

UNIVERSIDADE SÃO JUDAS TADEU
CURSO DE SISTEMAS DE INFORMAÇÃO

ALEX ALDIB – RA 823165544
CAUE HERRAIZ – RA 82211809
ENZO MONTEIRO – RA 822125009
GUILHERME FIGUEIREDO – RA 823116166
LEONARDO RIBEIRO – RA 823213510
PEDRO VIEIRA GONÇALVES – RA 822140239

DOCUMENTAÇÃO DE DESENVOLVIMENTO DE SOFTWARE
SISTEMA DE AGENDAMENTO PARA CLÍNICAS

SÃO PAULO
2025

UNIVERSIDADE SÃO JUDAS TADEU
CURSO DE SISTEMAS DE INFORMAÇÃO

ALEX ALDIB – RA 823165544
CAUE HERRAIZ – RA 82211809
ENZO MONTEIRO – RA 822125009
GUILHERME FIGUEIREDO – RA 823116166
LEONARDO RIBEIRO – RA 823213510
PEDRO VIEIRA GONÇALVES – RA 822140239

DOCUMENTAÇÃO DE DESENVOLVIMENTO DE SOFTWARE
SISTEMA DE AGENDAMENTO PARA CLÍNICAS

Trabalho apresentado à disciplina de Gestão e Qualidade de Software do curso de Sistemas de Informação da Universidade São Judas Tadeu, como parte dos requisitos para avaliação do Projeto A3.

Professor: Robson Calvetti

SÃO PAULO

2025

SUMÁRIO

1. Introdução.....	4
2. Planejamento de Testes de Software.....	5
2.1. Cronograma de atividades.....	5
2.2. Alocação de recursos.....	5
2.3. Marcos do projeto.....	5
3. Documentos de Desenvolvimento de Software.....	6
3.1. Plano de Projeto.....	6
3.1.1. Planejamento do projeto.....	6
3.1.2. Escopo.....	6
3.1.3. Recursos.....	7
3.1.4. Estimativas de projeto.....	8
3.2. Documento de Requisitos.....	9
3.3. Planejamento de Testes.....	10
3.3.1. Plano de Testes.....	10
3.3.1.1. Introdução.....	10
3.3.1.2. Escopo.....	10
3.3.1.3. Objetivos.....	10
3.3.1.4. Requisitos a serem testados.....	10
3.3.1.5. Estratégias, tipos de testes e ferramentas.....	11
3.3.1.6. Recursos a serem empregados.....	12
3.3.1.7. Cronograma das atividades.....	13
3.3.1.8. Definição dos marcos do projeto.....	13
3.3.2. Casos de Testes.....	13
3.3.3. Roteiro de Testes.....	13
4. Gestão de Configuração de Software.....	14
5. Repositório de Gestão de Configuração de Software.....	14
6. Conclusão.....	15
7. Referências Bibliográficas.....	16

1. Introdução

Este documento apresenta a documentação completa do desenvolvimento de um sistema de agendamento para clínicas e consultórios, elaborado como parte do Projeto A3 da disciplina de Gestão e Qualidade de Software. O sistema proposto tem como objetivo otimizar o fluxo de trabalho em ambientes de saúde, permitindo o cadastro de pacientes, o gerenciamento de horários dos médicos, o agendamento e cancelamento de consultas, além do acompanhamento de históricos clínicos.

Com base em práticas consolidadas da engenharia de software, o projeto contempla todas as etapas necessárias para a construção de um sistema confiável e de alta qualidade. A equipe foi responsável por realizar o planejamento detalhado do projeto, o levantamento dos requisitos funcionais e não funcionais, a definição dos casos de uso, a elaboração do plano de testes e a gestão da configuração do software.

Foram consideradas diferentes estratégias de teste, como testes funcionais, não funcionais, de regressão, de integração e de aceitação, aplicadas sobre os principais fluxos do sistema. Além disso, foram definidas ferramentas adequadas para cada tipo de validação, incluindo Selenium, Postman, TestLink, entre outras, garantindo cobertura completa das funcionalidades previstas.

