 36 – Tela edição e complemento de dados médico



Fonte: O autor (2025).

Além das funcionalidades de cadastro e alteração, o sistema permite a consulta de médicos previamente registrados, com a possibilidade de aplicar filtros por nome, CRM, CPF ou especialidade. A Figura 37 apresenta a interface dessa funcionalidade.

Figura 37 – Tela de consulta de médicos.



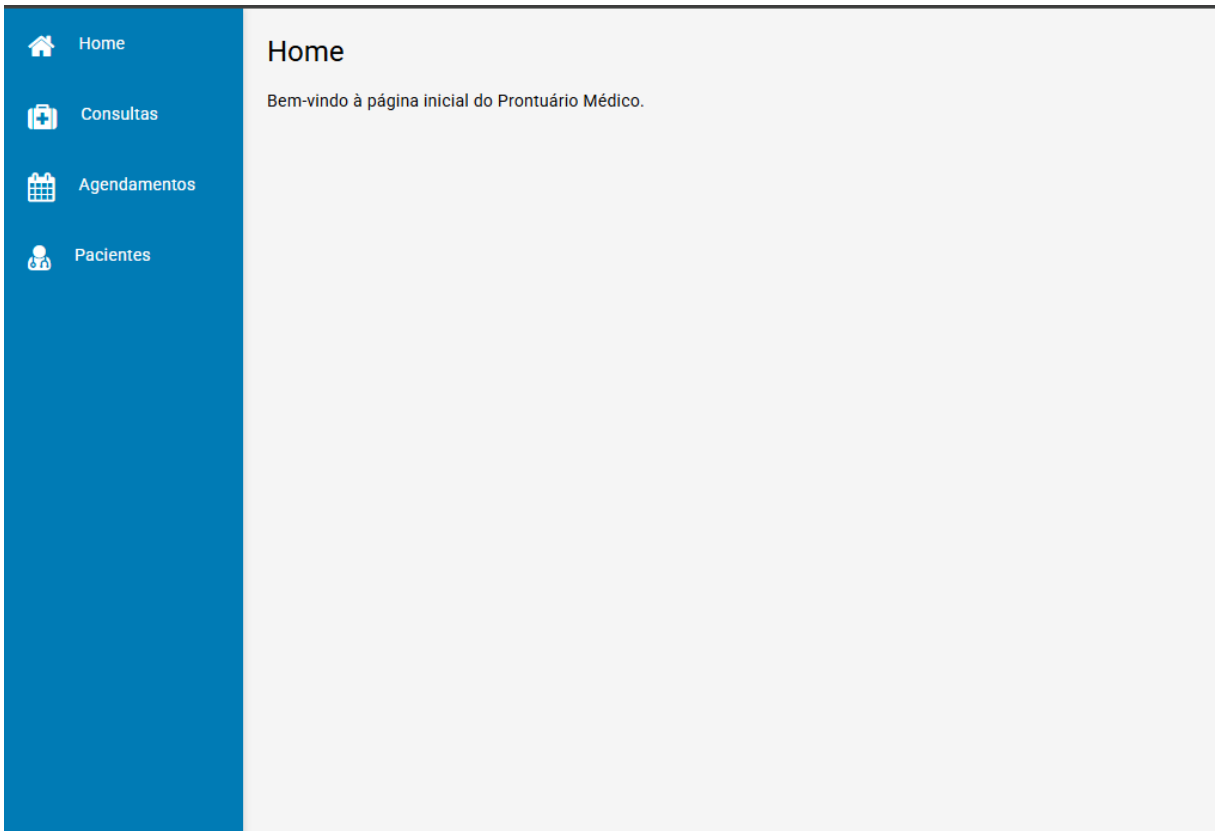
Fonte: O autor (2025).

Dessa forma, concluem-se as funcionalidades disponibilizadas ao perfil administrativo no sistema.

6.1.4 Fluxo exclusivo médico

Após a autenticação no sistema, o menu inicial exibido para o usuário com perfil de médico apresenta as funcionalidades disponíveis de acordo com suas permissões. A Figura 38 apresenta as opções acessíveis para a realização de operações no sistema.

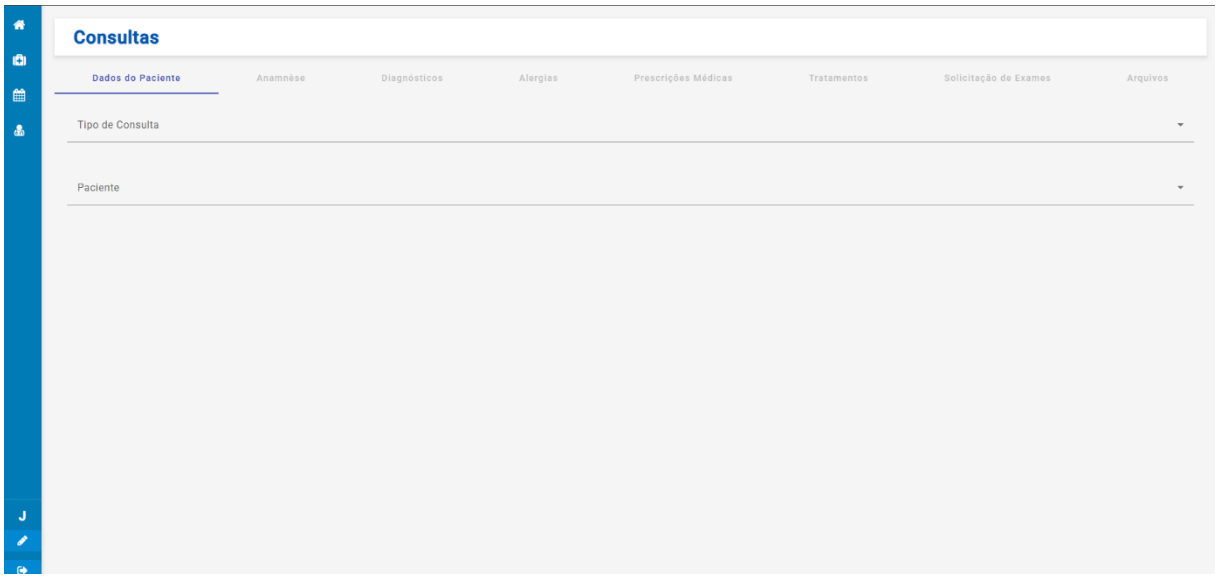
Figura 38 – Tela inicial usuário autenticado como médico.



Fonte: O autor (2025).

Entre as funcionalidades disponíveis para usuários com perfil médico, destaca-se o gerenciamento de consultas, que representa a principal atividade desempenhada nesse perfil. Por meio dessa funcionalidade, o médico pode iniciar o atendimento ao paciente e, a partir disso, realizar diversas ações que contribuem para o diagnóstico e para a organização da consulta. A Figura 39 ilustra a interface da tela de consulta na perspectiva do médico.

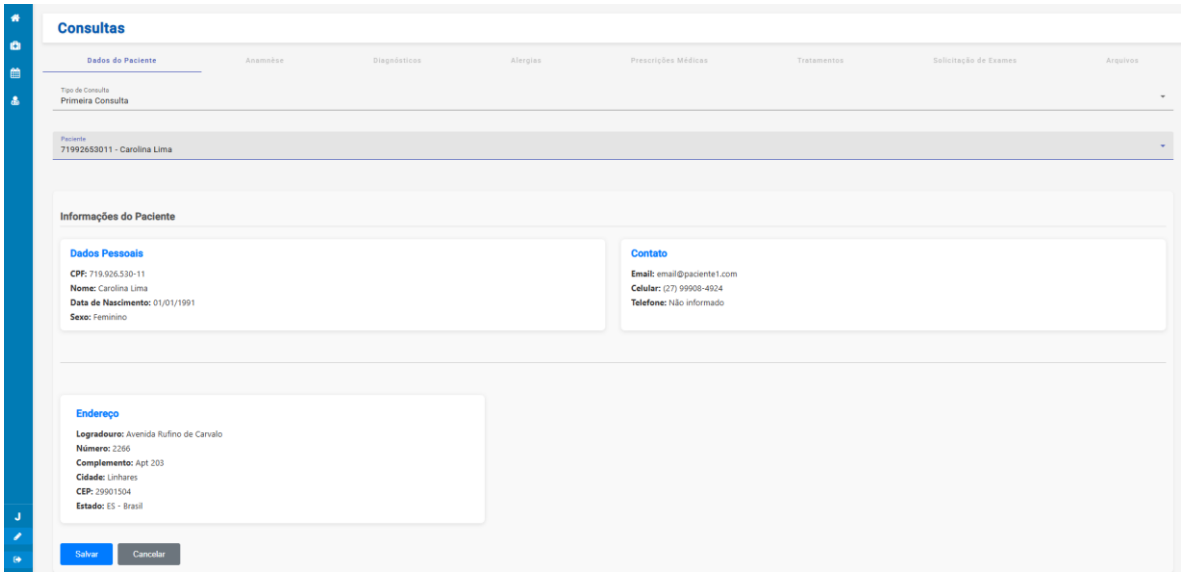
Figura 39 – Tela de cadastro de consulta, etapa dados paciente.



