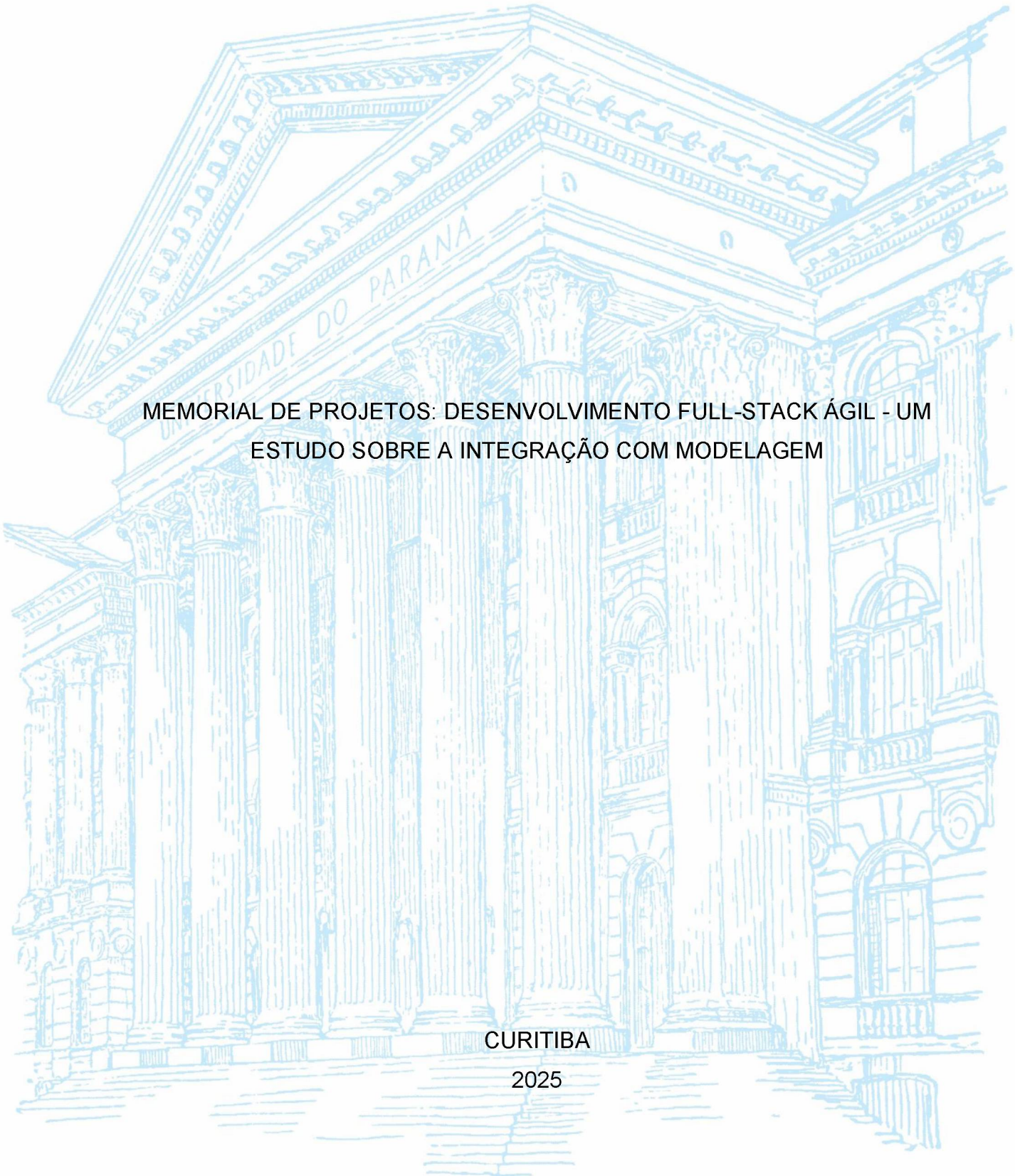
UNIVERSIDADE FEDERAL DO PARANÁ

DANILO MACHADO DE OLIVEIRA



MEMORIAL DE PROJETOS: DESENVOLVIMENTO FULL-STACK ÁGIL - UM ESTUDO SOBRE A INTEGRAÇÃO COM MODELAGEM

CURITIBA

2025

DANILO MACHADO DE OLIVEIRA

MEMORIAL DE PROJETOS: DESENVOLVIMENTO FULL-STACK ÁGIL - UM ESTUDO SOBRE A INTEGRAÇÃO COM MODELAGEM

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Desenvolvimento Ágil de Software, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Desenvolvimento Ágil de Software.

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2025

RESUMO

O presente memorial tem como objetivo relatar a trajetória acadêmica e prática do aluno ao longo do curso de Especialização em Desenvolvimento Ágil de Software, consolidando os conhecimentos adquiridos por meio de artefatos desenvolvidos nas disciplinas. Cada módulo contribuiu para a construção de um portfólio técnico que representa o ciclo completo de desenvolvimento full-stack. Os trabalhos incluem modelagem ágil com histórias de usuário, diagramas UML, wireframes, planos de release com métodos Scrum e Kanban, implementação de sistemas web e mobile utilizando Angular, Spring Boot e Kotlin, além de testes automatizados e práticas DevOps com Docker e GitLab. A integração entre teoria e prática foi evidenciada na articulação entre modelagem, codificação e entrega contínua, destacando a aplicação dos princípios ágeis desde a concepção até a entrega de software funcional. O memorial evidencia como os conteúdos das disciplinas se complementaram na formação de um profissional multidisciplinar, apto a atuar em todas as camadas do desenvolvimento de software com foco em agilidade, qualidade e usabilidade.

Palavras-chave: ágil, desenvolvimento, integração, modelagem.

ABSTRACT

This report aims to chronicle the student's academic and practical journey throughout the Agile Software Development Specialization course, consolidating the knowledge acquired through artifacts developed in the courses. Each module contributed to the construction of a technical portfolio that represents the complete full-stack development cycle. The work includes agile modeling with user stories, UML diagrams, wireframes, release plans using Scrum and Kanban methods, implementation of web and mobile systems using Angular, Spring Boot, and Kotlin, as well as automated testing and DevOps practices with Docker and GitLab. The integration of theory and practice was demonstrated in the articulation of modeling, coding, and continuous delivery, highlighting the application of agile principles from conception to delivery of functional software. The report highlights how the course contents complemented each other in the development of a multidisciplinary professional, capable of working in all layers of software development with a focus on agility, quality, and usability.

Keywords: agile, development, integration, modeling.

SUMÁRIO

1 PARECER TÉCNICO

Este parecer técnico apresenta uma síntese dos projetos desenvolvidos ao longo do curso de Especialização em Desenvolvimento Ágil de Software da Universidade Federal do Paraná, evidenciando como os conteúdos das disciplinas se integraram de forma coerente e progressiva na construção de uma base sólida para o desenvolvimento de software (Pressman; Maxim, 2016). O memorial está estruturado de modo a refletir a evolução prática do estudante, articulando teoria, ferramentas e metodologias aplicadas.

A jornada se inicia com o módulo de Métodos Ágeis (MADS), no qual foi construído um mapa mental abrangente que explorou desde os modelos tradicionais de processos de software até os princípios ágeis, incluindo Scrum, Kanban e Extreme Programming (Shore; Warden, 2021). Esse artefato serviu de base para compreender o mindset ágil e guiar as tomadas de decisão nos projetos subsequentes.

Nas disciplinas de Modelagem Ágil (MAG1 e MAG2), a aplicação prática desses conceitos foi concretizada através da modelagem de um Sistema de Gestão de Condomínio. Foram desenvolvidas histórias de usuário, diagramas de classe, sequência e telas, promovendo a transição da análise para a implementação de software. Essa etapa foi essencial para alinhar os requisitos de negócio às boas práticas de documentação leve e incremental, característica dos métodos ágeis (Ambler, 2002; Larman, 2007).

No eixo de Gerenciamento Ágil de Projetos (GAP1 e GAP2), foram aplicados conceitos como planejamento de releases e simulações com o Kanban Board Game, que ilustraram a importância da visualização do fluxo de trabalho e do controle da capacidade de entrega. Essas atividades destacaram o valor da iteração contínua e da adaptação frente a mudanças, princípios centrais do Manifesto Ágil (Beck et al., 2001).

As disciplinas técnicas como Introdução à Programação (INTRO), Banco de Dados (BD) e Aspectos Ágeis de Programação (AAP) reforçaram a importância da implementação orientada a testes (TDD) e da qualidade de código (Fowler, 2018). No projeto de INTRO, por exemplo, a meta de passar 40 testes unitários com JUnit exemplificou a mentalidade de entrega confiável e contínua de valor (Silva, 2014).

No módulo de Desenvolvimento Web (WEB1 e WEB2), a aplicação de Angular e Spring Boot evidenciou a construção de sistemas em camadas com separação de

responsabilidades (MVC), integrando front-end e back-end em CRUDs funcionais com banco de dados PostgreSQL. Esses projetos representaram uma simulação prática de squads ágeis, onde componentes são desenvolvidos em paralelo e integrados com repositórios compartilhados, como preconizado por práticas DevOps (Humble; Farley, 2010).

A disciplina de UX no Desenvolvimento Ágil (UX) complementou o processo com uma abordagem centrada no usuário, incluindo pesquisa de usabilidade, justificativa de design e coleta de feedback. Essa etapa demonstra como a experiência do usuário é incorporada iterativamente ao ciclo de desenvolvimento ágil, em consonância com práticas de Design Thinking (Brown, 2009).

Já em Mobile (MOB1 e MOB2), a criação de apps em Kotlin com integração via APIs externas consolidou habilidades de desenvolvimento multiplataforma, ampliando o escopo full-stack do aluno. Os projetos refletem a capacidade de entrega contínua com foco em valor para o usuário final.

Por fim, os módulos de Infraestrutura (INFRA) e Testes Automatizados (TEST) forneceram uma visão prática sobre integração contínua, versionamento e containers com Docker e GitLab, além da automação de testes com Playwright. Esses conhecimentos são fundamentais para garantir a escalabilidade e robustez dos sistemas em ambientes reais.

Conforme preconiza Sommerville (2011), o desenvolvimento de software moderno exige a articulação entre engenharia de requisitos, codificação, testes e deploy em ciclos curtos e contínuos. O memorial apresentado demonstra essa capacidade ao consolidar teoria, prática e integração tecnológica com metodologias ágeis. A formação obtida, portanto, não apenas promoveu o domínio técnico, mas preparou o aluno para atuar em contextos reais com alta adaptabilidade e visão sistêmica.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

A disciplina de Métodos Ágeis para Desenvolvimento de Software (MADS) teve como principal objetivo introduzir os fundamentos teóricos e práticos que sustentam o desenvolvimento ágil de software. Através da construção de um mapa mental, foram sistematizados conceitos essenciais como processos de software, manifesto ágil, princípios ágeis, modelos tradicionais, além de frameworks como Scrum, Kanban e Extreme Programming. Também foram abordadas práticas avançadas como Lean Software Development e Entrega Contínua.

Essa introdução conceitual foi fundamental para guiar todas as demais disciplinas do curso, servindo como alicerce para o entendimento dos papéis, artefatos, cerimônias e valores que norteiam as equipes ágeis. A clareza conceitual obtida nesta etapa permitiu que, ao longo do curso, as decisões técnicas fossem embasadas em princípios de agilidade como colaboração contínua com o cliente, resposta rápida às mudanças, entrega incremental de valor e foco em software funcional.

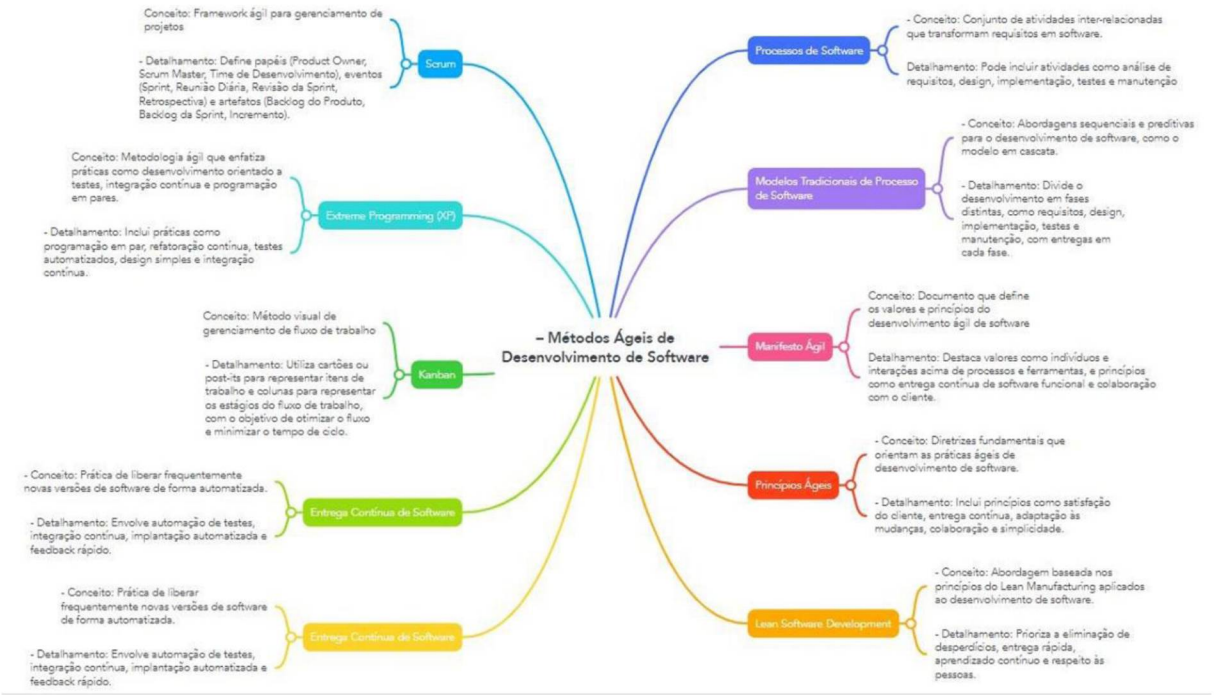
A partir dessa base, as demais disciplinas puderam explorar práticas específicas em suas áreas, como a modelagem ágil de requisitos (MAG1 e MAG2), a implementação de sistemas web e mobile com frameworks modernos (WEB1, WEB2, MOB1, MOB2), práticas de DevOps e testes automatizados. O conteúdo de MADS também foi essencial para a compreensão dos fluxos de trabalho abordados na disciplina de Gerenciamento Ágil de Projetos (GAP), especialmente na aplicação prática de Scrum e Kanban.