A documentação apresentada reflete um esforço coletivo na organização, planejamento e simulação de um projeto real, com foco em qualidade, segurança, usabilidade e eficiência. O sistema desenvolvido atende a necessidades práticas e oferece um ambiente sólido para aplicação de testes e práticas de controle de qualidade de software.

2. Planejamento de Testes de Software

2.1 Cronograma de Atividades

O cronograma a seguir apresenta a sequência das atividades, desde a definição dos testes até a validação final do sistema.

Atividade	Responsável	Início	Fim	Duração
Definição dos tipos de testes	Pessoa 1	10/06/2025	11/06/2025	2 dias
Levantamento dos requisitos com base no sistema de agendamento	Pessoa 3	10/06/2025	12/06/2025	3 dias
Elaboração do plano de testes (parte 1)	Pessoa 4	12/06/2025	16/06/2025	5 dias
Criação dos casos de teste	Pessoa 5	13/06/2025	17/06/2025	5 dias
Execução dos testes	Equipe	18/06/2025	21/06/2025	4 dias
Correção de erros encontrados	Desenvolvedores	22/06/2025	24/06/2025	3 dias
Reexecução e validação final	Equipe	25/06/2025	26/06/2025	2 dias

2.2 Alocação de Recursos

Recursos Humanos e Recursos Técnicos

A equipe contou com a Pessoa 1 no planejamento e coordenação dos testes, Pessoa 3 no levantamento de requisitos, Pessoas 4 e 5 na elaboração e execução dos testes, e os desenvolvedores foram responsáveis pelas correções e ajustes após os testes.

O ambiente de testes foi estável e funcional, utilizando dados fictícios no banco de dados. Todas as funcionalidades do sistema, como agendamento, cancelamento, login e cadastro, estiveram disponíveis para validação.

Ferramentas previstas:

Utilizou-se o **TestLink** ou **Xray** para gerenciar os testes, o **Selenium** para automatizar testes da interface, o **Postman** para testar APIs e o **Trello** ou **Jira** para organizar as tarefas da equipe.

2.3 Marcos do Projeto

Os marcos estabelecidos neste projeto representam os principais pontos de acompanhamento e controle da evolução do processo de testes. A fase de planejamento é concluída em 11 de junho de 2025, seguida pela entrega do plano de testes completo no dia 16 de junho. A execução dos testes tem início em 18 de junho, permitindo a identificação e correção de erros até 24 de junho.

Por fim, a validação final do sistema e o encerramento formal da fase de testes estão previstos para o dia 26 de junho de 2025.

3. Documentos de Desenvolvimento de Software

3.1 Plano de Projeto

3.1.1 Planejamento do Projeto

Objetivo

O objetivo deste projeto é desenvolver um sistema de agendamento voltado para clínicas e consultórios, com foco na otimização do fluxo de trabalho entre pacientes, médicos e recepcionistas. A solução incluirá funcionalidades essenciais como cadastro de pacientes, gerenciamento de horários, agendamento e cancelamento de consultas, histórico de atendimentos e controle de acesso conforme o perfil do usuário.

Justificativa

O sistema foi projetado para responder a demandas recorrentes enfrentadas por clínicas, como a desorganização no controle de consultas, sobreposição de horários, dificuldade no acesso a históricos médicos e o retrabalho da equipe de recepção. A proposta visa aumentar a eficiência operacional, melhorar a qualidade do atendimento e garantir maior confiabilidade na gestão das informações.

Metodologia

O projeto será desenvolvido com base nas boas práticas da Engenharia de Software, seguindo uma abordagem incremental. As etapas incluem o levantamento e documentação dos requisitos, modelagem do sistema, planejamento e execução dos testes, controle de qualidade com gestão de configuração, além da elaboração dos relatórios e da entrega final.

3.1.2 Escopo do Projeto

Funcionalidades Incluídas

O sistema contará com funcionalidades essenciais para a rotina clínica, como o cadastro de pacientes com informações pessoais e histórico médico, e o cadastro de médicos com dados profissionais e horários disponíveis. Também permitirá o agendamento, cancelamento e remarcação de consultas, seguindo regras de disponibilidade. Além disso, oferecerá acesso ao histórico de atendimentos e login com perfis diferenciados para pacientes, recepcionistas e médicos.