Fonte: O autor (2025).

As etapas da consulta médica no sistema proposto estão organizadas de forma sequencial e estruturada, iniciando-se pela identificação do paciente. Nesse momento, o profissional de saúde localiza o paciente previamente cadastrado e classifica o tipo de consulta a ser realizada. Ao selecionar o paciente, suas informações são apresentadas em formato de resumo e, após a confirmação, não será mais possível alterá-las. Essa etapa é fundamental para assegurar que o prontuário seja corretamente vinculado e que o atendimento siga o fluxo adequado. A Figura 40 apresenta a interface correspondente a essa funcionalidade.

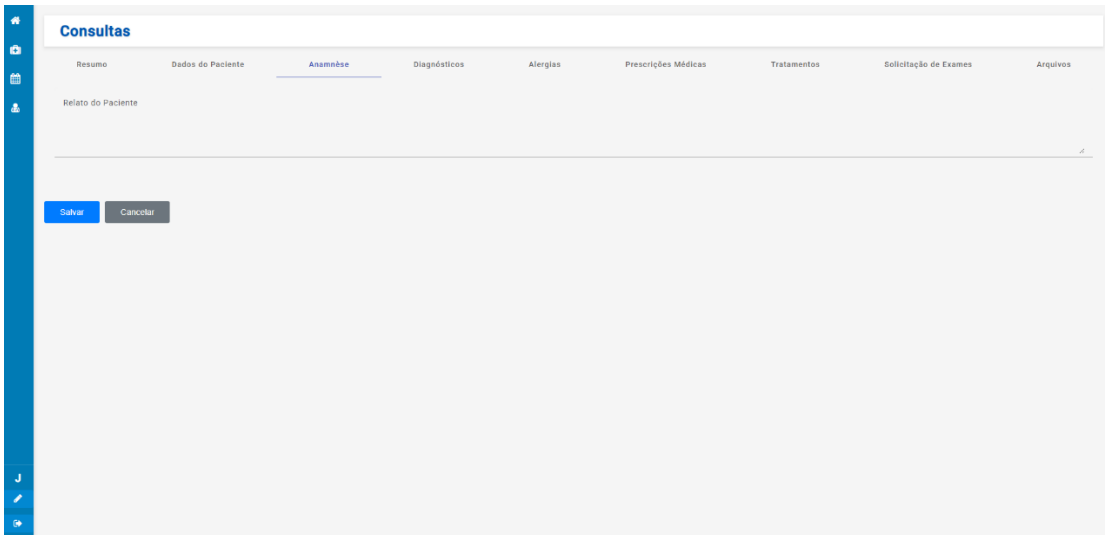
Figura 40 – Tela de cadastro de consulta, selecionando um paciente.



Fonte: O autor (2025).

Na sequência, a consulta evolui para a etapa de anamnese, em que o paciente relata, de forma livre ou guiada, os sintomas, desconfortos e histórico recente relacionados à sua condição clínica. O sistema disponibiliza um campo específico para o registro desse relato, permitindo ao médico compor uma visão inicial do quadro clínico apresentado. A Figura 41 apresenta a interface correspondente a funcionalidade de cadastro da anamnese.

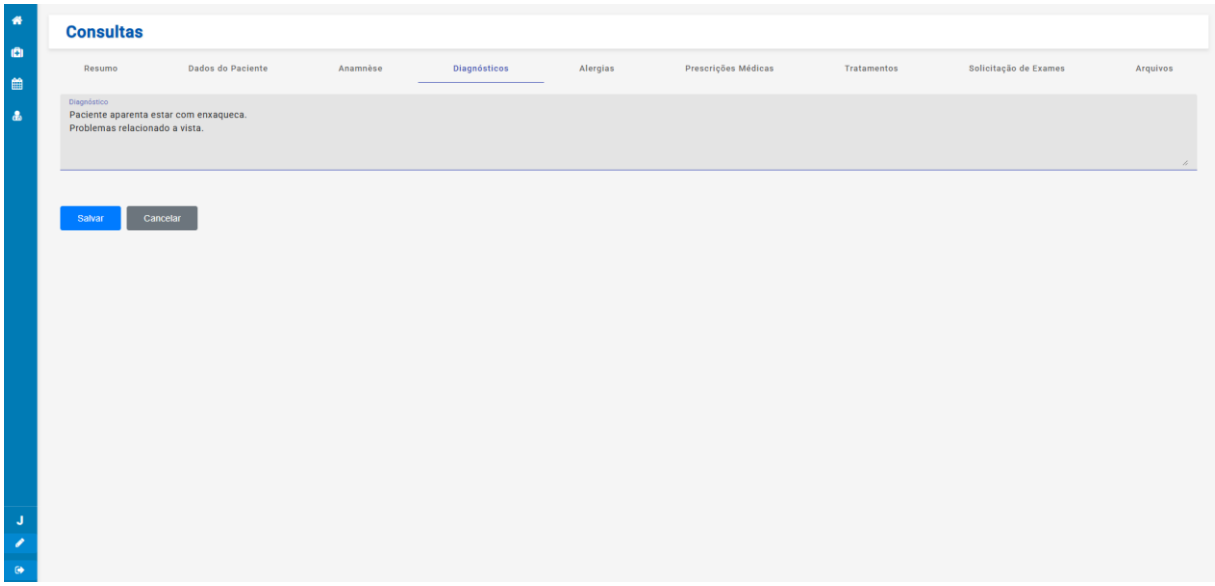
Figura 41 – Tela de cadastro de anamnese.



Fonte: O autor (2025).

A etapa de diagnóstico sucede a anamnese e tem como objetivo o registro das observações clínicas do profissional de saúde. Nesse momento, são documentados os achados relevantes do atendimento e as possíveis doenças associadas, garantindo um histórico clínico preciso e consistente. Essa etapa é fundamental para subsidiar decisões terapêuticas e manter a integridade das informações do prontuário. A Figura 42 exemplifica a interface destinada ao cadastro do diagnóstico.

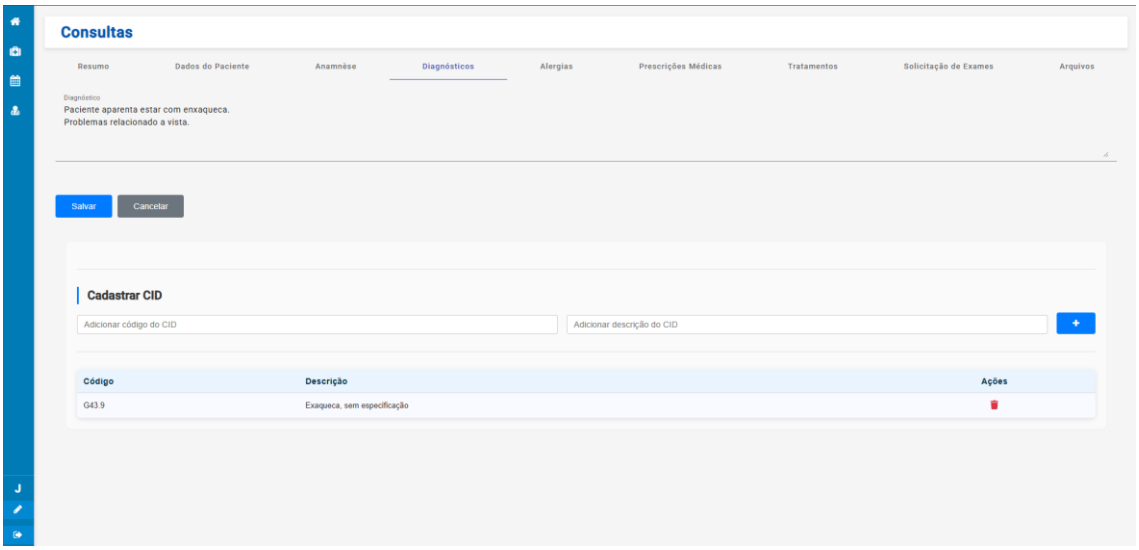
Figura 42 – Tela de cadastro de diagnóstico.



Fonte: O autor (2025).

Complementarmente, o sistema permite o registro dos códigos da Classificação Internacional de Doenças (CID) relacionados ao diagnóstico definido. Essa funcionalidade assegura a padronização das informações médicas, favorecendo a interoperabilidade e a conformidade com normas de registro em saúde. A Figura 43 apresenta a tela utilizada para o cadastro do CID.