Dessa forma, MADS foi o ponto de partida para a internalização da mentalidade ágil no desenvolvimento de software, promovendo não apenas conhecimento técnico, mas uma mudança de paradigma sobre como desenvolver sistemas com maior eficiência, colaboração e foco em valor. Como reforçam Beck e Andres (2004), o ágil é mais do que um método é uma filosofia centrada em pessoas, feedback e melhoria contínua, princípios que orientaram a prática ao longo de todo o curso. A seguir, apresenta-se a versão textual do mapa mental elaborado com a ferramenta *MindX*:

2.1 ARTEFATOS DO PROJETO

A seguir, apresenta-se a Figura 1 que representa o mapa mental elaborado, abrangendo os principais tópicos e subtópicos exigidos:

FIGURA 1 – MAPA MENTAL



FONTE: O autor (2025).

3 DISCIPLINA: MAG1 E MAG2 – MODELAGEM ÁGIL DE SOFTWARE 1 E 2

Nas disciplinas MAG1 e MAG2, o principal foco foi desenvolver a modelagem ágil de um sistema de software realista, aplicando práticas alinhadas aos princípios ágeis. Durante essas etapas, elaborei a modelagem completa de um sistema de gestão, abrangendo desde as histórias de usuário até os diagramas técnicos, como diagramas de classes e diagramas de sequência, além dos desenhos de tela das interfaces.

A utilização das histórias de usuário foi fundamental para representar as funcionalidades sob a perspectiva do cliente final, garantindo maior alinhamento entre as necessidades do usuário e a entrega da equipe de desenvolvimento. Os desenhos de tela (mockups) ajudaram a visualizar o fluxo e a interação com o sistema de forma clara e acessível. Já os diagramas UML facilitaram a comunicação entre os envolvidos no projeto e contribuíram para a organização da arquitetura do sistema.

Essa modelagem ágil proporcionou uma base sólida para a etapa de codificação nas disciplinas seguintes, como INTRO (programação), BD (banco de dados), WEB1/WEB2 e MOB1/MOB2, além de estar fortemente integrada com as práticas de GAP1/GAP2 (gerenciamento ágil de projetos).

Em resumo, a modelagem ágil permitiu antecipar problemas, reduzir retrabalho e tornar o desenvolvimento mais eficiente e iterativo, refletindo o espírito dos métodos ágeis: entregar valor continuamente, com base no feedback do cliente e na adaptação constante do projeto.

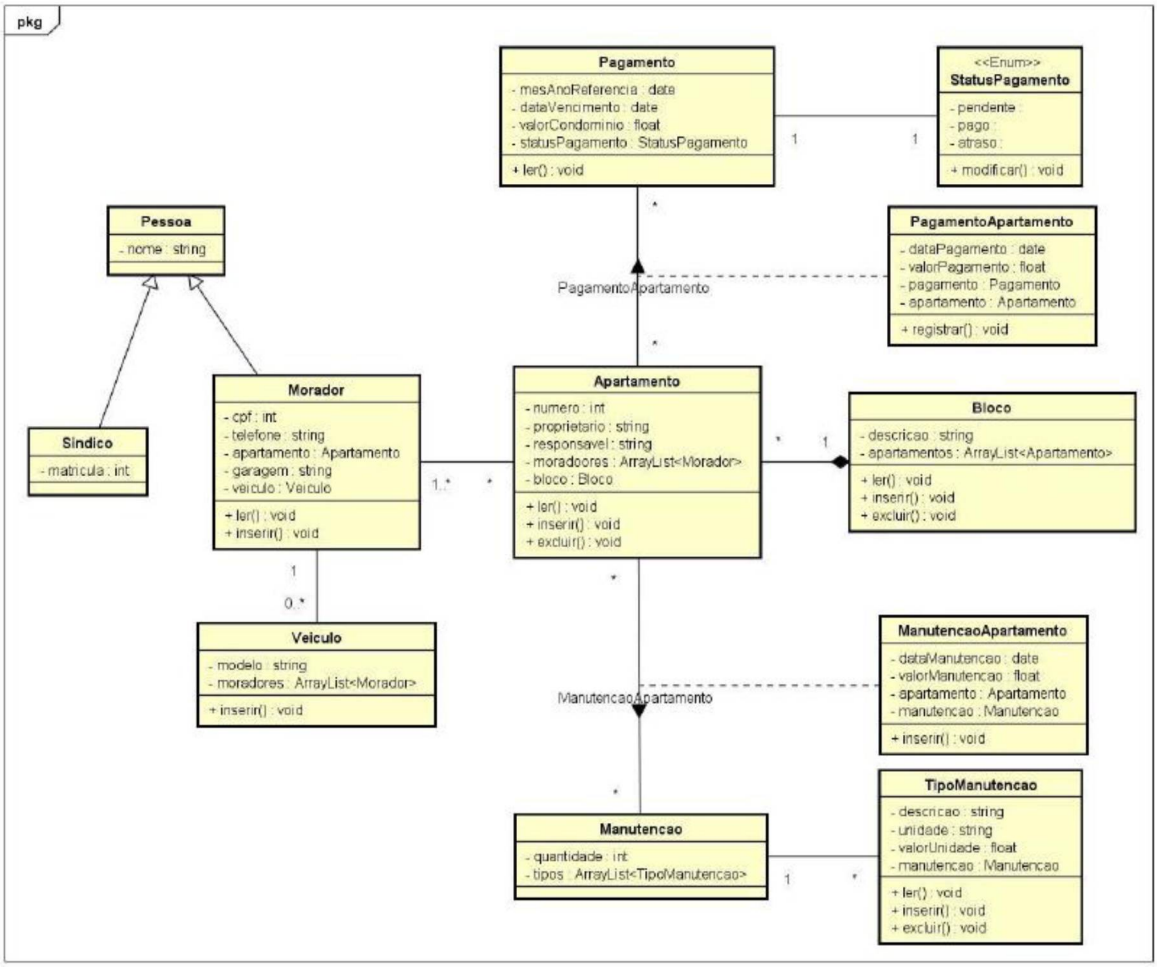
3.1 ARTEFATOS DO PROJETO

FIGURA 2 – CASO DE USO

História de Usuário	Caso de Uso
HU001	Pesquisar Morador
HU002	Manter Morador
HU003	Pesquisar Bloco
HU004	Manter Bloco
HU005	Pesquisar Apartamento
HU006	Manter apartamento
HU007	Pesquisar Tipo de Manutenção
HU008	Manter Tipo de Manutenção
HU009	Registrar Manutenção
HU010	Registrar Pagamento

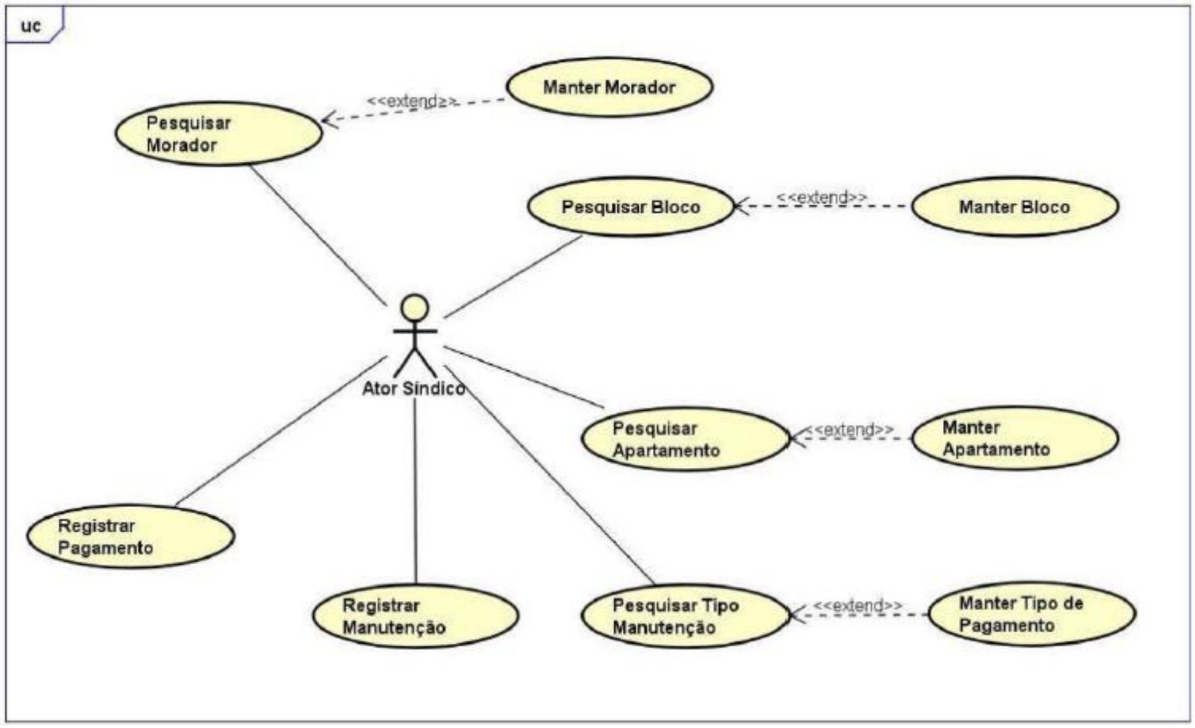
FONTE: O autor (2025).

FIGURA 3 – DIAGRAMA DE CLASSES



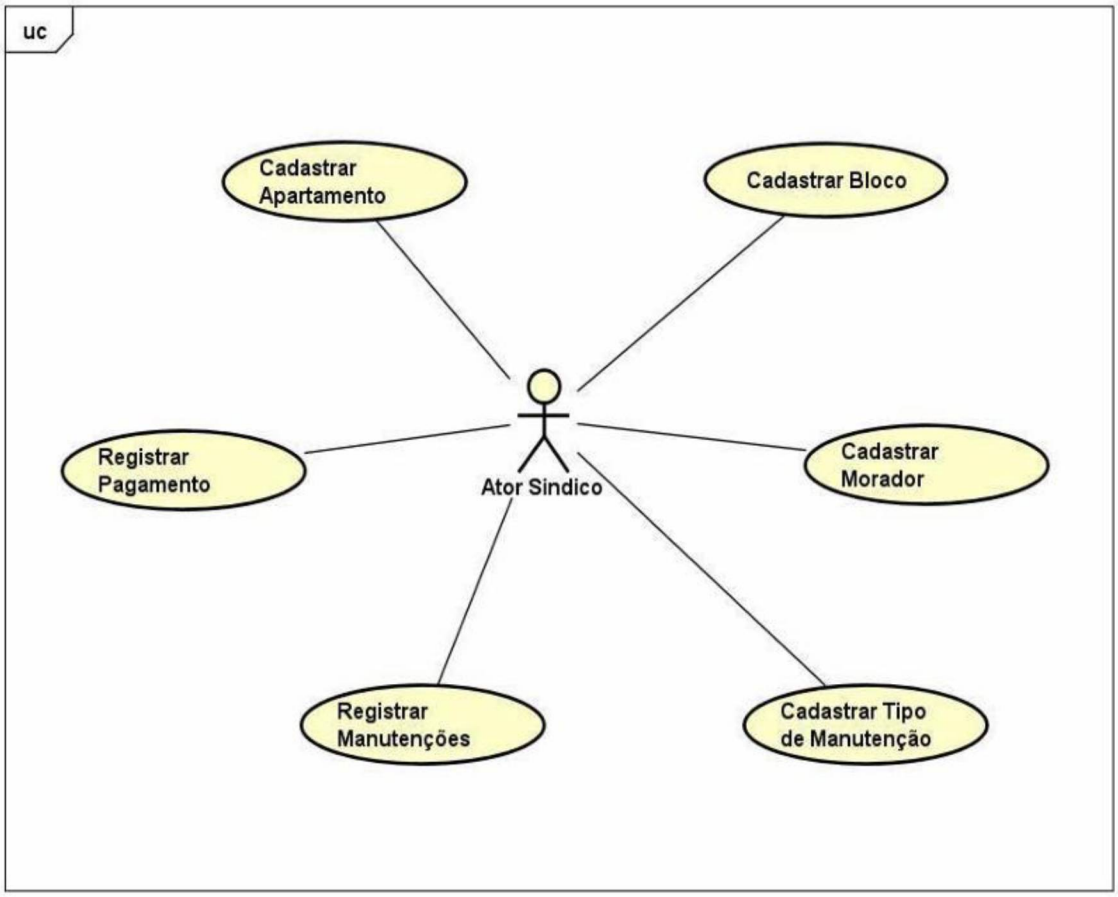
FONTE: O autor (2025).

FIGURA 4 – DIAGRAMA DE CASO DE USO DE NÍVEL 2



FONTE: O autor (2025).

Figura 5 – DIAGRAMA DE CASO DE USO DE NÍVEL 1



FONTE: O autor (2025).