Funcionalidades Fora do Escopo

Integrações com convênios; Emissão de notas fiscais e Prontuário eletrônico detalhado.

3.1.3 Recursos do Projeto

Recursos Humanos

A equipe do projeto é composta por profissionais com funções bem definidas. Dois desenvolvedores são responsáveis por implementar o sistema, realizar correções e propor melhorias. Um analista de requisitos atua no levantamento e documentação das necessidades do projeto, garantindo que as funcionalidades estejam alinhadas aos objetivos. Dois testadores (QA) elaboram e executam os testes, assegurando a qualidade do produto. Por fim, o gerente de projeto coordena a equipe, planeja as atividades e monitora o progresso para garantir o cumprimento dos prazos e metas estabelecidas.

Recursos Técnicos

A infraestrutura do projeto conta com computadores devidamente configurados com ambientes de desenvolvimento e testes. Um banco de dados com dados fictícios é utilizado para simulações. São empregadas ferramentas como Postman para testes de API, Selenium para testes automatizados e TestLink ou Xray para o gerenciamento dos testes. A gestão das tarefas e o acompanhamento do projeto são realizados por meio do Jira ou Trello. Para o desenvolvimento, utiliza-se o Visual Studio Code ou uma IDE equivalente, além de um servidor local para a realização dos testes.

Recursos Materiais

Ambiente físico ou virtual para reuniões de equipe, conexão com internet estável e Armazenamento em nuvem (Google Drive, GitHub).

3.1.4 Estimativas do Projeto

Cronograma Geral

Atividade	Início	Fim	Duração
Levantamento de Requisitos	10/06/2025	12/06/2025	3 dias
Planejamento dos Testes	10/06/2025	11/06/2025	2 dias
Desenvolvimento Inicial	10/06/2025	16/06/2025	7 dias
Elaboração do Plano de Testes	12/06/2025	16/06/2025	5 dias
Criação de Casos de Teste	13/06/2025	17/06/2025	5 dias

Atividade	Início	Fim	Duração
Execução dos Testes	18/06/2025	21/06/2025	4 dias
Correções de Bugs	22/06/2025	24/06/2025	3 dias
Validação Final	25/06/2025	26/06/2025	2 dias
Preparação da Apresentação	09/06/2025	11/06/2025	3 dias

Marcos do Projeto

O cronograma do projeto está organizado de forma a garantir o andamento eficiente das atividades. Em 11 de junho de 2025 será entregue o planejamento de testes, seguido pela finalização do documento de requisitos no dia 12. O plano de testes completo estará pronto até 16 de junho, permitindo o início da execução dos testes em 18 de junho. As correções serão concluídas até o dia 24, com a validação final do sistema marcada para 26 de junho. Por fim, a apresentação final ocorrerá em 27 de junho de 2025, encerrando o ciclo do projeto.

Estimativa de Esforço

Total de integrantes: 6. Estimativa de horas por integrante: 30 horas.

Esforço total estimado: 180 horas

3.2 Documentos de Requisitos

Requisitos Funcionais

Código Requisito

- RF001 O sistema deve permitir o cadastro de pacientes com dados como nome, CPF, telefone, endereço e, opcionalmente, plano de saúde.
- RF002 O sistema deve possibilitar o agendamento de consultas com validação de horários disponíveis e prevenção de sobreposições.
- RF003 Médicos e administradores devem poder registrar e editar seus dias e horários de atendimento.
- RF004 O sistema deve armazenar o histórico de atendimentos de cada paciente, acessível a médicos e recepcionistas.

Código Requisito

- RF005 Consultas devem poder ser canceladas ou reagendadas por pacientes ou recepcionistas de forma intuitiva.
- RF006 O sistema deve permitir login com perfis diferenciados (médico, paciente, recepcionista), oferecendo acesso conforme a função.
- RF007 O sistema deve permitir que médicos e recepcionistas visualizem a agenda diária com horários disponíveis e consultas marcadas.
- RF008 O sistema deve enviar notificações de confirmação e lembrete de consultas por e-mail ou SMS.