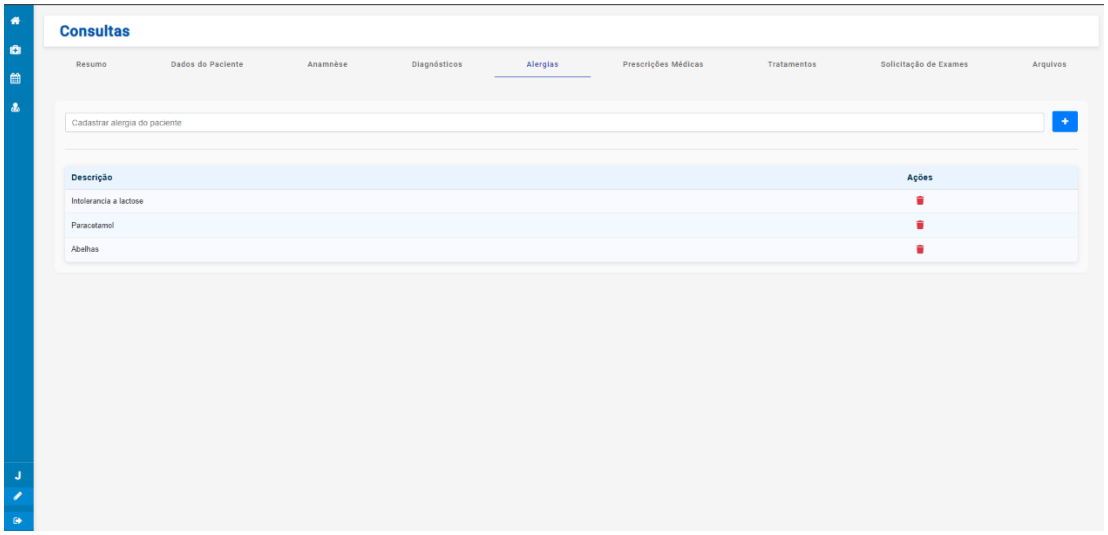
Figura 43 – Cadastro de CIDs.



Fonte: O autor (2025).

Em seguida, a etapa de alergias permite o registro de qualquer sensibilidade ou reação adversa a substâncias, medicamentos ou alimentos, assegurando a segurança do paciente durante as etapas posteriores do atendimento, especialmente na prescrição médica. As alergias cadastradas na etapa de cadastro do paciente são refletidas nesta parte da consulta e vice-versa. Assim é garantido que nenhuma informação referente ao atendimento seja desconsiderada. A Figura 44 ilustra o cadastro de alergia na etapa de consulta.

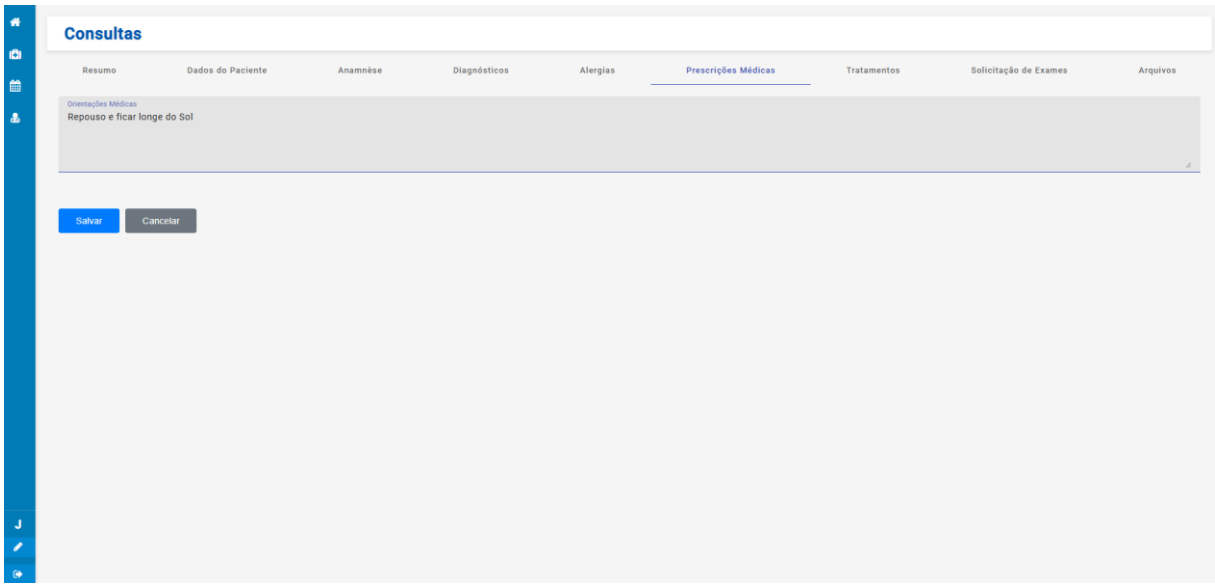
Figura 44 – Cadastro de alergias.



Fonte: O autor (2025).

A etapa de prescrição médica no sistema é destinada ao registro das indicações do profissional de saúde quanto ao uso de medicamentos pelo paciente. Nessa fase, são inseridos os dados necessários para orientar de forma clara a utilização do medicamento prescrito, assegurando que as informações fiquem documentadas de maneira padronizada no prontuário. A Figura 45 ilustra o fluxo observado para a realização da prescrição médica no sistema.

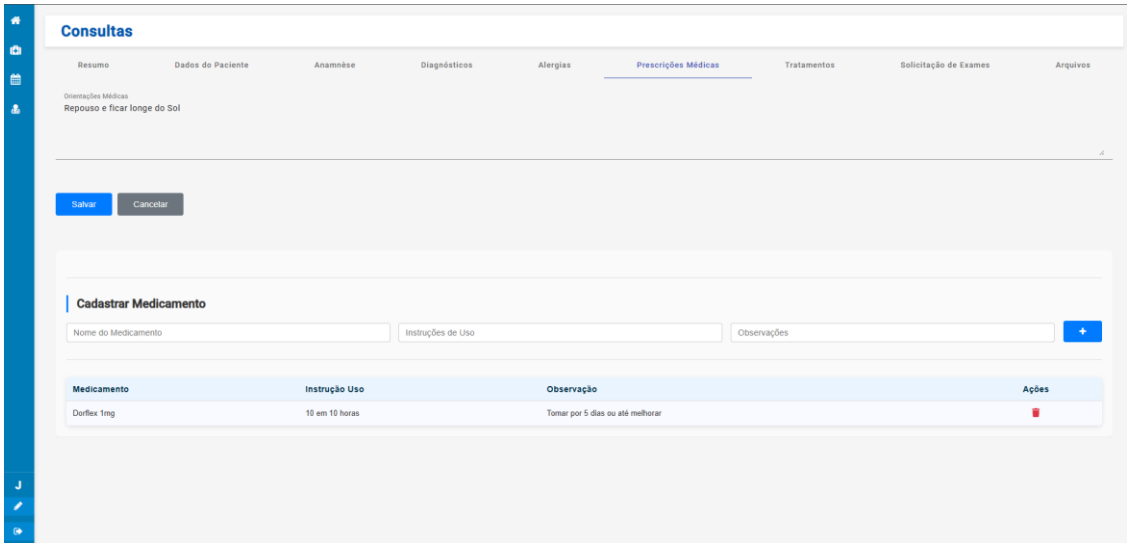
Figura 45 – Cadastro de prescrições médicas.



Fonte: O autor (2025).

No que se refere aos medicamentos, o sistema permite detalhar as orientações de uso, como dosagem, frequência e duração, bem como registrar observações adicionais relevantes à administração segura do fármaco. Esse recurso garante a clareza das instruções fornecidas ao paciente e facilita o acompanhamento pelo profissional de saúde. A Figura 46 apresenta o fluxo observado para o registro de medicamentos, orientações de uso e observações no sistema.

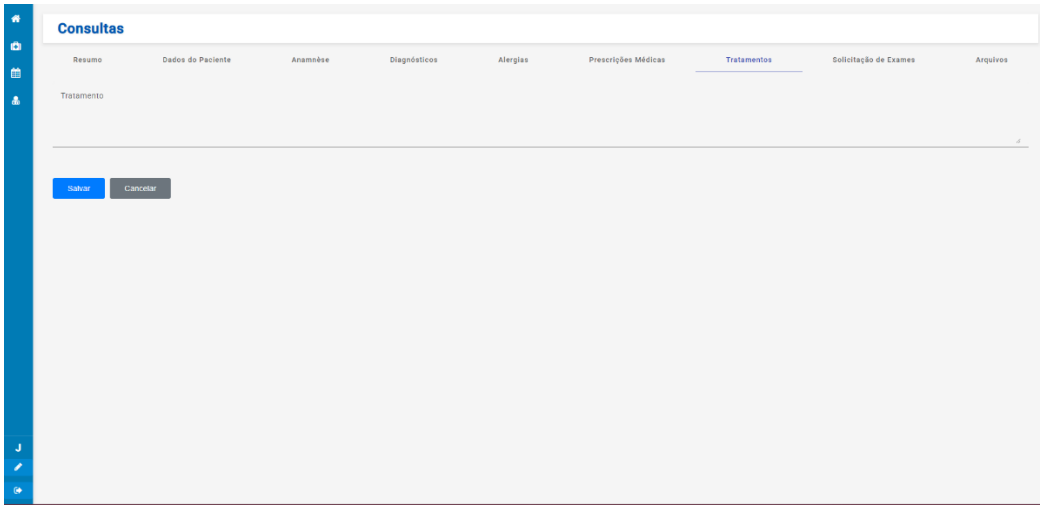
Figura 46 – Cadastro de medicamentos.