Figura 6 – DESENHO DA TELA

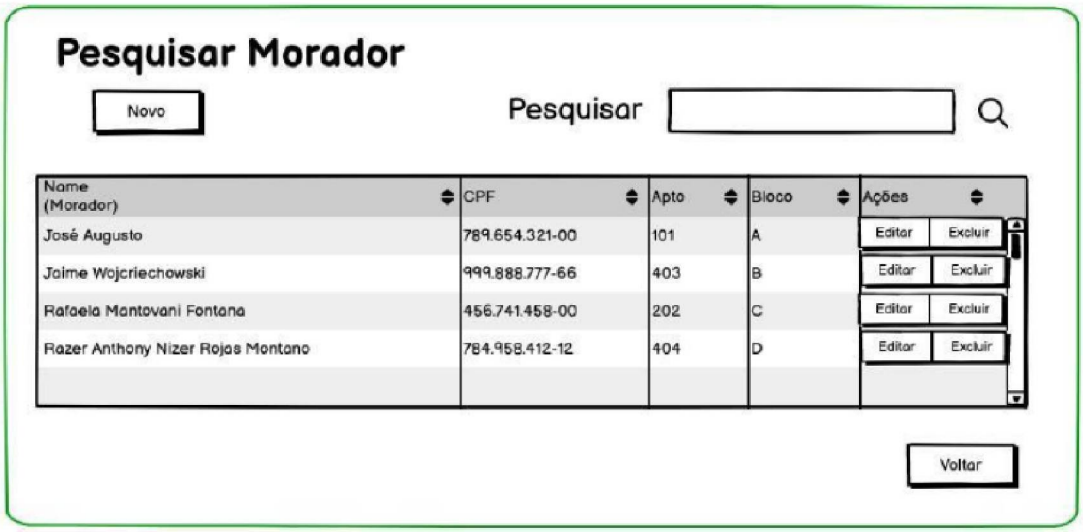
HU001 – Pesquisar Morador

SENDO O Síndico

QUERO Pesquisar por moradores

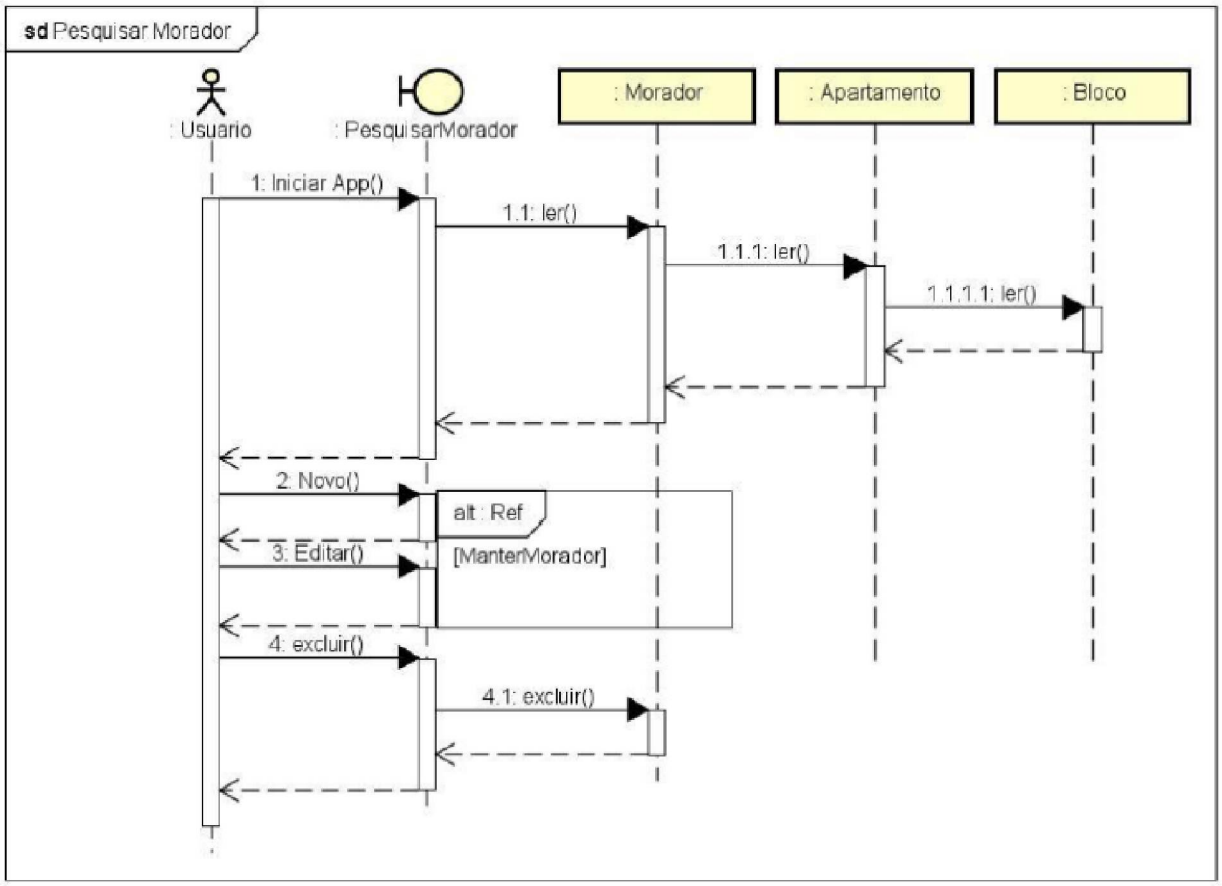
PARA Fazer manutenções nos seus dados

DESENHO DA TELA



FONTE: O autor (2025).

Figura 7 – DIAGRAMA DE SEQUÊNCIA



FONTE: O autor (2025).

4 DISCIPLINA: GAP1 E GAP2 – GERENCIAMENTO ÀGIL DE PROJETOS DE SOFTWARE 1 E 2

As disciplinas GAP1 e GAP2 tiveram papel fundamental para consolidar os fundamentos da gestão ágil no contexto do desenvolvimento de software. Em GAP1, o foco foi o planejamento de um projeto utilizando o conceito de plano de release com base em estimativas de esforço, estruturado em sprints. Esse planejamento incluiu histórias de usuário elaboradas no formato "Sendo/Quero/Para", conectando diretamente com as necessidades reais do cliente. A aplicação do conceito de velocidade (onde 1 ponto equivale a 8 horas) permitiu organizar as entregas de forma realista, respeitando o ritmo da equipe e priorizando a entrega de valor contínuo.

Em GAP2, a simulação prática por meio do ambiente KanbanBoardGame.com proporcionou uma vivência lúdica e realista da gestão de fluxo de trabalho em um ambiente ágil. A ferramenta simulou os desafios de adaptação e tomada de decisão que ocorrem na realidade, reforçando princípios como melhoria contínua, balanceamento de capacidade e identificação de gargalos.

Esses projetos integraram-se às demais disciplinas ao fornecerem uma base concreta para o desenvolvimento e organização das tarefas vistas em MAG1/MAG2 (modelagem de histórias e requisitos), WEB1/WEB2 e MOB1/MOB2 (implementações), TEST (validação) e DEVOPS (entrega contínua). A gestão estruturada permitiu acompanhar a evolução dos artefatos e garantir que o desenvolvimento ocorresse dentro de um ciclo ágil iterativo e incremental. Assim, GAP1 e GAP2 consolidam-se como disciplinas-chave para o sucesso de um projeto ágil, promovendo entregas constantes, controle de qualidade e resposta eficiente a mudanças.

4.1 ARTEFATOS DO PROJETO

Este plano de release foi elaborado com base nas diretrizes da disciplina GAP1 – Gerenciamento Ágil de Projetos de Software. O objetivo é organizar e distribuir o desenvolvimento de um software, considerando que será realizado por uma única pessoa e utilizando como referência a estimativa de 1 ponto equivalente a 8 horas de trabalho.

Cálculo da Velocidade:

- Considerando que 1 ponto equivale a 8 horas (um dia ideal de trabalho);
- Cada sprint terá duração de 10 dias úteis (2 semanas);
- O total de horas disponíveis por sprint é de 80 horas (10 dias x 8 horas/dia);
- Portanto, a velocidade da entrega é de 10 pontos por sprint.

Plano de Release por Sprint Sprint 1

Período: 01/05/2024 a 14/05/2024
Histórias de Usuário:

- HU01 – Sendo um usuário, quero criar um novo projeto para iniciar meu planejamento. (3 pontos / 24h)
- HU02 – Sendo um usuário, quero adicionar tarefas a um projeto para organizar o trabalho. (3 pontos / 24h)
- HU03 – Sendo um colaborador, quero ver as tarefas que me foram atribuídas para saber minhas responsabilidades. (2 pontos / 16h)
- HU04 – Sendo um usuário, quero marcar tarefas como concluídas para acompanhar meu progresso. (2 pontos / 16h)

Sprint 2
Período: 15/05/2024 a 28/05/2024
Histórias de Usuário:

- HU05 – Sendo um gerente, quero acompanhar o progresso das tarefas para avaliar o desempenho. (2 pontos / 16h)
- HU06 – Sendo um colaborador, quero atualizar o status das tarefas para manter o time informado. (2 pontos / 16h)
- HU07 – Sendo um usuário, quero receber atualizações sobre as tarefas para acompanhar mudanças. (3 pontos / 24h)
- HU08 – Sendo um gerente, quero reatribuir tarefas para redistribuir a carga de trabalho. (3 pontos / 24h)

Sprint 3
Período: 29/05/2024 a 11/06/2024
Histórias de Usuário:

- HU09 – Sendo um colaborador, quero adicionar comentários às tarefas para esclarecer dúvidas. (3 pontos / 24h)

- HU10 – Sendo um gerente, quero gerar relatórios de progresso para análise e apresentação. (3 pontos / 24h)
- HU11 – Sendo um usuário, quero ver o histórico de tarefas concluídas para revisar entregas. (2 pontos / 16h)
- HU12 – Sendo um gerente, quero planejar novas tarefas para as próximas etapas do projeto. (2 pontos / 16h)

Sprint 4

Período: 12/06/2024 a 25/06/2024

Histórias de Usuário:

- HU13 – Sendo um colaborador, quero compartilhar tarefas com colegas para dividir responsabilidades. (3 pontos / 24h)
- HU14 – Sendo um usuário, quero exportar tarefas para outros formatos para usá-las em outros sistemas. (3 pontos / 24h)
- HU15 – Sendo um administrador, quero configurar permissões de acesso para controlar quem vê o quê. (2 pontos / 16h)
- HU16 – Sendo um usuário, quero redefinir minha senha para manter minha conta segura. (2 pontos / 16h)

Todas as histórias foram escritas no formato SENDO / QUERO / PARA com a respectiva estimativa em pontos, respeitando o limite de 10 pontos por sprint. O planejamento contempla todas as sprints com datas definidas e tarefas distribuídas, como exigido na atividade.

GAP_01:

FIGURA 8 – PLANO DE RELEASE

Gerenciamento Ágil de Projetos I
Profa. Dra. Rafaela Mantovani Fontana
Template para o Plano de Release

Cálculo da Velocidade:

Horas disponíveis por dia:	8 horas	Tamanho da Sprint:	2 semanas (10 dias úteis)
Horas disponíveis por Sprint:	80 horas (10 dias * 8 horas por dia)	Velocidade:	10 pontos (80horas/8 horas por dia)

Plano de Release:

Iteração/Sprint 1	**Iteração/Sprint 2**	**Iteração/Sprint 3**	**Iteração/Sprint N**
Data Início: 01/05/2024	Data Início: 15/05/2024	Data Início: 29/05/2024	Data Início:12/06/2024
Data Fim: 14/05/2024	Data Fim: 28/05/2024	Data Fim: 11/06/2024	Data Fim:25/06/2024
História de Usuário 1 SENDO → Um gerente de projeto QUERO→Ser capaz de criar um novo projeto PARA → Iniciar o planejamento e a execução ESTIMATIVA → 3 pontos (24 horas de trabalho)	História de Usuário 5 SENDO → Um gerente de projeto QUERO → Ser capaz de acompanhar o progresso das tarefas PARA → Garantir que o projeto está no caminho certo ESTIMATIVA → 2 pontos (16 horas de trabalho)	História de Usuário 9 SENDO → Um membro da equipe QUERO → Ser capaz de adicionar comentários às minhas tarefas PARA → Compartilhar atualizações e informações relevantes com a equipe ESTIMATIVA → 3 pontos (24 horas de trabalho)	História de Usuário 13 SENDO → um usuário QUERO → poder compartilhar tarefas com colegas de equipe PARA → facilitar a colaboração e atribuição de atividades e ESTIMATIVA → 3 pontos (24 horas de trabalho)
História de Usuário 2 SENDO → Um gerente de projeto QUERO → Ser capaz de adicionar tarefas a um projeto PARA → Organizar o trabalho a ser feito ESTIMATIVA → 3 pontos (24 horas de trabalho)	História de Usuário 6 SENDO → Um membro da equipe QUERO → Ser capaz de atualizar o status das minhas tarefas PARA → Informar a equipe sobre o meu progresso ESTIMATIVA → 2 pontos (16 horas de trabalho)	História de Usuário 10 SENDO → Um gerente de projeto QUERO → Ser capaz de ver um relatório de progresso do projeto PARA → Ter uma visão geral do status do projeto ESTIMATIVA → 3 pontos (24 horas de trabalho)	História de Usuário 14 SENDO → um usuário QUERO → poder exportar minha lista de tarefas para outros formatos, e PARA → integração com outras ferramentas de gerenciamento e ESTIMATIVA → 3 pontos (24 horas de trabalho)
História de Usuário 3 SENDO → Um membro da equipe QUERO → Ser capaz de ver as tarefas que me foram atribuídas PARA → Saber o que preciso fazer e priorizar meu trabalho ESTIMATIVA → 2 pontos (16 horas de trabalho)	História de Usuário 7 SENDO → Um stakeholder do projeto QUERO → Receber atualizações regulares sobre o progresso do projeto PARA → Ficar informado e tomar decisões baseadas em dados ESTIMATIVA → 3 pontos (24 horas de trabalho)	História de Usuário 11 SENDO → Um stakeholder do projeto QUERO → Ser capaz de ver o histórico de tarefas concluídas PARA → Avaliar o desempenho da equipe ESTIMATIVA → 2 pontos (16 horas de trabalho)	História de Usuário 15 SENDO → um administrador do sistema QUERO → poder configurar permissões de acesso PARA → garantir que apenas usuários autorizados possam acessar determinadas funcionalidades ESTIMATIVA → 2 pontos (16 horas de trabalho)
História de Usuário 4 SENDO → Um membro da equipe QUERO → Ser capaz de marcar uma tarefa como concluída PARA → Informar que terminei a tarefa ESTIMATIVA → 2 pontos (16 horas de trabalho)	História de Usuário 8 SENDO → Um gerente de projeto QUERO → Ser capaz de reatribuir tarefas se necessário PARA → Assegurar que o trabalho seja feito de forma eficiente ESTIMATIVA → 3 pontos (24 horas de trabalho)	História de Usuário 12 SENDO → Um membro da equipe QUERO → Ser capaz de ver as tarefas pendentes PARA → Planejar meu trabalho ESTIMATIVA → 2 pontos (16 horas de trabalho)	História de Usuário 16 SENDO → um usuário QUERO → ter a opção de redefinir minha senha PARA → recuperar o acesso à minha conta caso eu a esqueça ou ela seja comprometida ESTIMATIVA → 2 pontos (16 horas de trabalho)

FONTE: O autor (2025).