Requisitos Não Funcionais

Código Requisito

- RNF001 Os dados sensíveis (como CPF e informações médicas) devem ser protegidos por criptografia.
- RNF002 O sistema deve ser compatível com os principais navegadores (Chrome, Firefox, Edge).
- RNF003 A resposta para ações como login, agendamento e cadastro devem ocorrer em até 2 segundos.
- RNF004 O sistema deve realizar backups automáticos do banco de dados diariamente.
- RNF005 A interface do sistema deve ser responsiva e adaptável a computadores, tablets e smartphones.

3.3 Planejamento de testes

3.3.1 Plano de teste

3.3.1.1 Introdução

O sistema de agendamento para clínicas e consultórios desenvolvido neste projeto exige um plano de testes robusto, que aborde diferentes tipos de validações, usuários e cenários de uso. Considerando os fluxos principais, a variedade de perfis (médico, paciente, recepcionista) e as regras de negócio envolvidas, foram definidas estratégias e ferramentas específicas para garantir a qualidade e confiabilidade do sistema.

3.3.1.2 Escopo

O sistema desenvolvido abrangerá funcionalidades essenciais para o gerenciamento de atendimentos em clínicas e consultórios. Entre os principais recursos, estão o cadastro de pacientes com dados pessoais, contato, CPF e histórico clínico, além do cadastro de médicos com suas especialidades, horários disponíveis e número de registro (CRM).

Também será possível realizar agendamentos de consultas por meio de uma interface intuitiva, acessível a recepcionistas, médicos e pacientes, com suporte ao cancelamento e remarcação seguindo regras de disponibilidade.

O sistema permitirá ainda o acompanhamento do histórico de consultas realizadas, com acesso restrito conforme o perfil do usuário. O login será segmentado por perfis (paciente, recepcionista e médico), garantindo a segurança e o acesso personalizado às funcionalidades correspondentes a cada tipo de usuário.

3.3.1.3 Objetivos dos Testes

Os testes definidos no projeto têm como objetivo principal garantir a qualidade e a confiabilidade do sistema. Eles visam assegurar o funcionamento correto do agendamento, prevenindo conflitos de horário, além de validar a integridade dos dados entre pacientes, médicos e consultas. Também são testadas as funcionalidades conforme os diferentes perfis de usuários, o cumprimento das regras de negócio, como horários e validações, e a consistência do histórico de consultas ao longo do tempo de uso.

3.3.1.4 Requisitos a Serem Testados

Os requisitos a serem validados nos testes abrangem funcionalidades essenciais do sistema. No cadastro de pacientes, serão verificados campos obrigatórios, unicidade de CPF e a geração do histórico inicial. O agendamento de consultas será testado quanto à disponibilidade de horários, regras de antecedência e duração das consultas. O gerenciamento de médicos envolve o cadastro de especialidades, horários de atendimento e regras específicas para folgas e feriados. O histórico de consultas deve ser gerado automaticamente após cada atendimento, mantendo vínculo com paciente e médico. Por fim, o controle de acesso por perfil será validado conforme permissões e funcionalidades específicas de cada tipo de usuário.

3.3.1.5 Estratégias de Teste

A estratégia de testes abrange aspectos funcionais, não funcionais e regressivos. Os testes funcionais garantem o correto funcionamento das operações CRUD em todas as entidades, além da validação das regras de negócio e dos fluxos operacionais. Os testes não funcionais avaliam o desempenho sob carga, simulando agendamentos simultâneos, verificam a segurança contra acessos não autorizados e analisam a usabilidade, considerando clareza e acessibilidade dos fluxos. Já os testes regressivos asseguram que novas implementações não comprometam funcionalidades já existentes, com foco especial em áreas críticas como o agendamento e o login.