Fonte: O autor (2025).

O médico pode registrar as orientações terapêuticas na seção de tratamento, especificando o tipo de intervenção proposta, sua duração, frequência e eventuais observações pertinentes. Essa etapa é essencial para organizar e documentar as condutas indicadas, garantindo que o acompanhamento clínico ocorra de forma estruturada e segura. A Figura 47 apresenta o fluxo observado para o registro do tratamento no sistema.

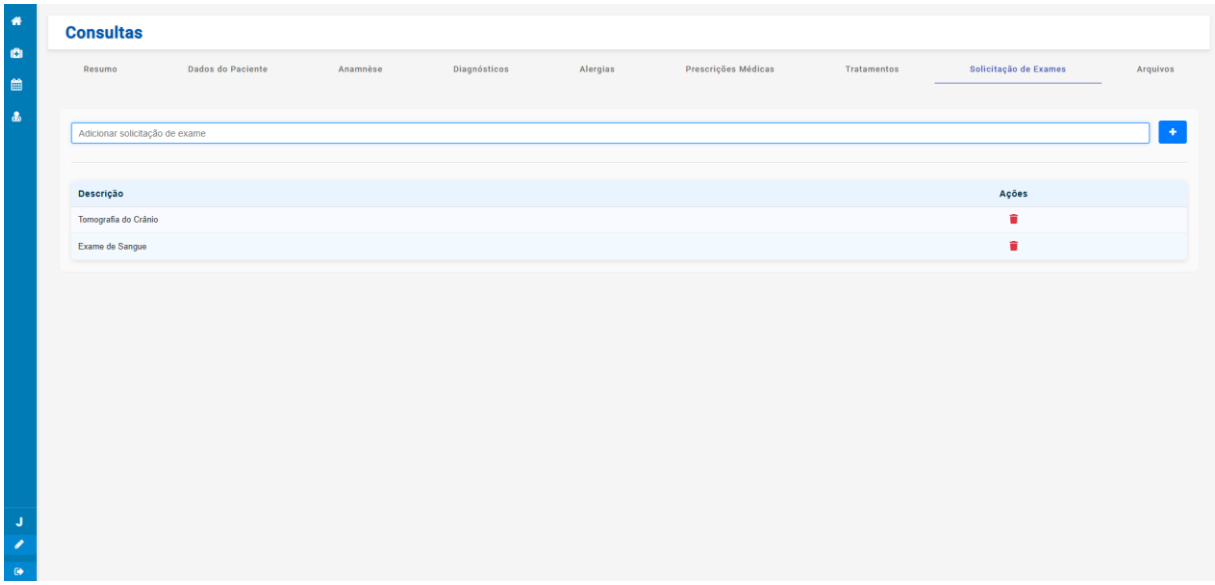
Figura 47 – Cadastro de tratamentos.



Fonte: O autor (2025).

Na etapa de solicitações de exames, o sistema permite que o médico registre os exames a serem realizados pelo paciente durante a consulta. Nessa funcionalidade, o profissional insere uma descrição detalhada que identifique claramente o exame solicitado. Essa etapa contribui para a organização do histórico clínico e para a integração das informações no prontuário eletrônico. A Figura 48 ilustra o fluxo observado para o cadastro de solicitações de exames no sistema.

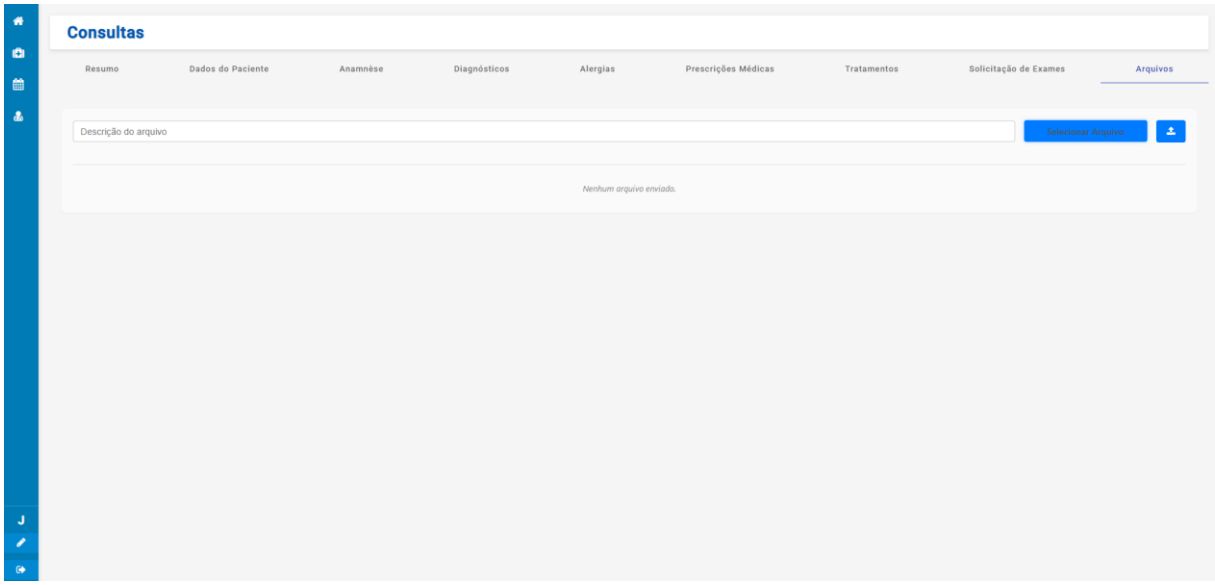
Figura 48 – Solicitação de exames da consulta.



Fonte: O autor (2025).

Na última etapa, o sistema permite o controle dos artefatos gerados durante a consulta, permitindo a anexação de arquivos pertinentes, como resultados de exames, atestados ou quaisquer outros documentos relacionados ao contexto do atendimento. Essa funcionalidade assegura que todos os registros complementares fiquem organizados e vinculados ao prontuário do paciente. A Figura 49 apresenta o fluxo observado para o gerenciamento de artefatos da consulta no sistema.

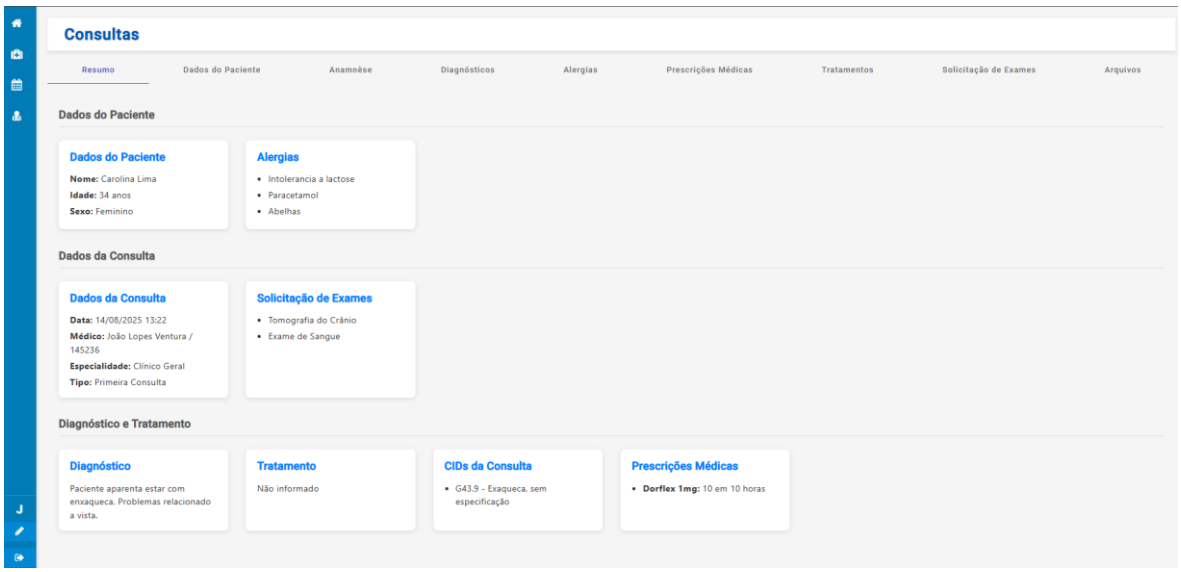
Figura 49 – Tela de *upload* de arquivos referentes a consulta.



Fonte: O autor (2025).

Por fim, no menu superior, na navegação por abas, é possível acessar a seção Resumo da consulta. Nessa área, o usuário pode visualizar as principais informações registradas ao longo do atendimento, reunindo de forma organizada os dados essenciais para consulta rápida. A Figura 50 apresenta a interface dessa funcionalidade.