GAP_02 (Execução do Kanban Board Game):

A segunda parte do trabalho consistiu em simular a gestão de um projeto utilizando o ambiente virtual da plataforma kanbanboardgame.com, no modo “Standard Game”, com duração de 35 dias.

A execução prática permitiu colocar em uso conceitos aprendidos durante a disciplina, como gestão visual, controle de fluxo, análise de gargalos, limites de WIP e fluxo contínuo.

- Print do dia 15:

A imagem mostra o quadro Kanban no dia 15 da simulação. É possível observar a evolução das tarefas, divididas entre as colunas: Pendências, Pronto, Análise, Desenvolvimento, Teste e Implantado. Nesse ponto, o saldo era de US$ 2.480, com várias tarefas em progresso e algumas já entregues.

- Resultado final do jogo:

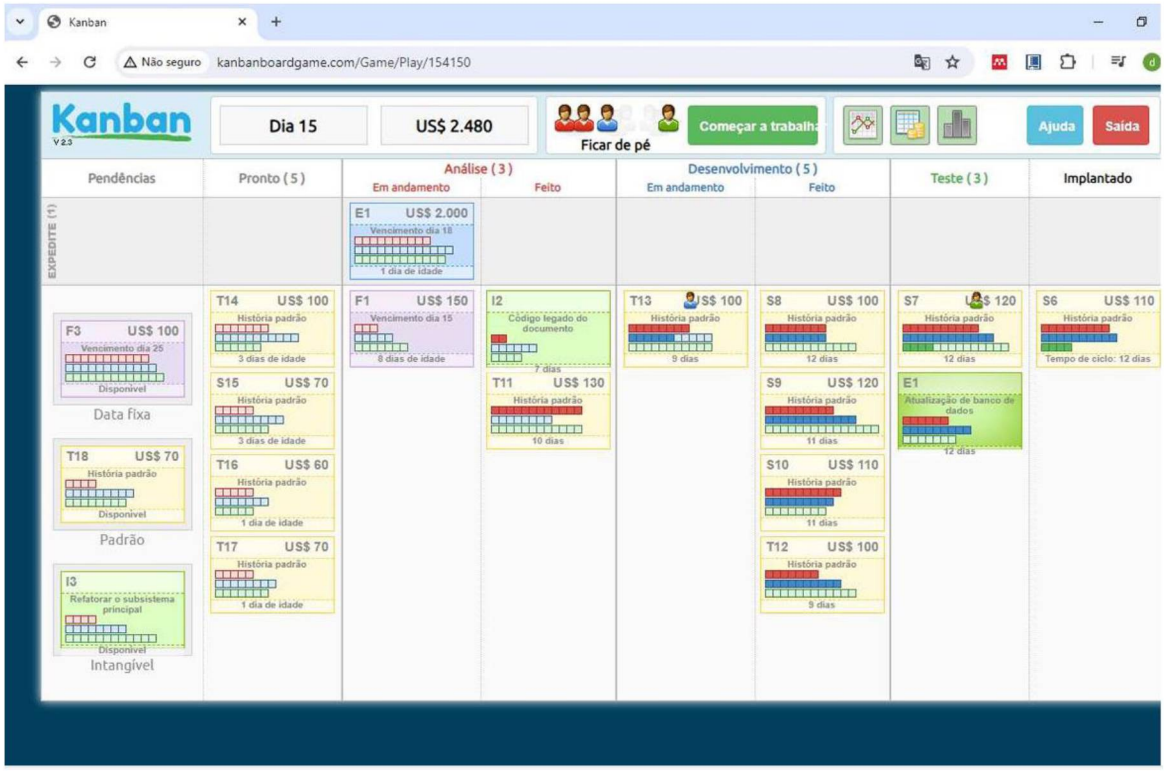
Ao término dos 35 dias, o jogo foi concluído com receita final de US$ 33.840, conforme indicado na imagem da tela de encerramento com o status Game Over.

- Diagrama de fluxo cumulativo (CFD):

O gráfico CFD gerado pela ferramenta apresenta a evolução do fluxo de trabalho ao longo dos 35 dias. É possível identificar claramente o volume de tarefas em cada etapa (Preparar, Análise, Desenvolvimento, Teste e Implantado), evidenciando a progressão contínua e os momentos de maior acúmulo, além de uma boa taxa de entrega.

Print de telas:

FIGURA 9 – PRINT DA TELA DA EXECUÇÃPO DO JOGO



FONTE: O autor (2025).

FIGURA 10 – PRINT DA TELA DO RESULTADO EXIBINDO A RECEITA

FONTE: O autor (2025).

FIGURA 11 – PRINT FA TELA DO CFD

FONTE: O autor (2025).

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação tem papel fundamental na formação de profissionais voltados ao desenvolvimento ágil de software. O projeto desenvolvido nesse módulo consistiu na construção do backend de um sistema bancário simplificado, com foco em cadastro e manipulação de contas correntes e de investimento. O desafio envolveu implementar as classes do sistema utilizando os princípios da programação orientada a objetos, de modo que fossem compatíveis com os testes unitários previamente definidos em JUnit. A exigência de deixar ao menos 40 testes funcionando corretamente proporcionou uma experiência prática de extrema relevância, reforçando a importância da qualidade e da confiabilidade do código desde as etapas iniciais do desenvolvimento.

Essa abordagem favorece o alinhamento com os princípios do desenvolvimento ágil, especialmente no que tange à entrega contínua e incremental de valor, à manutenção de um design limpo e à adoção de testes automatizados como ferramenta de validação contínua. Além disso, o uso de banco de dados com script DDL fornecido amplia a visão prática da integração entre aplicação e persistência de dados, reforçando o conceito de software funcional e preparado para evoluções.

Sistema de Conta Corrente:

Casos de Teste

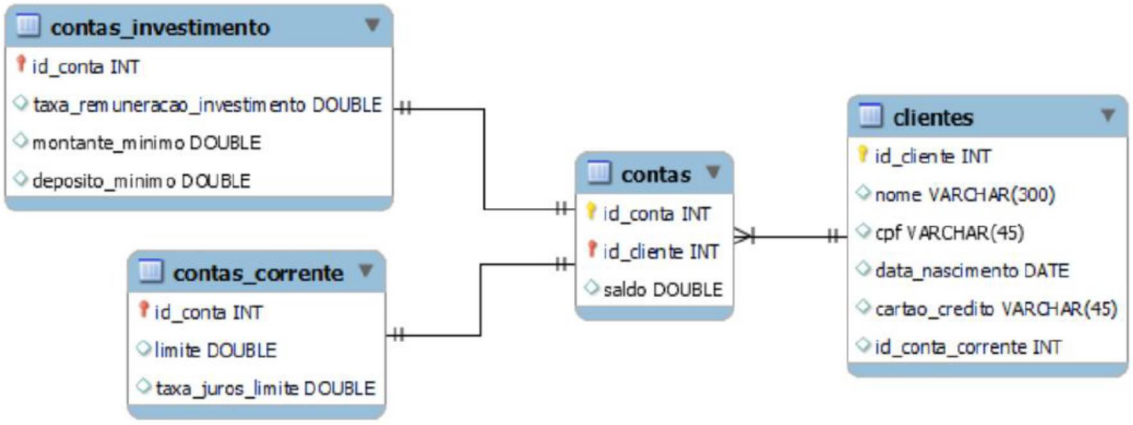
Os casos de testes estão escritos no JUnit, dentro do projeto que está disponível para vocês no Moodle. São 42 casos de teste no total. Você deve implementar o software de modo a deixar 40 casos de testes no verde simultaneamente. Cada caso teste verde simultaneamente vale 2,5 pontos. Com 40 casos de teste verdes simultâneos você terá atingido a nota máxima.

Entrega

O trabalho foi entregue no moodle da instituição. Cada aluno entregou um zip do trabalho contendo o projeto com as classes implementadas, juntamente com os casos de testes. Além do projeto, o aluno postou um PDF contendo o relatório do trabalho contendo uma única página, com as evidências dos testes que funcionaram, coletadas na *IDE* do *Netbeans* em formato de imagem.

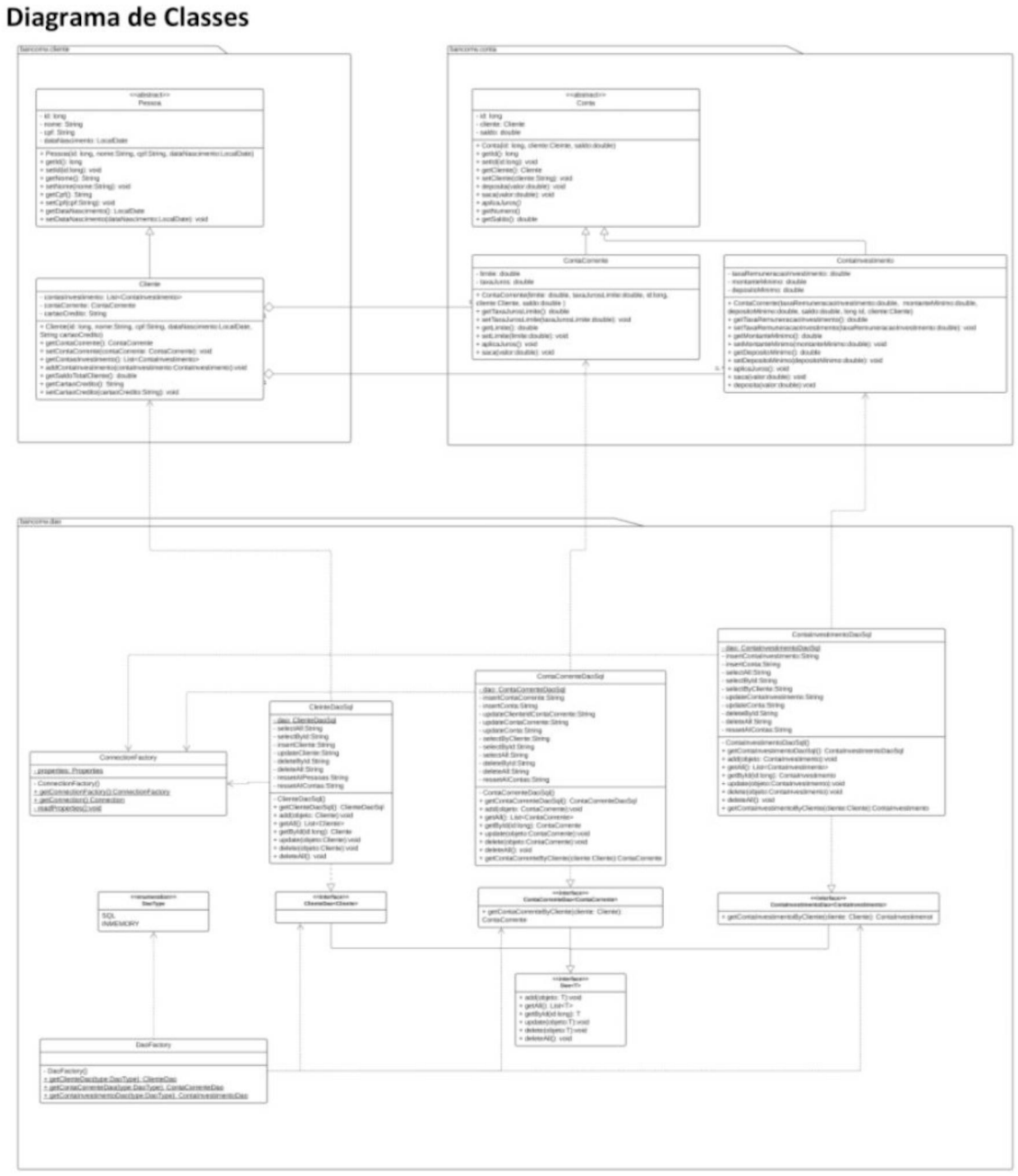
5.1 ARTEFATOS DO PROJETO

FIGURA 12 – DIAGRMA ER DO BANCO CRIADO



FONTE: O autor (2025).