Tipos de Testes a Serem Aplicados

Os testes do sistema serão divididos em três categorias principais. Os testes funcionais visam garantir que todas as operações CRUD funcionem corretamente em cada entidade e que os fluxos e regras de negócio sejam executados conforme o esperado. Já os testes não funcionais abordarão aspectos como desempenho sob carga, segurança contra acessos indevidos e usabilidade, avaliando a clareza e acessibilidade do sistema. Por fim, os testes regressivos serão realizados para assegurar que novas funcionalidades não impactem negativamente as já existentes, com atenção especial a processos essenciais como agendamento e login.

Ferramentas Recomendadas

Para garantir a qualidade do sistema em desenvolvimento, a equipe optou por utilizar um conjunto abrangente de ferramentas de testes, selecionadas conforme as necessidades específicas de cada etapa do projeto. Nos testes unitários e de integração, serão empregadas ferramentas como JUnit, pytest, Mocha/Chai e Mockito, com foco na verificação individual dos componentes e suas interações. Para os testes relacionados ao banco de dados, foram escolhidos o DBUnit, que automatiza os testes, e o Liquibase, responsável pelo controle de versões do esquema. A validação da interface será feita com o Selenium WebDriver e o Cypress, que possibilitam a automação eficaz dos testes de front-end.

Em relação ao desempenho do sistema, o Apache JMeter e o Gatling serão utilizados para simular carga e avaliar a performance. Já nos testes de APIs, o Postman e o RestAssured garantirão a confiabilidade dos serviços. Além disso, será adotado o Cucumber, que segue a abordagem BDD (Behavior Driven Development), facilitando a colaboração entre equipe técnica e partes interessadas. Para o controle de qualidade do código, o SonarQube fará a análise estática e o JaCoCo medirá a cobertura dos testes. O uso integrado dessas ferramentas visa assegurar um sistema confiável, eficiente e de fácil manutenção.

3.3.1.6 Recursos a serem empregados

Recursos Humanos

A equipe será composta por diferentes perfis profissionais, cada um com funções específicas no processo de testes. O Analista de Testes (QA) ficará responsável pela elaboração dos casos de teste, execução das validações, documentação dos resultados e verificação dos critérios de aceitação. Os desenvolvedores atuarão na correção de falhas encontradas, além de oferecer suporte técnico sobre o funcionamento das funcionalidades. O Product Owner (PO) servirá como elo entre a equipe técnica e os usuários, sendo o responsável pelo esclarecimento de requisitos e pela validação dos testes de aceitação. Já o

usuário final, representado por um profissional da clínica, participará dos testes de aceitação, garantindo que o sistema atenda aos fluxos reais de uso.

Recursos Tecnológicos

Ambientes de Teste

Serão utilizados dois ambientes distintos: o ambiente de desenvolvimento, voltado para a execução de testes unitários e rápidos durante a codificação, e o ambiente de homologação, que simula o ambiente de produção e será utilizado para testes de integração, sistema e aceitação.

Ferramentas de Teste

Diversas ferramentas serão utilizadas conforme o tipo de teste. Para testes unitários e de integração, serão adotados JUnit, pytest, Mocha/Chai e Mockito. Os testes de interface contarão com o uso do Selenium WebDriver e do Cypress. JMeter e Gatling serão empregados para testar carga e performance. Postman e RestAssured serão usados nos testes de API. Já o Cucumber será responsável por testes baseados em BDD, e o JaCoCo pela análise da cobertura dos testes.

Infraestrutura

A infraestrutura de testes será composta por servidores locais ou em nuvem para hospedagem dos ambientes. Além disso, ferramentas como Jenkins e GitLab CI serão utilizadas para automação dos testes e deploy contínuo em ambientes específicos.

Recursos de Informação

Os recursos informacionais essenciais incluem a documentação de requisitos funcionais e não funcionais, que será a base para a criação dos testes. Também serão utilizadas as especificações técnicas, como diagramas de entidades, modelos de dados e documentação das APIs. Os critérios de aceitação, definidos em conjunto com o PO, servirão como referência para os testes de validação. O histórico de erros e melhorias registrados será um importante guia para a realização de testes regressivos.