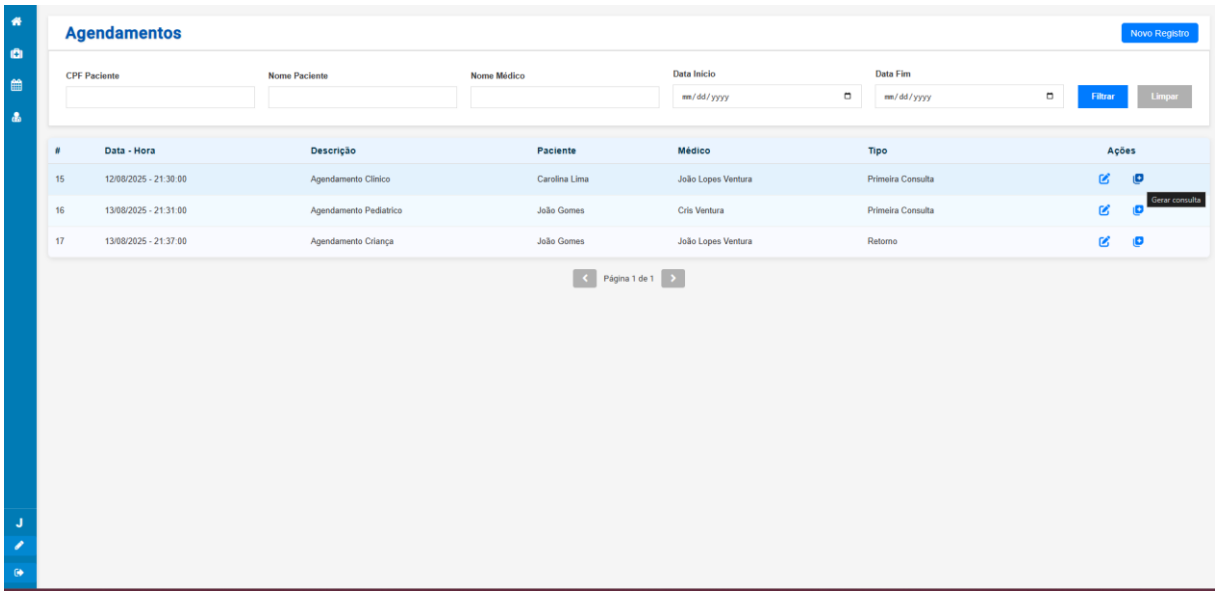
Figura 50 – Tela de resumo da consulta.



Fonte: O autor (2025).

Apesar de a funcionalidade de agendamento estar disponível para ambos os perfis de usuário, o perfil de médico possui uma opção adicional no campo de ações referente aos agendamentos. Conforme ilustrado na Figura 51, é possível gerar uma consulta a partir de um agendamento previamente registrado. Essa funcionalidade está restrita a médicos autenticados no sistema.

Figura 51 – Tela de consulta de agendamentos, botão gerar consulta.



Fonte: O autor (2025).

Dessa forma, concluem-se as funcionalidades disponibilizadas ao perfil de médico no sistema.

7 CONCLUSÕES E CONSIDERAÇÕES FINAIS

A implementação do sistema de prontuário eletrônico médico desenvolvido neste trabalho demonstrou-se eficaz em atender ao objetivo proposto, que era disponibilizar uma solução informatizada capaz de organizar e estruturar informações clínicas no contexto do SASAS/UFES. Os resultados técnicos evidenciaram que a aplicação alcançou um nível satisfatório de robustez, apresentando tempo de resposta inferior a dois segundos para operações de cadastro, consulta e atualização, ausência de falhas em cem execuções consecutivas de operações críticas e 100% de consistência dos dados armazenados e recuperados no banco de dados PostgreSQL. Esses indicadores confirmam o cumprimento dos requisitos não funcionais levantados e sustentam a viabilidade prática da solução.

O sistema atendeu aos objetivos específicos definidos, ao disponibilizar interface intuitiva acessível via navegador, gerenciamento integrado de dados de pacientes e profissionais, registro de consultas, exames e tratamentos, bem como mecanismos de autenticação e controle de acesso por perfil. Dessa forma, o trabalho alcançou tanto a contribuição técnica, com a entrega de um *software* funcional, seguro e estável, quanto a contribuição acadêmica, representada pelo processo de levantamento de requisitos, análise crítica de sistemas de mercado e aplicação de conceitos de engenharia de *software* em um problema real.

No entanto, algumas limitações foram observadas. A ausência de testes com o médico do setor restringiu a avaliação em termos de usabilidade e aceitação prática. Também não foi possível realizar comparações diretas com outros sistemas de mercado, o que limitou a análise a um nível teórico. Por fim, o sistema ainda não foi integrado às demais plataformas institucionais da UFES, o que reduz, por ora, seu potencial de interoperabilidade no contexto universitário.

Essas limitações abrem caminho para trabalhos futuros, como a realização de testes pilotos com médicos e pacientes, a validação em longo prazo em diferentes cenários de uso, a integração com sistemas acadêmicos e administrativos já existentes e a ampliação da análise comparativa com outras soluções disponíveis no mercado. Dessa forma, o sistema poderá evoluir não apenas como produto tecnológico, mas também como objeto de estudo em pesquisas futuras relacionadas à usabilidade, interoperabilidade e gestão da informação em saúde no ambiente acadêmico.

Finalmente, destaca-se que o desenvolvimento do projeto permitiu a aplicação prática dos conhecimentos adquiridos ao longo do curso, abrangendo desde a modelagem de banco de dados até práticas de programação, segurança da informação e documentação técnica. O código-fonte está disponível publicamente para acesso, estudo e evolução, podendo ser consultado nos repositórios: *back-end* (https://github.com/oDavidPapa/prontuario) e *front-end* (https://github.com/oDavidPapa/prontuario-web).

8 CRONOGRAMA

O cronograma foi estruturado considerando as fases de desenvolvimento do sistema em paralelo às etapas de elaboração do TCC, prevendo eventuais ajustes conforme a necessidade de adaptações ao progresso incremental do projeto.

Descrição das atividades.

ETAPA 1	Revisão da literatura
ETAPA 2	Levantamento de requisitos
ETAPA 3	Definição dos casos de uso
ETAPA 4	Documentação e especificação
ETAPA 5	Modelagem do projeto
ETAPA 6	Codificação do sistema
ETAPA 7	Testes
ETAPA 8	Correções e adequações
ETAPA 9	Entrega final e Apresentação

Cronograma detalhado.

	DEZ 2024	JAN 2025	FEV 2025	MAR 2025	ABRIL 2025	MAIO 2025	JUNHO 2025	JULHO 2025	AGO 2025
ETAPA 1	X	X							
ETAPA 2	X	X							
ETAPA 3		X	X						
ETAPA 4			X						
ETAPA 5			X	X					
ETAPA 6				X	X	X	X	X	
ETAPA 7							X	X	
ETAPA 8								X	
ETAPA 9									X

REFERENCIAS

ALMEIDA, M. de; SOUZA, P. R. **Gestão de projetos de software**. 2. ed. São Paulo: Atlas, 2020.

AMAZON WEB SERVICES (AWS). **O que é um framework?**. Disponível em: https://aws.amazon.com/pt/what-is/framework/. Acesso em: 24 mar. 2025.

AMAZON WEB SERVICES (AWS). **O que é uma API?**. Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 24 mar. 2025.

AMAZON WEB SERVICES (AWS). SQL on AWS – **O que é SQL?**. Disponível em: https://aws.amazon.com/pt/what-is/sql/. Acesso em: 24 mar. 2025.

ANDERSON, David J. **Kanban**: successful evolutionary change for your technology business. Blue Hole Press, 2010.

ANGULAR. **What is Angular?.** Disponível em: https://angular.io/guide/what-is-angular. Acesso em: 24 mar. 2025.

ASTAH. Astah UML. 2024. Disponível em: https://astah.com. Acesso em: 15 dez. 2024.
BECK, Kent. **Test Driven Development: By Example**. Addison-Wesley Professional, 2003.

BAKULINA, Olga. **O que é Sistema Kanban?** Blog Kanbanchi, 14 fev. 2023. Kanbanchi. Disponível em: https://www.kanbanchi.com/pt/blog/o-que-e-sistema-kanban. Acesso em: 15 jul. 2025. Wikipedia+5Kanbanchi+5Kanbanchi+5

BETHA. **Betha Saúde**. Disponível em: https://betha.store/produto/saude. Acesso em: 23 mar. 2025.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML – **Guia do Usuário**. 2. ed. Rio de Janeiro: Elsevier, 2005.

BOOTSTRAP. **Introduction** – Bootstrap v5.3 Documentation. Disponível em: https://getbootstrap.com/docs/5.3/getting-started/introduction/. Acesso em: 24 mar. 2025.