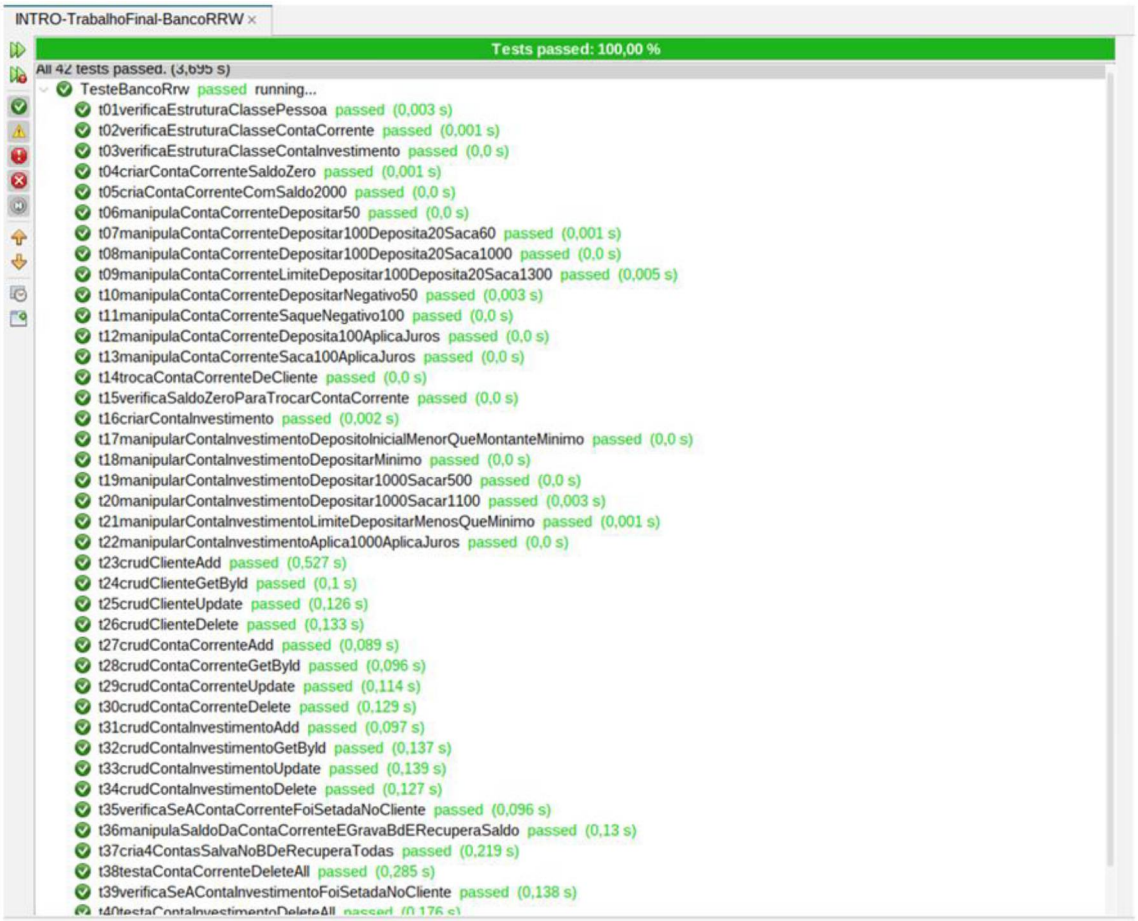
FIGURA 13 – DIAGRAMA DE CLASSES



FONTE: O autor (2025).

FIGURA 14 - CASOS DE TESTE

100% dos 42 testes ficaram “verdes” e funcionaram.



FONTE: O autor (2025).

6 DISCIPLINA: BD – BANCO DE DADOS

O projeto da disciplina de Banco de Dados teve como foco a modelagem e implementação de um sistema robusto de controle de estoque, atendendo às exigências práticas de um cenário empresarial real. Foi elaborado em equipe, com atividades divididas entre modelagem conceitual (entidades e relacionamentos), modelagem lógica (tabelas, chaves primárias e estrangeiras) e implementação via scripts SQL (DDL e DML). O sistema permitiu o cadastro e gerenciamento de produtos, fornecedores, funcionários, localizações e operações como abastecimento, desabastecimento e reposição de estoque.

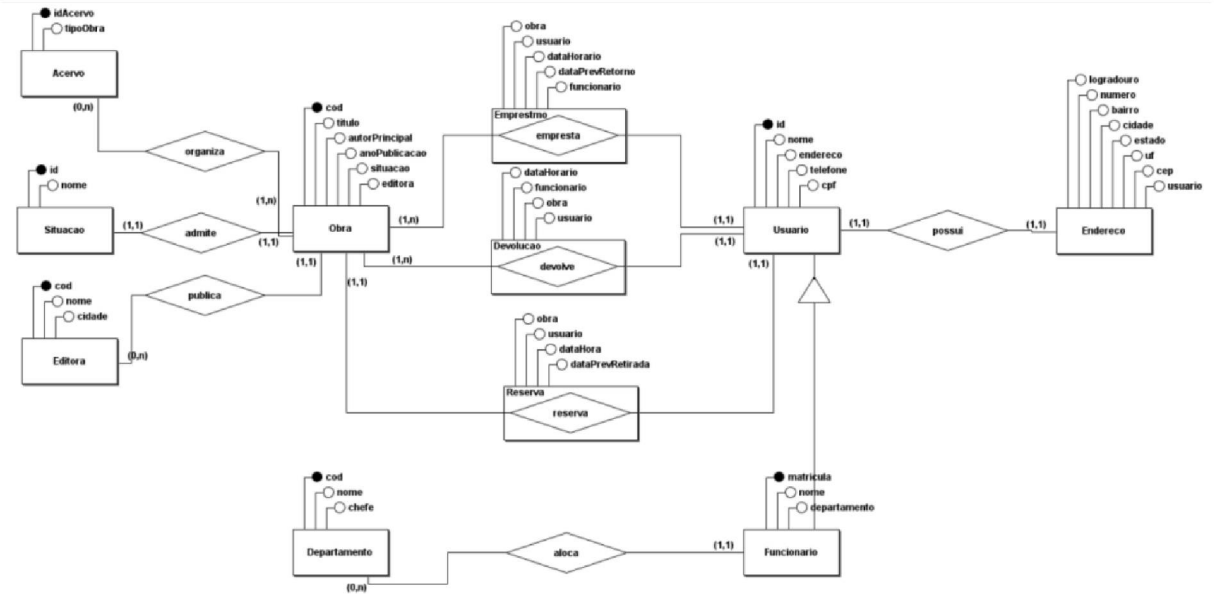
No contexto do desenvolvimento ágil de software, a disciplina de Banco de Dados mostrou-se essencial para garantir a persistência e integridade dos dados, promovendo entregas contínuas e incrementais com base em dados reais. O conhecimento adquirido foi fundamental para as etapas de backend nas disciplinas de desenvolvimento Web, Mobile e testes automatizados, possibilitando a criação de aplicações funcionais com acesso estruturado e confiável ao banco de dados.

Além disso, a prática com modelagem e normalização permitiu uma integração natural com as fases de levantamento de requisitos e modelagem ágil (disciplinas MAG1/MAG2), onde os diagramas de classes e de sequência puderam ser convertidos diretamente em estruturas de dados relacionais. Essa sinergia entre modelagem, implementação e uso real dos dados foi uma evidência concreta da aplicação dos princípios ágeis, como colaboração contínua, entrega frequente e adaptação às mudanças.

Por fim, o domínio das operações em SQL, juntamente com a organização eficiente dos dados, fortaleceu a base para a construção de APIs e sistemas com maior escalabilidade e desempenho, reforçando a importância da disciplina de Banco de Dados como um pilar técnico e estratégico dentro do ciclo ágil de desenvolvimento de software.

6.1 ARTEFATOS DO PROJETO

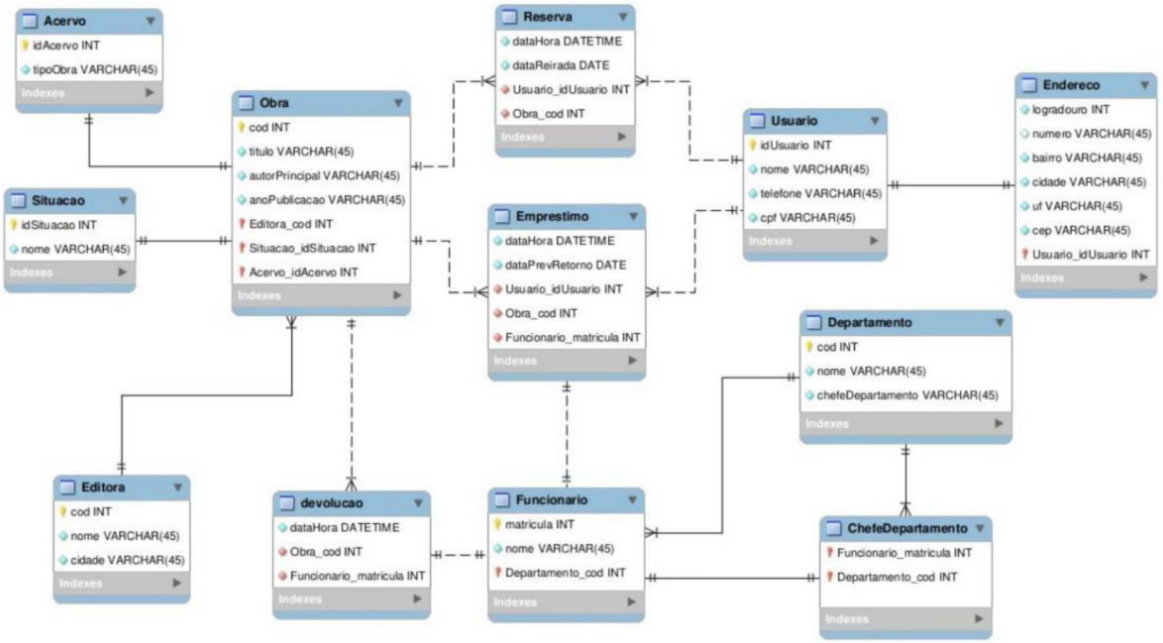
FIGURA 15 – MODELO ENTIDADE-RELACIONAMENTO CONCEITUAL



FONTE: O autor (2025).

O modelo entidade-relacionamento conceitual é desenvolvido em ferramentas como o brModelo e apresenta apenas as entidades e os relacionamentos existentes entre elas, sem entrar em detalhes técnicos de implementação. Já o modelo lógico, representado pelo diagrama entidade-relacionamento (DER), transforma essas entidades em tabelas, detalhando os campos de cada uma, além de identificar as chaves primárias e estrangeiras, mantendo os relacionamentos definidos no modelo conceitual.

FIGURA16 – MODELO LÓGIVO – DIAGRAMA DE ENTIDADE RELACIONAMENTO (TABELAS)



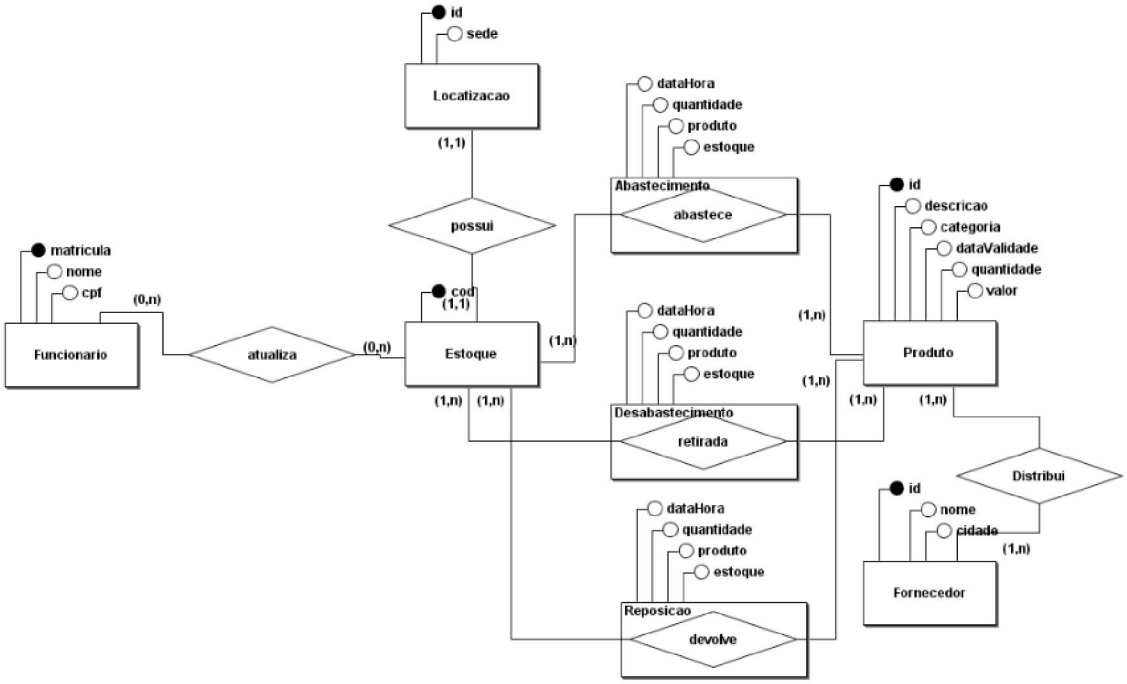
FONTE: O autor (2025).

Questão 2 – Especificação e Implementação em SQL. Tema: Controle de Estoque

O controle de estoque abrange as sedes de um grupo empresarial, onde cada sede possui um único estoque identificado por código e nome. Os produtos armazenados possuem código, descrição, categoria, validade, quantidade e valor, e são fornecidos por diferentes fornecedores, que possuem identificação, nome e cidade. A movimentação do estoque é feita por funcionários identificados por matrícula e nome.