3.3.1.7. Cronograma das Atividades

O cronograma de testes está dividido em sete etapas principais. A primeira etapa consiste na análise dos requisitos e definição dos testes, a ser realizada pelo Analista de Testes em dois dias. Em seguida, a equipe de DevOps ou TI realizará, em dois dias, a preparação do ambiente e a configuração das ferramentas. A terceira etapa é dedicada à criação dos casos de teste, com base nos requisitos e regras de negócio, e será executada pelo Analista de Testes ao longo de quatro dias. A quarta etapa compreende a execução dos testes unitários, de integração, sistema, regressão e performance, conduzida pela equipe de QA durante seis dias. Após isso, os testes de aceitação serão realizados com a participação do PO e do usuário final, em dois dias. A etapa

seguinte envolve a correção de falhas e reexecução dos testes corrigidos, com duração de três dias, sob responsabilidade da equipe de desenvolvimento e QA. Por fim, será gerado um relatório final com os resultados e avaliação do processo de testes, em um dia, também a cargo do Analista de Testes.

3.3.1.8. Definição dos Marcos do Projeto

Pós a análise de requisitos, prepara-se o ambiente e as ferramentas para iniciar os testes. Durante a fase de testes, os bugs críticos são identificados, corrigidos e validados. Com os testes concluídos, é lançada a versão beta para avaliação dos usuários. Após aprovação final, o sistema é liberado para produção e lançado oficialmente.

3.3.2. Casos de Testes

Os casos de teste descrevem cenários específicos para garantir o funcionamento correto das funcionalidades do sistema. Um exemplo é o caso TC-AG-001, que verifica se o sistema permite o agendamento de consultas em horários disponíveis. Ele exige que paciente e médico estejam cadastrados e que haja um horário livre. O procedimento inclui login como recepcionista, preenchimento dos dados e confirmação. O resultado esperado é a confirmação do agendamento, o registro correto e o bloqueio do horário.

Outro exemplo é o caso TC-PA-001, que avalia o cadastro de um paciente com dados válidos. A pré-condição é que o CPF não esteja duplicado e que o usuário tenha as permissões necessárias. O processo envolve acessar o módulo de pacientes, preencher os dados e concluir o cadastro. O resultado esperado é a confirmação do cadastro com o paciente ativo no sistema.

3.3.3. Roteiro de Testes

O roteiro de testes será elaborado para cada caso de teste e incluirá uma introdução com os objetivos do teste, a identificação do projeto (SAGC), o escopo e o tipo de teste (como funcional, de sistema, integração ou banco de dados). Serão definidas as pré-condições necessárias, os procedimentos a serem seguidos, os resultados esperados e as pós-condições. Esse roteiro garante padronização e rastreabilidade dos testes realizados.

4. Gestão de Configuração de Software

Objetivo

A gestão de configuração visa estabelecer os processos, ferramentas e responsabilidades necessários para o controle de versões, rastreabilidade e integridade dos artefatos relacionados ao sistema SAGC.

Itens de Configuração (SCI)

Serão controlados itens como código-fonte, scripts de banco de dados, documentação, casos e planos de teste, relatórios, releases e artefatos de produção.

Ferramentas e Ambientes

Serão utilizadas ferramentas como Git, GitHub, GitHub Projects e GitHub Actions para versionamento e automação. Para testes e documentação, serão usados Postman, JMeter, VSCode e Markdown. Os ambientes de CI/CD, locais e em VPS, serão usados para integração e entrega contínua. A produção será hospedada em soluções de cloud como AWS, Heroku ou Render.

Segurança e Treinamentos

Será adotada autenticação de dois fatores (2FA), uso de variáveis de ambiente seguras (.env) e treinamentos regulares sobre Git, CI/CD, testes e padronização.

Processos SCM

Os processos de controle de configuração incluirão o uso de branches para controle de versão, revisão de código por meio de pull requests, versionamento semântico, auditorias quinzenais e definição de baselines formais para cada fase do projeto.