BRASIL. [Constituição (1988)]. **Constituição da República Federativa do Brasil.** Brasília: Senado Federal, 2009. Disponível em: https://bvsms.saude.gov.br/bvs/saudelegis/gm/2009/prt1820_13_08_2009.html. Acesso em: 15 mar. 2025.

CARVALHO, André de Oliveira; EDUARDO, Maria Bernadete de Paula. **Sistemas de informação em saúde para municípios**. 1998. p. 112-112.

CLÁUDIA, A.; KLUTHCOVSKY, G.; MAGOSSO TAKAYANAGUI, **A. Quality of life conceptual issues**. Disponível em: http://www.observatorionacionaldoidoso.fiocruz.br/biblioteca/_artigos/12.pdf. Acesso em: 9 mar. 2025.

COCKBURN, A. **Escrevendo Casos de Usos Eficazes**: Um guia prático para desenvolvedores de software. Porto Alegre: Bookman, 2005.

CUNHA, Gabriela Ribas da; CUNHA, Rogério Dias da. **Prontuário eletrônico do paciente***:* avaliação da situação atual e perspectivas. Revista de Saúde Pública, São Paulo, v. 43, n. 3, p. 523-530, jun. 2009.

DATASUS. **Ferramenta CASE**. Disponível em: https://datasus.saude.gov.br/glossario/ferramenta-case/. Acesso em: 24 mar. 2025.

DATASUS. **Sistema Gerenciador de Banco de Dados (SGBD)**. Disponível em: https://datasus.saude.gov.br/glossario/sistema-gerenciador-de-banco-de-dados-sgbd/. Acesso em: 24 mar. 2024.

ESCUDÉLARIO, Bruna de Freitas. **Encriptando senhas com o Bcrypt.** iMasters, 2023. Disponível em: https://imasters.com.br/devsecops/encriptando-senhas-com-o-bcrypt. Acesso em: 2 jul. 2025.

FARINA, A. **Prontuário Médico**. Disponível em: https://portal.cfm.org.br/artigos/prontuario-medico/. Acesso em: 7 mar. 2025.

FOWLER, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

FOWLER, Martin. **Continuous Integration. ThoughtWorks**, 2012. Disponível em: https://martinfowler.com/articles/continuousIntegration.html. Acesso em: 9 ago. 2025.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns**: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

GLOBAL CARE. **Pesquisa revela as principais doenças crônicas no Brasil**. Disponível em: https://www.globalcare.com.br/br/blog/pesquisa-revela-as-principais-doencas-cronicas-no-brasil. Acesso em: 10 mar. 2025.

IBM. **Java Spring Boot.** Disponível em: https://www.ibm.com/br-pt/topics/java-spring-boot. Acesso em: 24 mar. 2024.

IMPACT PLUS. **Agile marketing using scrum**. 2024. Disponível em: https://www.impactplus.com/blog/agile-marketing-using-scrum. Acesso em: 15 dez. 2024.

JAVA**. What is Java technology and why do I need it?.** Disponível em: https://www.java.com/. Acesso em: 24 mar. 2025.

MATTOS, Antonio Carlos M. **Sistemas de informação**. São Paulo: Saraiva Educação SA, 2017.

MEC/SEF. **Parâmetros curriculares nacionais:** terceiro e quarto ciclos do ensino fundamental. Brasília: MEC, 1998. Disponível em: http://portal.mec.gov.br/seb/arquivos/pdf/saude.pdf. Acesso em: 12 mar. 2024.

MEDITECH. **Expanse for Physicians**. Disponível em: https://ehr.meditech.com/ehrsolutions/expanse-for-physicians. Acesso em: 23 mar. 2025.

MICROSOFT AZURE. **O que é PostgreSQL?.** Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-postgresql/. Acesso em: 24 mar. 2025.

MINAYO, M. C. de S.; HARTZ, Z. M. de A.; BUSS, P. M. **Qualidade de vida e saúde**: um debate necessário. Ciência & Saúde Coletiva, v. 5, p. 7–18, 2000.

MINISTÉRIO DA EDUCAÇÃO (MEC). **Hospitais Universitários.** Disponível em: http://portal.mec.gov.br/hospitais-universitarios. Acesso em: 7 mar. 2025.

MOZILLA DEVELOPER NETWORK (MDN). **JSON**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/JSON. Acesso em: 24 mar. 2025.

MOZILLA DEVELOPER NETWORK (MDN). **O que é JavaScript?**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 24 mar. 2025.

Mozilla Contributors. HTML5 — living standard. The HTML specification. MDN Web Docs, 11 jul. 2025. Disponível em: MDN Web Docs — HTML5. Acesso em: 30 ago. 2025.

NINSAUDE. **Prontuário Eletrônico:** Software Médico. Disponível em: https://www.ninsaude.com/pt-br/prontuario-eletronico-software-medico/. Acesso em: 23 mar. 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software:** uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.
RESTFUL API. RESTful API. Disponível em: https://restfulapi.net/. Acesso em: 24 mar. 2025.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação**. Rio de Janeiro: Brasport, 2006.

SOARES, M. S.; CALDEIRA, Vagner**. Metodologias Ágeis**: Extreme Programming. 2004.

SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, L. **Pesquisa**: 80% dos brasileiros estão preocupados com o acesso à saúde. Disponível em: https://agenciabrasil.ebc.com.br/saude/noticia/202111/pesquisa-80-dos-brasileiros-estao-preocupados-com-acesso-saude. Acesso em: 12 mar. 2025.

SPRING FRAMEWORK (SPRING). **Documentação do Spring Framework 3.2.x**. Disponível em: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html. Acesso em: 24 mar. 2024.
SPRING. Spring Data JPA. Disponível em: https://spring.io/projects/spring-data-jpa. Acesso em: 24 mar. 2025.

SPRING.IO. Spring Security Reference Documentation. 2024. Disponível em: https://docs.spring.io/spring-security/reference/. Acesso em: 2 jul. 2025.

TRELLO. **Trello**. 2024. Disponível em: https://trello.com. Acesso em: 15 dez. 2024.

TYPESCRIPT. **TypeScript é JavaScript com sintaxe para tipos**. Disponível em: https://www.typescriptlang.org/. Acesso em: 24 mar. 2025.

VAROTO, Ane Cristina. **Visões em arquitetura de software**. 2002. Tese (Doutorado) – Universidade de São Paulo, São Paulo.

APÊNDICE A - NARRATIVAS DE CASOS DE USO

A.1 - UC003 - Cadastrar endereço do paciente

Descrição: Processo de inserção de endereço do paciente.

Atores: Usuário (Médico/Administrativo).

Pré-condições: Usuário autenticado e paciente cadastrado.

Pós-condições: Um novo endereço para o paciente é registrado no sistema.

Fluxo Principal:

[FP03] - Cadastro de endereço do paciente.

01 - O sistema exibe a tela de cadastro de paciente.

02 - O usuário preenche o Logradouro do endereço.

03 - O usuário preenche o Número do endereço.

04 - O usuário preenche o Complemento do endereço.

05 - O usuário preenche o Bairro do endereço.

06 - O usuário preenche o CEP do endereço.

07 - O usuário preenche a Cidade do endereço.

08 - O usuário preenche o País do endereço.

09 - O usuário clica no botão salvar. [FE03][FA05].

10 - O sistema cadastra o endereço do paciente no banco de dados.

11 - O sistema exibe a MSG006.

Fluxo Alternativo:

[FA05] - Cancelamento do cadastro.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de pacientes.

Fluxos de Exceção:

[FE03] - Campo obrigatório não preenchido.

01 - O sistema exibe o erro MSG007 abaixo do campo obrigatório.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP03] no passo 09.

Mensagens do Sistema:

MSG006: “Endereço cadastrado com sucesso.”

MSG007: “Campo obrigatório.”

A.2 - UC004 - Cadastrar contato do paciente

Descrição: Processo de inserção de contato do paciente.

Atores: Usuário (Médico/Administrativo).

Pré-condições: Usuário autenticado e paciente cadastrado.

Pós-condições: Um novo contato para o paciente é registrado no sistema.

Fluxo Principal:

[FP04] - Cadastro de contato do paciente.

01 - O sistema exibe a tela de cadastro de contato do paciente.

02 - O usuário preenche o Celular do contato.

03 - O usuário preenche o Telefone do contato.

04 - O usuário preenche o E-mail do contato.

05 - O usuário seleciona o Tipo do contato.

06 - O usuário clica no botão [+] [FE04][FE05].

10 - O sistema cadastra o endereço do paciente no banco de dados.

11 - O sistema exibe a MSG007.

Fluxos de Exceção:

[FE04] - Campo obrigatório não preenchido.

01 - O sistema exibe a mensagem de erro abaixo do campo obrigatório.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP04] no passo 06.

[FE05] – Contato tipo principal já cadastrado.

01 - O sistema exibe a mensagem de erro MSG008.

02 – O usuário altera o tipo de contato para SECUNDÁRIO.