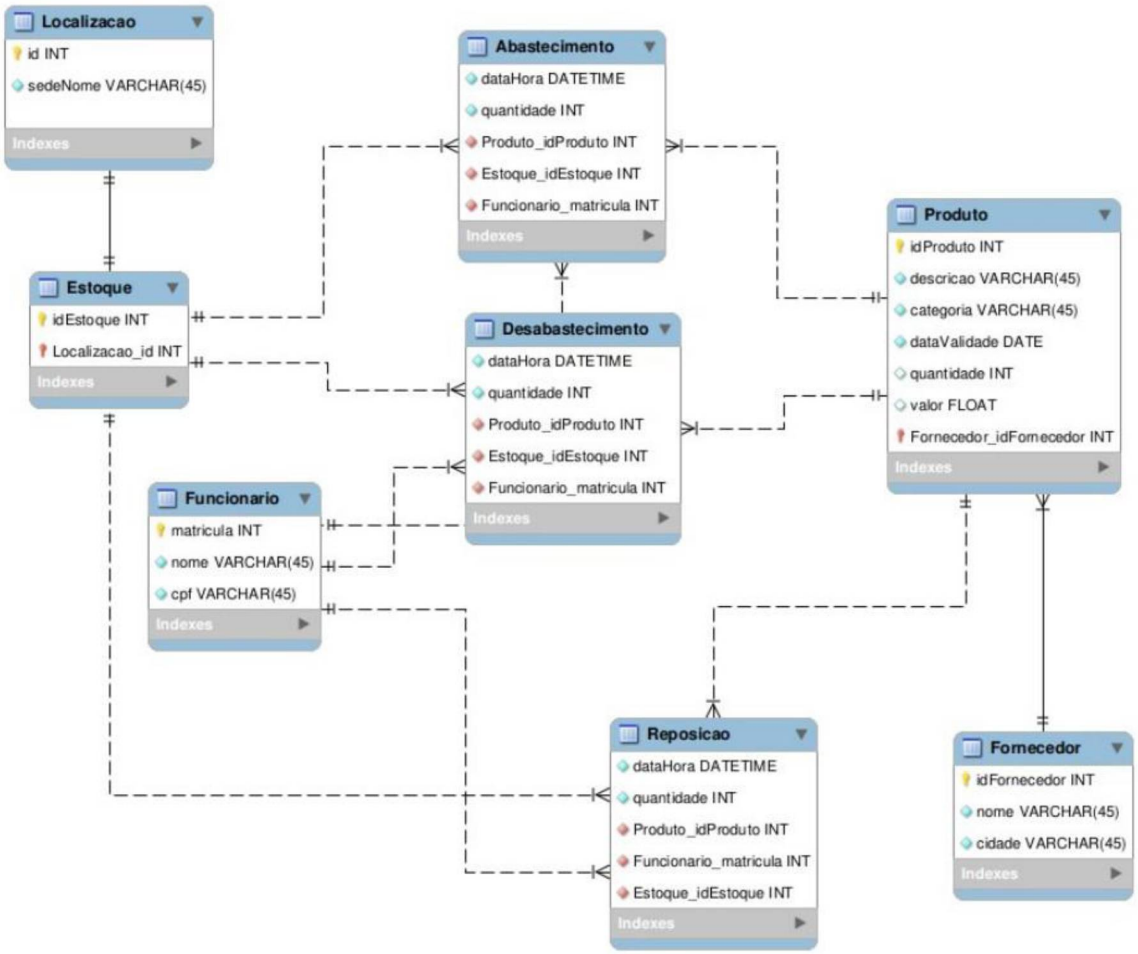
Todas as saídas (retiradas) e entradas (abastecimentos e devoluções) de produtos devem ser registradas com data, hora, quantidade, local do estoque e responsável pela ação. Em casos de devolução entre sedes, o estoque de destino é atualizado com os dados da transferência. O objetivo é garantir um controle preciso e atualizado de todos os estoques da empresa.

FIGURA 17 – MODELO ENTIDADE – RELACIONAMENTO CONCEITUAL



FONTE: O autor (2025).

FIGURA 18 – MODELO LÓGICO – DIAGRAMA DE ENTIDADE DE RELACIONAMENTO



FONTE: O autor (2025).

FIGURA 19 – SCRIPT DDL (TABELAS) – PARTE 1

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Acervo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Acervo` (
  `idAcervo` INT NOT NULL AUTO_INCREMENT,
  `tipoObra` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idAcervo`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Situacao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Situacao` (
  `idSituacao` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idSituacao`))
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 20 – SCRIPT DDL (TABELAS) – PARTE 2

```
-- -----------------------------------------------------
-- Table `mydb`.`Editora`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Editora` (
  `cod` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `cidade` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`cod`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Obra`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Obra` (
  `cod` INT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(45) NOT NULL,
  `autorPrincipal` VARCHAR(45) NOT NULL,
  `anoPublicacao` VARCHAR(45) NOT NULL,
  `Editora_cod` INT NOT NULL,
  `Situacao_idSituacao` INT NOT NULL,
  `Acervo_idAcervo` INT NOT NULL,
  PRIMARY KEY (`cod`, `Editora_cod`, `Situacao_idSituacao`, `Acervo_idAcervo`),
  INDEX `fk_Obra_Editora1_idx` (`Editora_cod` ASC) VISIBLE,
  INDEX `fk_Obra_Situacao1_idx` (`Situacao_idSituacao` ASC) VISIBLE,
  INDEX `fk_Obra_Acervo1_idx` (`Acervo_idAcervo` ASC) VISIBLE,
  CONSTRAINT `fk_Obra_Editora1`
    FOREIGN KEY (`Editora_cod`)
    REFERENCES `mydb`.`Editora` (`cod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Obra_Situacao1`
    FOREIGN KEY (`Situacao_idSituacao`)
    REFERENCES `mydb`.`Situacao` (`idSituacao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Obra_Acervo1`
    FOREIGN KEY (`Acervo_idAcervo`)
    REFERENCES `mydb`.`Acervo` (`idAcervo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 21 – SCRIPT DDL (TABELAS) – PARTE 3

```
-- -----------------------------------------------------
-- Table `mydb`.`devolucao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`devolucao` (
  `dataHora` DATETIME NOT NULL,
  `Obra_cod` INT NOT NULL,
  `Funcionario_matricula` INT NOT NULL,
  INDEX `fk_devolucao_Obra1_idx` (`Obra_cod` ASC) VISIBLE,
  INDEX `fk_devolucao_Funcionario1_idx` (`Funcionario_matricula` ASC) VISIBLE,
  CONSTRAINT `fk_devolucao_Obra1`
    FOREIGN KEY (`Obra_cod`)
    REFERENCES `mydb`.`Obra` (`cod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_devolucao_Funcionario1`
    FOREIGN KEY (`Funcionario_matricula`)
    REFERENCES `mydb`.`Funcionario` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Usuario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Usuario` (
  `idUsuario` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `telefone` VARCHAR(45) NOT NULL,
  `cpf` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idUsuario`))
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 22 – SCRIPT DDL (TABELAS) – PARTE 4

```
-- -----------------------------------------------------
-- Table `mydb`.`Emprestimo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Emprestimo` (
  `dataHora` DATETIME NOT NULL,
  `dataPrevRetorno` DATE NOT NULL,
  `Usuario_idUsuario` INT NOT NULL,
  `Obra_cod` INT NOT NULL,
  `Funcionario_matricula` INT NOT NULL,
  INDEX `fk_Emprestimo_Usuario1_idx` (`Usuario_idUsuario` ASC) VISIBLE,
  INDEX `fk_Emprestimo_Obra1_idx` (`Obra_cod` ASC) VISIBLE,
  INDEX `fk_Emprestimo_Funcionario1_idx` (`Funcionario_matricula` ASC) VISIBLE,
  CONSTRAINT `fk_Emprestimo_Usuario1`
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `mydb`.`Usuario` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Emprestimo_Obra1`
    FOREIGN KEY (`Obra_cod`)
    REFERENCES `mydb`.`Obra` (`cod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Emprestimo_Funcionario1`
    FOREIGN KEY (`Funcionario_matricula`)
    REFERENCES `mydb`.`Funcionario` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Reserva`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Reserva` (
  `dataHora` DATETIME NOT NULL,
  `dataReirada` DATE NOT NULL,
  `Usuario_idUsuario` INT NOT NULL,
  `Obra_cod` INT NOT NULL,
  INDEX `fk_Reserva_Usuario1_idx` (`Usuario_idUsuario` ASC) VISIBLE,
  INDEX `fk_Reserva_Obra1_idx` (`Obra_cod` ASC) VISIBLE,
  CONSTRAINT `fk_Reserva_Usuario1`
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `mydb`.`Usuario` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Reserva_Obra1`
    FOREIGN KEY (`Obra_cod`)
    REFERENCES `mydb`.`Obra` (`cod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 23 – SCRIPT DDL (TABELAS) – PARTE 5

```
-- -----------------------------------------------------
-- Table `mydb`.`Endereco`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Endereco` (
  `logradouro` INT NOT NULL,
  `numero` VARCHAR(45) NULL,
  `bairro` VARCHAR(45) NOT NULL,
  `cidade` VARCHAR(45) NOT NULL,
  `uf` VARCHAR(45) NOT NULL,
  `cep` VARCHAR(45) NOT NULL,
  `Usuario_idUsuario` INT NOT NULL,
  PRIMARY KEY (`Usuario_idUsuario`),
  CONSTRAINT `fk_Endereco_Usuario`
    FOREIGN KEY (`Usuario_idUsuario`)
    REFERENCES `mydb`.`Usuario` (`idUsuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`ChefeDepartamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`ChefeDepartamento` (
  `Funcionario_matricula` INT NOT NULL,
  `Departamento_cod` INT NOT NULL,
  PRIMARY KEY (`Funcionario_matricula`, `Departamento_cod`),
  INDEX `fk_ChefeDepartamento_Departamento1_idx` (`Departamento_cod` ASC) VISIBLE,
  CONSTRAINT `fk_ChefeDepartamento_Funcionario1`
    FOREIGN KEY (`Funcionario_matricula`)
    REFERENCES `mydb`.`Funcionario` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ChefeDepartamento_Departamento1`
    FOREIGN KEY (`Departamento_cod`)
    REFERENCES `mydb`.`Departamento` (`cod`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Funcionario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Funcionario` (
  `matricula` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `cpf` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`matricula`))
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 24 – SCRIPT DDL (TABELAS) – PARTE 6

```
-- -----------------------------------------------------
-- Table `mydb`.`Localizacao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Localizacao` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `sedeNome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Fornecedor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Fornecedor` (
  `idFornecedor` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `cidade` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idFornecedor`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Produto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Produto` (
  `idProduto` INT NOT NULL AUTO_INCREMENT,
  `descricao` VARCHAR(45) NOT NULL,
  `categoria` VARCHAR(45) NOT NULL,
  `dataValidade` DATE NOT NULL,
  `quantidade` INT NULL,
  `valor` FLOAT NULL,
  `Fornecedor_idFornecedor` INT NOT NULL,
  PRIMARY KEY (`idProduto`, `Fornecedor_idFornecedor`),
  INDEX `fk_Produto_Fornecedor1_idx` (`Fornecedor_idFornecedor` ASC) VISIBLE,
  CONSTRAINT `fk_Produto_Fornecedor1`
    FOREIGN KEY (`Fornecedor_idFornecedor`)
    REFERENCES `mydb`.`Fornecedor` (`idFornecedor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 25 – SCRIPT DDL (TABELAS) – PARTE 7

```
-- -----------------------------------------------------
-- Table `mydb`.`Estoque`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Estoque` (
  `idEstoque` INT NOT NULL AUTO_INCREMENT,
  `Localizacao_id` INT NOT NULL,
  PRIMARY KEY (`idEstoque`),
  INDEX `fk_Estoque_Localizacao1_idx` (`Localizacao_id` ASC) VISIBLE,
  CONSTRAINT `fk_Estoque_Localizacao1`
    FOREIGN KEY (`Localizacao_id`)
    REFERENCES `mydb`.`Localizacao` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Abastecimento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Abastecimento` (
  `dataHora` DATETIME NOT NULL,
  `quantidade` INT NOT NULL,
  `Produto_idProduto` INT NOT NULL,
  `Estoque_idEstoque` INT NOT NULL,
  `Funcionario_matricula` INT NOT NULL,
  INDEX `fk_Abastecimento_Produto1_idx` (`Produto_idProduto` ASC) VISIBLE,
  INDEX `fk_Abastecimento_Estoque1_idx` (`Estoque_idEstoque` ASC) VISIBLE,
  INDEX `fk_Abastecimento_Funcionario1_idx` (`Funcionario_matricula` ASC) VISIBLE,
  CONSTRAINT `fk_Abastecimento_Produto1`
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `mydb`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Abastecimento_Estoque1`
    FOREIGN KEY (`Estoque_idEstoque`)
    REFERENCES `mydb`.`Estoque` (`idEstoque`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Abastecimento_Funcionario1`
    FOREIGN KEY (`Funcionario_matricula`)
    REFERENCES `mydb`.`Funcionario` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

FONTE: O autor (2025).