Comitê de Controle de Mudanças (CCB)

Um comitê composto por gerente de projeto, gerente de configuração, líder técnico, analista de qualidade e PO será responsável por avaliar, aprovar e registrar mudanças críticas no sistema.

Funções e Relatórios

As funções envolvidas incluem gerente de configuração, desenvolvedores, revisores de código e analistas de qualidade (QA). Relatórios de status de configuração serão emitidos mensalmente para acompanhar a evolução do projeto.

5. Repositório de Gestão de Configuração de Software

O gerenciamento de configuração no projeto garante controle de versões, rastreabilidade e integridade dos artefatos. As alterações são organizadas por meio de ramificações (branches), seguindo convenções padronizadas e aprovadas por revisão de código. Utiliza-se o versionamento semântico (MAJOR.MINOR.PATCH), com auditorias regulares e definição de baselines em marcos importantes do projeto.

Um Comitê de Controle de Mudanças (CCB) analisa e aprova modificações críticas, assegurando qualidade e coerência. Relatórios de status são emitidos periodicamente, e os entregáveis incluem versões estáveis, snapshots para testes e documentação devidamente versionada..

6. Conclusão

O desenvolvimento do projeto “Sistema de Agendamento para Clínicas e Consultórios” permitiu aplicar, de forma prática e estruturada, os principais conceitos estudados na disciplina de Gestão e Qualidade de Software. Ao longo

do trabalho, foram elaboradas todas as etapas fundamentais de um processo de desenvolvimento bem planejado, com foco na organização, rastreabilidade e controle da qualidade.

A partir da definição clara dos requisitos funcionais e não funcionais, foi possível construir casos de uso relevantes que representam os principais fluxos do sistema, atendendo às necessidades específicas dos perfis envolvidos: pacientes, médicos e recepcionistas. O plano de projeto e os recursos definidos demonstram a viabilidade técnica e operacional da solução.

O planejamento dos testes, por sua vez, foi elaborado com base em estratégias sólidas, contemplando diferentes tipos de testes, ambientes simulados e ferramentas apropriadas. A definição dos casos, o cronograma e os critérios de aceitação garantem uma abordagem completa para validação da qualidade do software.

Além disso, a documentação organizada, o cuidado com a gestão de configuração e o alinhamento entre todos os tópicos reforçam a importância da engenharia de software como disciplina essencial para garantir entregas confiáveis, seguras e eficientes.

Por fim, o trabalho em equipe foi essencial para o sucesso do projeto, permitindo a divisão equilibrada das tarefas e a construção colaborativa de um sistema realista, bem documentado e com alto potencial de aplicação no mercado da saúde.

7. Referências bibliográficas

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR ISO/IEC 9126-1: Engenharia de software – Qualidade de produto*. Rio de Janeiro: ABNT, 2003.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 6023: Informação e documentação – Referências – Elaboração*. Rio de Janeiro: ABNT, 2018.

POSTMAN. *API Platform for Building and Using APIs*. Disponível em: <https://www.postman.com>. Acesso em: 09 jun. 2025.

SELENIUM. *Selenium WebDriver*. Disponível em: <https://www.selenium.dev>. Acesso em: 09 jun. 2025.

APACHE JMETER. *Performance Testing Tool*. Disponível em: <https://jmeter.apache.org>. Acesso em: 09 jun. 2025.

GITHUB DOCS. *Using GitHub for Version Control*. Disponível em: <https://docs.github.com>. Acesso em: 09 jun. 2025.

CYPRESS. *Fast, Easy and Reliable Testing for Anything that Runs in a Browser*. Disponível em: <https://www.cypress.io>. Acesso em: 09 jun. 2025.

X-RAY. *Test Management for Jira*. Disponível em: <https://www.getxray.app>. Acesso em: 09 jun. 2025.

TRELLO. *Organize Anything, Together*. Disponível em: <https://trello.com>. Acesso em: 09 jun. 2025.

SONARQUBE. *Continuous Inspection of Code Quality*. Disponível em: <https://www.sonarqube.org>. Acesso em: 09 jun. 2025.