03 - Retorna ao [FP04] no passo 06.

Mensagens do Sistema:

MSG007: “Contato cadastrado com sucesso!”

MSG008: “Contato principal já cadastrado.”

A.3 - UC005 - Cadastrar alergia do paciente

Descrição: Processo de inserção de alergias do paciente.

Atores: Usuário (Médico/Administrativo).

Pré-condições: O usuário deve autenticado e paciente deve estar cadastrado.

Pós-condições: Uma nova alergia para o paciente é registrada no sistema.

Fluxo Principal:

[FP05] - Cadastro de alergia do paciente.

01 - O sistema exibe a tela de cadastro de alergia do paciente.

02 - O usuário preenche a descrição da alergia.

03 - O usuário clica no botão [+] [FE05].

10 - O sistema cadastra a alergia do paciente no banco de dados.

11 - O sistema exibe a MSG009.

Fluxos de Exceção:

[FE05] - Campo obrigatório não preenchido.

01 - O sistema exibe a mensagem de erro abaixo do campo obrigatório.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP05] no passo 03.

Mensagens do Sistema:

MSG009: “Alergia cadastrada com sucesso!”.

A.4. UC006 - Cadastrar médico

Descrição: Processo de inserção de médicos no sistema.

Atores: Usuário (Administrativo).

Pré-condições: O usuário autenticado no sistema.

Pós-condições: Um novo médico é registrado no sistema.

Fluxo Principal:

[FP06] - Cadastro de médicos.

01 - O sistema exibe a tela de cadastro médicos.

02 - O usuário preenche o CPF do médico.

03 - O usuário preenche o nome do médico.

04 - O usuário preenche o sexo do médico.

05 - O usuário preenche a especialidade do médico.

06 - O usuário preenche o CRM do médico.

07 - O usuário preenche a data de nascimento do médico.

08 - O usuário clica no botão salvar. [FE06][FA06].

10 - O sistema o médico no banco de dados.

10 – O sistema exibe a mensagem MSG010.

Fluxos Alternativos:

[FA06] - Cancelamento do cadastro.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de médicos.

Fluxos de Exceção:

[FE06] - Campo obrigatório não preenchido.

01 - O sistema exibe a mensagem de erro abaixo do campo obrigatório.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP06] no passo 08.

Mensagens do Sistema:

MSG010: “Médico cadastrado com sucesso!”.

A.5. UC007 - Cadastrar contato do médico

Descrição: Processo de inserção de contato para o médico.

Atores: Usuário (Administrativo).

Pré-condições: O usuário autenticado e médico deve estar cadastrado.

Pós-condições: Um novo contato para o médico é registrado no sistema.

Fluxo Principal:

[FP07] - Cadastro de contato do médico.

01 - O sistema exibe a tela de cadastro de contato do médico.

02 - O usuário preenche o Celular do contato.

03 - O usuário preenche o Telefone do contato.

04 - O usuário preenche o E-mail do contato.

05 - O usuário seleciona o Tipo do contato.

06 - O usuário clica no botão [+] [FE07][FE08].

10 - O sistema cadastra o endereço do médico no banco de dados.

11 - O sistema exibe a MSG011.

Fluxos de Exceção:

[FE07] - Campo obrigatório não preenchido.

01 - O sistema exibe a mensagem de erro abaixo do campo obrigatório.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP07] no passo 06.

[FE08] – Contato tipo principal já cadastrado.

01 - O sistema exibe a mensagem de erro MSG012.

02 – O usuário altera o tipo de contato para SECUNDÁRIO.

03 - Retorna ao [FP07] no passo 06.

Mensagens do Sistema:

MSG011: “Contato cadastrado com sucesso!”

MSG012: “Contato principal já cadastrado.”

A.6. UC008 - Cadastrar usuário

Descrição: Processo de inserção de um usuário no sistema.

Atores: Usuário (Administrativo).

Pré-condições: O usuário deve estar autenticado no sistema.

Pós-condições: Um novo usuário é registrado no sistema.

Fluxo Principal:

[FP08] - Cadastro de contato do médico.

01 - O sistema exibe a tela de cadastro de usuários.

02 - O usuário seleciona o perfil de cadastro [FA07].

03 - O usuário preenche o CPF do usuário a ser cadastrado.

04 - O usuário preenche o nome do usuário a ser cadastrado.

05 - O usuário preenche a data de nascimento do usuário a ser cadastrado.

06 - O usuário preenche o e-mail do usuário a ser cadastrado.

07 - O usuário preenche o celular do usuário a ser cadastrado.

08 - O usuário preenche o telefone do usuário a ser cadastrado.

09 - O usuário preenche o login do usuário a ser cadastrado.

10 - O usuário preenche a senha do usuário a ser cadastrado.

11 - O usuário preenche a confirmação de senha do usuário a ser cadastrado.

12 - O usuário clica no botão salvar [FA08][FE09][FE10].

10 - O sistema cadastra o usuário no banco de dados.

11 - O sistema exibe a MSG013.

Fluxos Alternativos:

[FA07] - Usuário seleciona o perfil médico.

01 - O usuário seleciona o médico para cadastrar seu usuário [FA09].

02 – O sistema preenche as informações do médico.

03 – Retorna no [FP08] passo 09.

[FA08] - Cancelamento do cadastro.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de usuários.

[FA09] - Médico não cadastrado.

01 – O usuário deve realizar os passos do UC006 para prosseguir com o cadastro do usuário do médico.

Fluxos de Exceção:

[FE09] - Campo obrigatório não preenchido.

01 - O sistema exibe a mensagem MSG014.

02 - O usuário preenche o campo obrigatório.

03 - Retorna ao [FP08] no passo 12.

[FE10] – Usuário cadastrado já existente.

01 - O sistema exibe a mensagem MSG015.

02 - Retorna ao [FP08] no passo 12.

Mensagens do Sistema:

MSG013: "Usuário cadastrado com sucesso!"

MSG014: "Campo obrigatório."

MSG015: "Usuário informado já existe."

A.7. UC009 – Ativar/Inativar usuários

Descrição: Processo de ativação/inativação de um usuário no sistema.

Atores: Usuário (Administrativo).

Pré-condições: O usuário deve estar logado no sistema.

Pós-condições: Um usuário é ativado/inativado no sistema.

Fluxo Principal:

[FP09] – Ativar/Inativar usuários.

01 - O sistema exibe a tela de listagem de usuários.

02 – O usuário clica no ícone de ativação/inativação no grid de usuários.

03 – O usuário é ativo/inativo no sistema.

03 – A mensagem MSG016 é exibida.

Mensagens do Sistema:

MSG016: "Operação realizada com sucesso!"

A.8. UC010 – Cadastrar anamnese da consulta

Descrição: Processo de inserção da anamnese do paciente na consulta.

Atores: Usuário (Médico).

Pré-condições: Usuário autenticado e consulta cadastrada.

Pós-condições: Uma nova anamnese é cadastrada .

Fluxo Principal:

[FP10] – Cadastro de anamnese da consulta.

01 – O sistema exibe a aba de anamnese na consulta do paciente.

02 – O médico escreve o relato do paciente no campo de texto disponível.

03 – O médico clica no botão salvar. [FA10]

04 – O sistema salva o relato do paciente no banco de dados.

05 – O sistema exibe a mensagem MSG017.

Fluxos Alternativos:

[FA10] - Cancelamento do cadastro.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de consultas.

Mensagens do Sistema:

MSG017: "Operação realizada com sucesso."

A.8. UC011 – Cadastrar diagnóstico da consulta

Descrição: Processo de inserção do diagnóstico do paciente na consulta.

Atores: Usuário (Médico).

Pré-condições: Usuário autenticado e consulta cadastrada.

Pós-condições: Um novo diagnóstico é cadastrado.

Fluxo Principal:

[FP11] – Cadastro do diagnóstico da consulta.

01 – O sistema exibe a aba de diagnóstico na consulta do paciente.

02 – O médico escreve o diagnóstico no campo de texto disponível.

03 – O médico clica no botão salvar [FA11].

04 – O sistema salva o diagnóstico no banco de dados.

05 – O sistema exibe a mensagem MSG018.

Fluxos Alternativos:

[FA11] - Cancelamento do cadastro.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de consultas.

Mensagens do Sistema:

MSG018: “Operação realizada com sucesso.”

A.9. UC012 – Cadastrar CIDs da consulta

Descrição: Processo de inserção dos CIDs do paciente na consulta.

Atores: Usuário (Médico).

Pré-condições: Usuário autenticado, consulta e diagnostico cadastrados.

Pós-condições: CIDs são criados o diagnóstico.

Fluxo Principal:

[FP12] – Cadastro dos CIDs da consulta.

01 – O sistema exibe a tela de cadastros de CIDs.

02 – O usuário informa o código do CID.

03 – O usuário informa a descrição do CID.

04 – O usuário clica no ícone[+] [FE11].

05 – O sistema exibe a mensagem MSG018.