FIGURA 26 – SCRIPT DDL (TABELAS) – PARTE 8

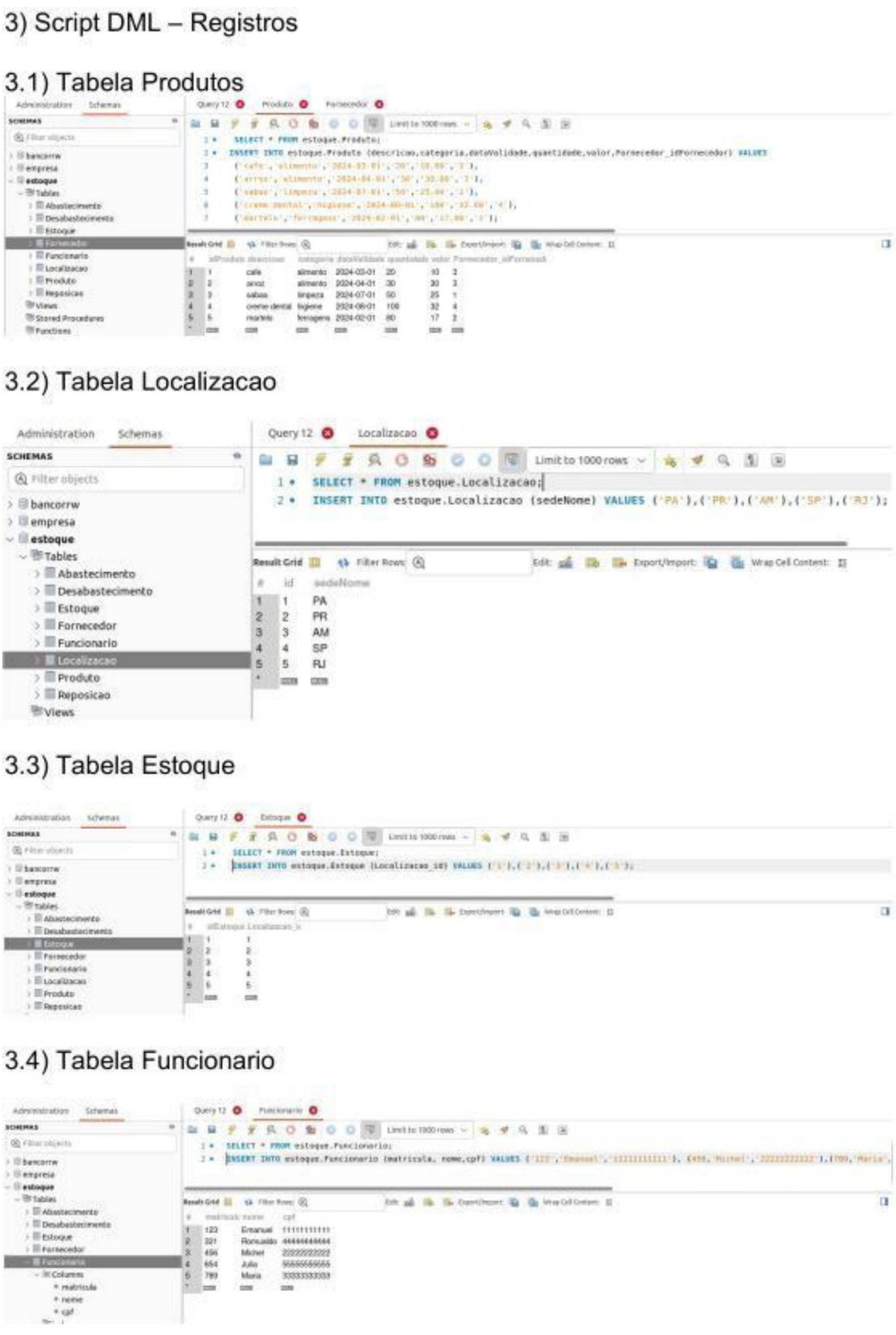
```
-- -----------------------------------------------------
-- Table `mydb`.`Reposicao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Reposicao` (
  `dataHora` DATETIME NOT NULL,
  `quantidade` INT NOT NULL,
  `Produto_idProduto` INT NOT NULL,
  `Funcionario_matricula` INT NOT NULL,
  `Estoque_idEstoque` INT NOT NULL,
  INDEX `fk_Reposicao_Produto1_idx` (`Produto_idProduto` ASC) VISIBLE,
  INDEX `fk_Reposicao_Funcionario1_idx` (`Funcionario_matricula` ASC) VISIBLE,
  INDEX `fk_Reposicao_Estoque1_idx` (`Estoque_idEstoque` ASC) VISIBLE,
  CONSTRAINT `fk_Reposicao_Produto1`
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `mydb`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Reposicao_Funcionario1`
    FOREIGN KEY (`Funcionario_matricula`)
    REFERENCES `mydb`.`Funcionario` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Reposicao_Estoque1`
    FOREIGN KEY (`Estoque_idEstoque`)
    REFERENCES `mydb`.`Estoque` (`idEstoque`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Desabastecimento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Desabastecimento` (
  `dataHora` DATETIME NOT NULL,
  `quantidade` INT NOT NULL,
  `Produto_idProduto` INT NOT NULL,
  `Estoque_idEstoque` INT NOT NULL,
  `Funcionario_matricula` INT NOT NULL,
  INDEX `fk_Desabastecimento_Produto1_idx` (`Produto_idProduto` ASC) VISIBLE,
  INDEX `fk_Desabastecimento_Estoque1_idx` (`Estoque_idEstoque` ASC) VISIBLE,
  INDEX `fk_Desabastecimento_Funcionario1_idx` (`Funcionario_matricula` ASC) VISIBLE,
  CONSTRAINT `fk_Desabastecimento_Produto1`
    FOREIGN KEY (`Produto_idProduto`)
    REFERENCES `mydb`.`Produto` (`idProduto`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Desabastecimento_Estoque1`
    FOREIGN KEY (`Estoque_idEstoque`)
    REFERENCES `mydb`.`Estoque` (`idEstoque`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Desabastecimento_Funcionario1`
    FOREIGN KEY (`Funcionario_matricula`)
    REFERENCES `mydb`.`Funcionario` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

FONTE: O autor (2025).

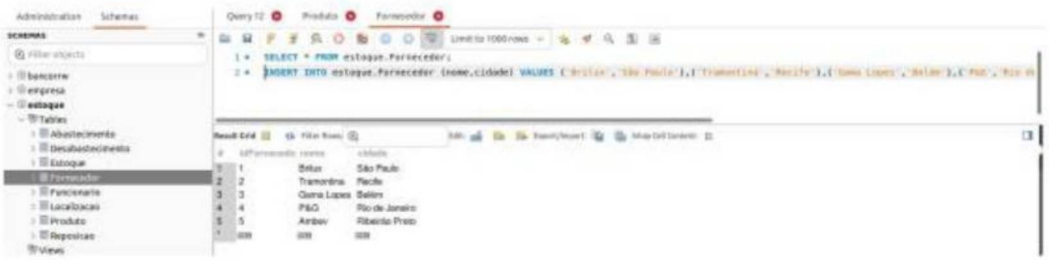
FIGURA 27 – SCRIPT DML (REGISTROS) – PARTE 1



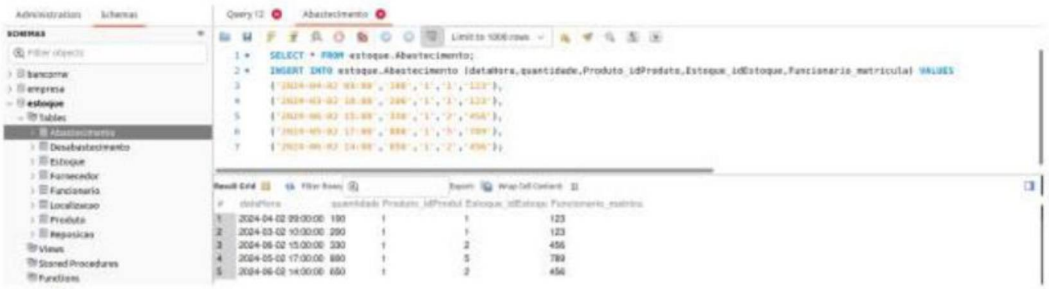
FONTE: O autor (2025).

FIGURA 28 – SCRIPT DML (REGISTROS) – PARTE 2

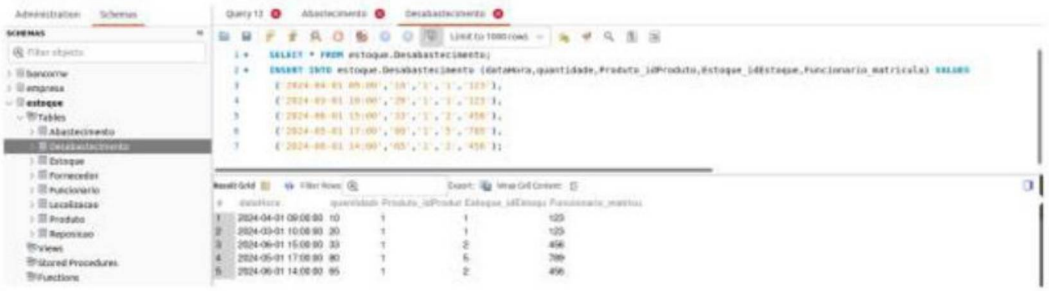
3.5) Tabela Fornecedor



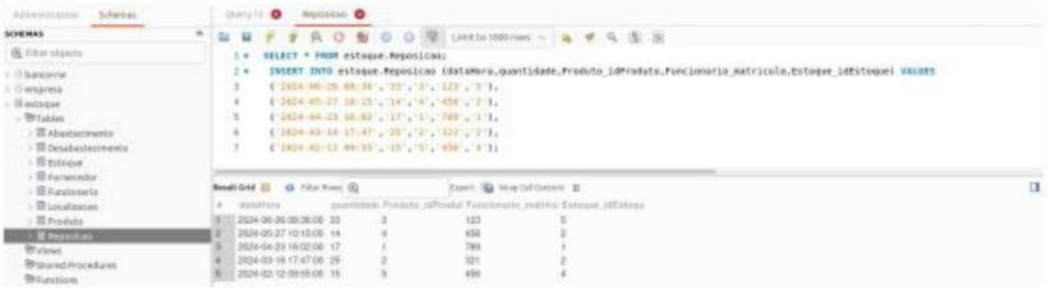
3.6) Tabela Abastecimento



3.7) Tabela Desabastecimento



3.8) Tabela Reposicao



FONTE: O autor (2025).

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como foco a aplicação prática de princípios de *clean code*, a partir da refatoração de um algoritmo previamente disponibilizado no caso, o Bubble Sort. O exercício consistiu em identificar pontos de melhoria no código original e aplicar ao menos três alterações que seguissem os princípios de legibilidade, simplicidade e organização recomendados por práticas ágeis de programação. A atividade incentivou uma análise crítica do código, promovendo a escrita de soluções mais claras e eficientes, características essenciais para equipes ágeis que dependem de colaboração contínua e código sustentável.

Essa disciplina se integra diretamente com outras áreas do curso, como desenvolvimento web, mobile, banco de dados e testes automatizados, pois todas compartilham a necessidade de um código limpo, padronizado e de fácil manutenção. O entendimento profundo dos aspectos ágeis de programação facilita o trabalho colaborativo em projetos multidisciplinares, agiliza as entregas e reduz a incidência de erros, o que está alinhado com o conceito de *entregas incrementais e contínuas* defendido por metodologias como Scrum e XP (Extreme Programming).

Além disso, ao praticar refatorações simples mas significativas, o aluno compreende como pequenas melhorias no código podem gerar grandes ganhos em produtividade, escalabilidade e qualidade do software. A disciplina, portanto, reforça a base técnica essencial para o desenvolvimento de soluções robustas e alinhadas às melhores práticas do desenvolvimento ágil de software.

7.1 ARTEFATOS DO PROJETO

FIGURA 29 – CÓDIGO EXEMPLO EM JAVA DO ALGORITMO BUBBLE SORT OTIMIZADO PARA REFATORAÇÃO

```java
// Implementação otimizada em Java do Bubble sort
// Código extraído de https://www.geeksforgeeks.org/bubble-sort/

//Emanuel Reis
// Modificações conforme explicado em sala

import java.io.*;

class BubbleSort {

    static void bubbleSort(int numeros[])
    {
        int tamanho = numeros.length;
        boolean foiTrocado;

        for (int i = 0; i <= tamanho; i++) {
            foiTrocado = false;

            for (int j = 0; j < (tamanho - 1) - i; j++) foiTrocado |= precisaTrocar(numeros, j);

            if (!foiTrocado) break;
        }
    }

    static boolean precisaTrocar(int numeros[], int j) {
        if (numeros[j] > numeros[j + 1]) {
            trocar(numeros, j);
            return true;
        }
        return false;
    }

    static void trocar(int numeros[], int j) {
        int temp = numeros[j];
        numeros[j] = numeros[j + 1];
        numeros[j + 1] = temp;
    }


    static void imprimirNumeros(int numeros[])
    {
        for(int numero : numeros)
            System.out.print(numero + " ");
        System.out.println();
    }

    public static void main(String args[])
    {
        int numeros[] = { 64, 34, 25, 12, 22, 90, 11 };
        bubbleSort(numeros);
        System.out.println("Array ordenado: ");
        imprimirNumeros(numeros);
    }
}