Fluxos Exceção:

[FE11] – Campo obrigatório não preenchido.

01 – O sistema exibe a MSG019 no campo não preenchido.

02 – O usuário preenche o campo indicado.

03 – Retorna ao [FP12] passo 04.

Mensagens do Sistema:

MSG019: “Campo obrigatório.”

A.10. UC013 – Cadastrar tratamento da consulta

Descrição: Processo de inserção do tratamento do paciente na consulta.

Atores: Usuário (Médico).

Pré-condições: Usuário autenticado e consulta cadastrada.

Pós-condições: Um novo tratamento é cadastrado.

Fluxo Principal:

[FP13] – Cadastro do tratamento da consulta.

01 – O sistema exibe a aba de tratamento na consulta do paciente.

02 – O usuário escreve o tratamento no campo de texto disponível.

03 – O usuário clica no botão salvar [FA12].

04 – O sistema salva o tratamento no banco de dados.

05 – O sistema exibe a mensagem MSG020.

Fluxos Alternativos:

[FA12] - Cancelamento do cadastro.

01 - O usuário clica no botão cancelar.

02 - O sistema redireciona o usuário para a tela de listagem de consultas.

Mensagens do Sistema:

MSG020: “Operação realizada com sucesso.”

A.11. UC014 – Agendar consulta

Descrição: Processo de agendamento de uma nova consulta

Atores: Usuário (Administrativo, Médico)

Pré-condições: Usuário autenticado, médico cadastrado, paciente cadastrado.

Pós-condições: Um novo agendamento é cadastrado.

Fluxo Principal:

[FP14] – Cadastro do agendamento da consulta

01 – O sistema exibe o formulário de agendamento.

02 – O usuário seleciona o paciente.

03 – O usuário seleciona o médico.

04 – O usuário informa a data da consulta.

05 – O usuário seleciona o tipo de consulta.

06 – O usuário clica no botão salvar [FA13].

07 – O sistema salva o agendamento no banco de dados.

08 – O sistema exibe a mensagem MSG021.

Fluxos Alternativos:

[FA13] – Cancelamento do agendamento

01 – O usuário clica no botão cancelar.

02 – O sistema redireciona o usuário para a tela de listagem de agendamentos.

Mensagens do Sistema:

MSG021: “Consulta agendada com sucesso.”

APENDICE B - DESCRIÇÃO DAS TABELAS DO BANCO DE DADOS

A Tabela 3 apresenta a estrutura da entidade Agenda do banco de dados.

Tabela 3 - Tabela Agenda.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
data_agendamento	TIMESTAMP	Data/hora agendamento
descricao	VARCHAR(255)	Descrição da agenda

Fonte: O autor (2025).

A Tabela 4 apresenta a estrutura da entidade AgendaConsulta do banco de dados.

Tabela 4 - Tabela AgendaConsulta.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
id_medico	INT8	Chave estrangeira da tabela Medico
id_paciente	INT8	Chave estrangeira da tabela Paciente
id_agenda	INT8	Chave estrangeira da tabela Agenda
tipo_consulta	VARCHAR(100)	Tipo da consulta agendada

Fonte: O autor (2025).

A Tabela 5 apresenta a estrutura da entidade AlergiaPaciente do banco de dados.

Tabela 5 - Tabela AlergiaPaciente.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
id_paciente	INT8	Chave estrangeira da tabela Paciente
descricao	VARCHAR(255)	Descrição da agenda

Fonte: O autor (2025).

A Tabela 6 apresenta a estrutura da entidade Arquivo do banco de dados.

Tabela 6 - Tabela Arquivo.

Coluna	Tipo de dados	Descrição

id	INT8	Chave primária da tabela
nome	VARCHAR(255)	Nome do arquivo
arquivo	BYTEA	Conteúdo do arquivo
descricao	VARCHAR(100)	Descrição do arquivo
tipo	VARCHAR(100)	Tipo do arquivo
id_consulta	INT8	Chave estrangeira tabela Consulta

Fonte: O autor (2025).

A Tabela 7 apresenta a estrutura da entidade CID do banco de dados.

Tabela 7 - Tabela CID.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
codigo	VARCHAR(45)	Código do Cid
descricao	VARCHAR(255)	Descrição do Cid
id_diagnostico	INT8	Chave estrangeira da tabela Diagnostico

Fonte: O autor (2025).

A Tabela 8 apresenta a estrutura da entidade Consulta do banco de dados.

Tabela 8- Tabela Consulta.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
anamnese	VARCHAR(4000)	Relato do paciente
data_consulta	TIMESTAMP	Data/hora da consulta
tipo	VARCHAR(50)	Tipo da consulta
id_medico	INT8	Chave estrangeira da tabela Medico
id_paciente	INT8	Chave estrangeira da tabela Paciente
id_consulta	INT8	Chave estrangeira da tabela Consulta
id_agenda_consulta	INT8	Chave estrangeira da tabela AgendaConulta

Fonte: O autor (2025).

A Tabela 9 apresenta a estrutura da entidade Contato do banco de dados.

Tabela 9 - Tabela Contato.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
celular	VARCHAR(20)	Celular da entidade contato
telefone	VARCHAR(20)	Telefone da entidade contato
email	VARCHAR(100)	E-mail da entidade contato
tipo_contato	VARCHAR(20)	Tipo de contato
id_pessoa	INT8	Chave estrangeira da tabela Pessoa

Fonte: O autor (2025).

A Tabela 10 apresenta a estrutura da entidade Diagnostico do banco de dados.

Tabela 10 - Tabela Diagnostico.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
diagnostico	VARCHAR(4000)	Diagnóstico do paciente
id_consulta	INT8	Chave estrangeira da tabela Consulta

Fonte: O autor (2025).

A Tabela 11 apresenta a estrutura da entidade Exame do banco de dados.

Tabela 11 - Tabela Exame.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
descricao	VARCHAR(255)	Descrição do exame
id_consulta	INT8	Chave estrangeira da tabela Consulta

Fonte: O autor (2025).

A Tabela 12 apresenta a estrutura da entidade Endereco do banco de dados.

Tabela 12 - Tabela Endereco.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
logradouro	VARCHAR(255)	Logradouro da entidade Endereco
numero	VARCHAR(20)	Número da entidade Endereco
complemento	VARCHAR(255)	Complemento da entidade Endereco
cep	VARCHAR(20)	CEP da entidade Endereco

bairro	VARCHAR(100)	Bairro da entidade Endereco
uf	VARCHAR(2)	UF da entidade Endereco
pais	VARCHAR(100)	País da entidade Endereco
id_pessoa	INT8	Chave estrangeira da tabela Pessoa

Fonte: O autor (2025).

A Tabela 13 apresenta a estrutura da entidade Medico do banco de dados.

Tabela 13 - Tabela Medico.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
crm	VARCHAR(20)	Código CRM do médico
especialidade	VARCHAR(255)	Especialidade do médico
id_pessoa	INT8	Chave estrangeira da tabela Pessoa

Fonte: O autor (2025).

A Tabela 14 apresenta a estrutura da entidade Paciente do banco de dados.

Tabela 14 – Tabela Paciente.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
peso	VARCHAR(6)	Peso do paciente
altura	VARCHAR(6)	Altura do paciente
id_pessoa	INT8	Chave estrangeira da tabela Pessoa

Fonte: O autor (2025).

A Tabela 15 apresenta a estrutura da entidade Prescricao do banco de dados.

Tabela 15 – Tabela Prescricao.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
descricao	VARCHAR(4000)	Descrição da prescrição
id_consulta	INT8	Chave estrangeira da tabela Consulta

Fonte: O autor (2025).

A Tabela 16 apresenta a estrutura da PrescricaoConsultaMedicamento no banco de dados.

Tabela 16 - Tabela PrescricaoConsultaMedicamento.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
instrucao_uso	VARCHAR(4000)	Instruções uso do Medicamento
medicamento	VARCHAR(255)	Descrição do medicamento
id_prescricao	INT8	Chave estrangeira da tabela Prescricao
observacao	VARCHAR(255)	Observações adicionais medicamento

Fonte: O autor (2025).

A Tabela 17 apresenta a estrutura da entidade Tratamento do banco de dados.

Tabela 17 – Tabela Tratamento

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
tratamento	VARCHAR(4000)	Detalhes do tratamento do paciente
id_consulta	INT8	Chave estrangeira da tabela Consulta

Fonte: O autor (2025).

A Tabela 18 apresenta a estrutura da entidade Usuario do banco de dados.

Tabela 18 - Tabela Usuário.

Coluna	Tipo de dados	Descrição
id	INT8	Chave primária da tabela
login	VARCHAR(100)	Login de acesso do usuário
senha	VARCHAR(255)	Senha criptografada do usuário
role	VARCHAR(20)	Perfil/Papel do usuário no sistema
status	VARCHAR(20)	Status da conta do usuário
id_pessoa	INT8	Chave estrangeira da tabela Pessoa

Fonte: O autor (2025).