// This code is contributed
```

FONTE: O autor (2025).

8 DISCIPLINA: WEB1 E WEB2 – DESENVOLVIMENTO WEB 1 E 2

Nas disciplinas WEB1 e WEB2 foram desenvolvidas habilidades práticas voltadas à construção de aplicações web completas, com foco na implementação de operações CRUD (Create, Read, Update e Delete). O objetivo principal foi capacitar os alunos na criação de sistemas com front-end em Angular e back-end em Java, utilizando o framework Spring Boot, com persistência de dados no banco PostgreSQL.

WEB_01

O primeiro projeto teve como escopo o desenvolvimento de dois CRUDs independentes: um para a entidade *Aluno* e outro para a entidade *Curso*. Os atributos definidos para cada entidade foram:

- **Aluno**:
  - id (numérico)
  - nome (string)
  - CPF (string)
  - e-mail (string)
  - data de nascimento (string)
- **Curso**:
  - id (numérico)
  - nome (string)
  - link (string)

As funcionalidades implementadas incluíram telas para listagem de registros, inserção de novos dados, edição de informações e exclusão de elementos com confirmação. Nessa fase, os dados foram armazenados no *Local Storage* do navegador. O uso de *Bootstrap* ou *Material Design* foi obrigatório para garantir responsividade e boa experiência visual. Não foram exigidas implementações de login, menus ou relacionamentos entre as entidades.

WEB_02

No segundo projeto, o sistema foi expandido para incluir três CRUDs: *Aluno*, *Curso* e *Matrícula*. Este último estabeleceu a relação entre alunos e cursos, contemplando os seguintes dados:

- **Matrícula**:
  - Aluno (referência à entidade Aluno)

- Curso (referência à entidade Curso)
- Data da matrícula (string)
- Nota (numérico)

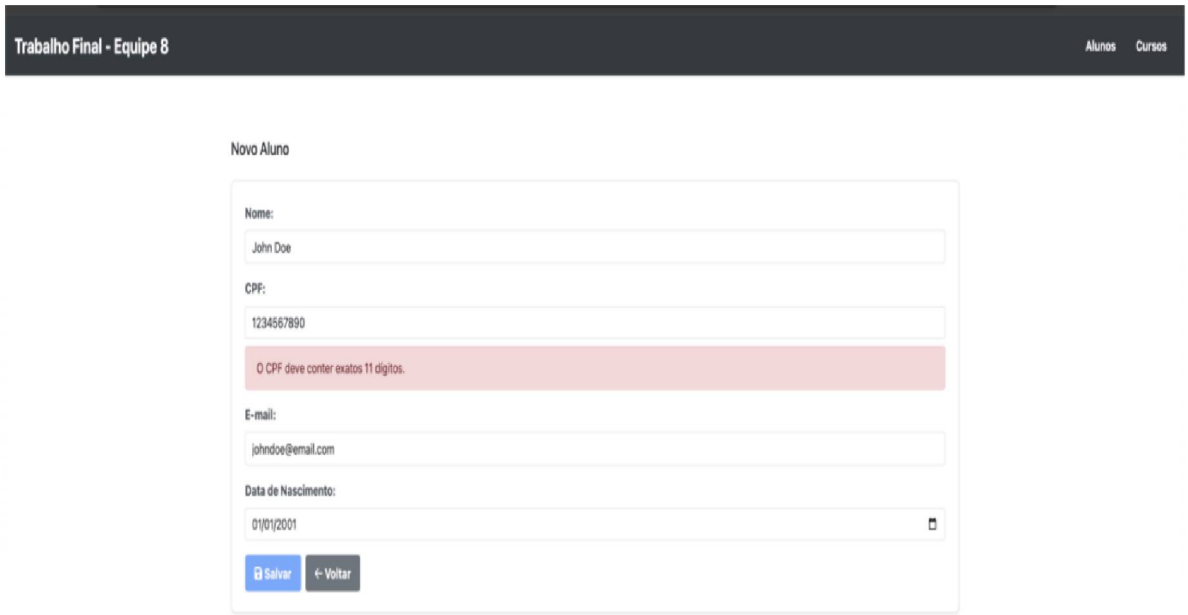
As telas de inserção e edição utilizaram *comboboxes* para seleção das entidades relacionadas. A aplicação passou a armazenar os dados em um banco de dados PostgreSQL, acessado via API REST implementada com Spring Boot. Além disso, o projeto exigiu a criação de um menu de navegação entre os CRUDs. Assim como no projeto anterior, não foi necessária a inclusão de autenticação ou permissões.

Essa atividade proporcionou uma experiência prática completa do ciclo de desenvolvimento full-stack, integrando conhecimentos de modelagem de dados, backend, frontend e usabilidade.

8.1 ARTEFATOS DO PROJETO

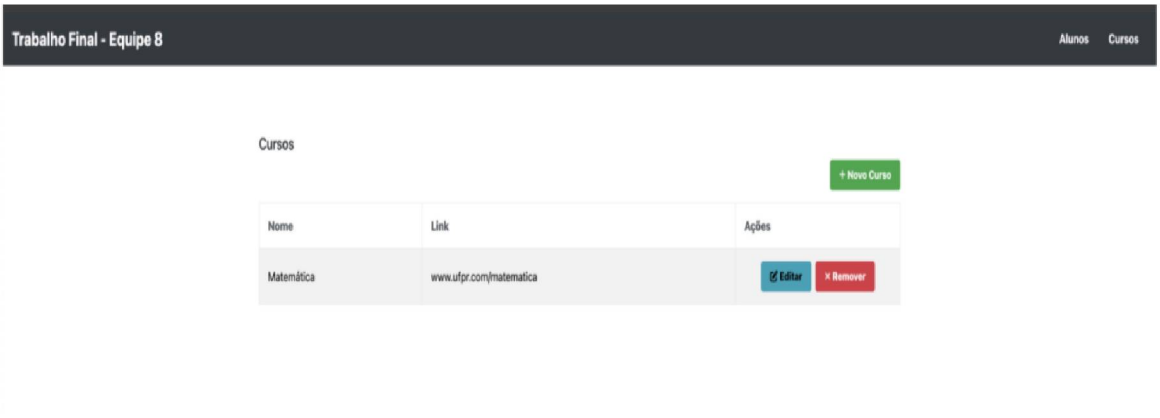
WEB_01: Aplicativo desenvolvido contém 2 CRUDs de entidade "Aluno"e "Curso".

FIGURA 30 – TELA NOVO ALUNO



FONTE: O autor (2025).

FIGURA 31 – TELA DO CURSO



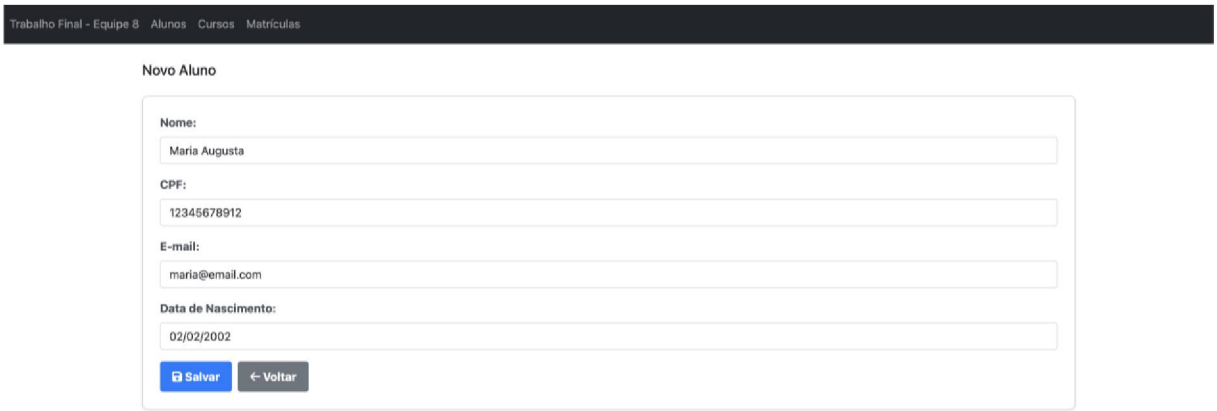
FONTE: O autor (2025).

Segue link de demonstração do trabalho:

https://ufprbr0my.sharepoint.com/:v:/g/personal/enzofurlan_ufpr_br/EUGLNq9Fi4hDrB8fEhBXZyEB OU_hZTzaZBXguMcOixvgLQ?e=6odbeh

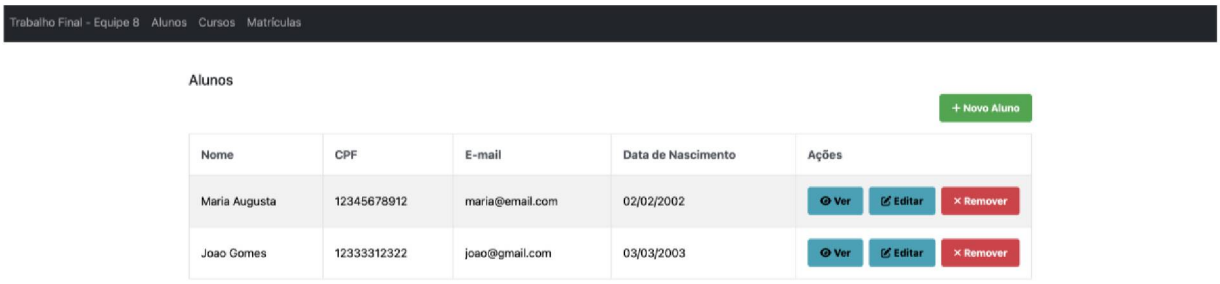
WEB_02: Aplicativo desenvolvido contém 3 CRUDs de entidade "Aluno", "Curso" e "Matricula".

FIGURA 32 – NOVO ALUNO (CADASTRO)



FONTE: O autor (2025).

FIGURA 33 – CADASTRO ALUNO

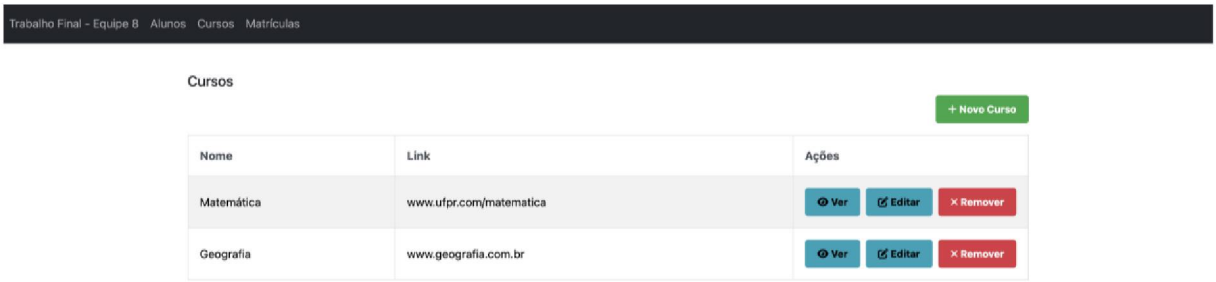


FONTE: O autor (2025).

FIGURA 34 – NOVO CURSO (CADASTRO)

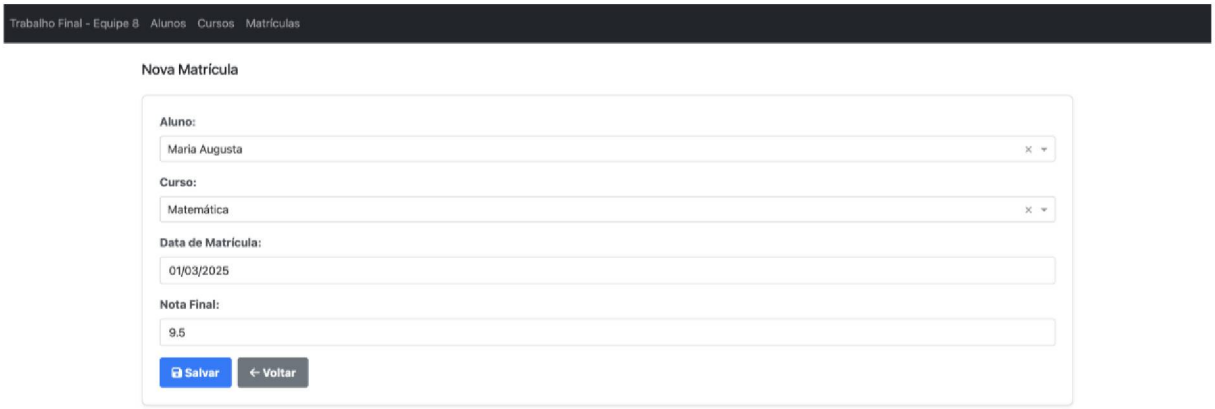
FONTE: O autor (2025).

FIGURA 35 – CURSOS CADASTRADOS



FONTE: O autor (2025).

FIGURA 36 - MATRÍCULA



FONTE: O autor (2025).

FIGURA 37 – MATRÍCULAS CADASTRADAS

FONTE: O autor (2025).

Segue link de demonstração do trabalho: https://ufprbr0my.sharepoint.com/:v:/g/personal/enzofurlan_ufpr_br/EeKjnjn1leNNntBlKhfhh9ABpe gUP4leRyZZz2TJKSjQtw

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

O desenvolvimento do trabalho seguiu as seguintes etapas:

1. **Descrição do produto**: apresentação detalhada do aplicativo ou site criado, incluindo suas funcionalidades, propósito e justificativa para o desenvolvimento.
2. **Criação de protótipos**: elaboração de pelo menos cinco telas prontas para representar a interface e a navegação do produto.
3. **Justificativa das escolhas de design**: explicação das decisões tomadas em relação a cores, layout, tipografia e demais elementos visuais, considerando a experiência do usuário.
4. **Validação com usuário**: apresentação das telas a, no mínimo, um usuário potencial, esclarecendo o funcionamento do produto, suas opções de interação e coleta de feedback sobre usabilidade e melhorias sugeridas.

9.1 ARTEFATOS DO PROJETO

Nome: AtendSmart

Função: O AtendSmart é uma plataforma digital (aplicativo e site) focada em facilitar o gerenciamento de serviços de saúde. Entre suas funcionalidades estão:

- Agendamento e reagendamento de consultas e exames;
- Sugestões personalizadas de profissionais com base no perfil do usuário;
- Triagem online e atendimento por videochamada;
- Fila online com tempo estimado de espera;
- Check-in digital via QR code;
- Localização de hospitais parceiros e próximos;
- Histórico de atendimentos e envio de feedback;
- Comunicação direta com atendentes;
- Digitalização de documentos médicos.

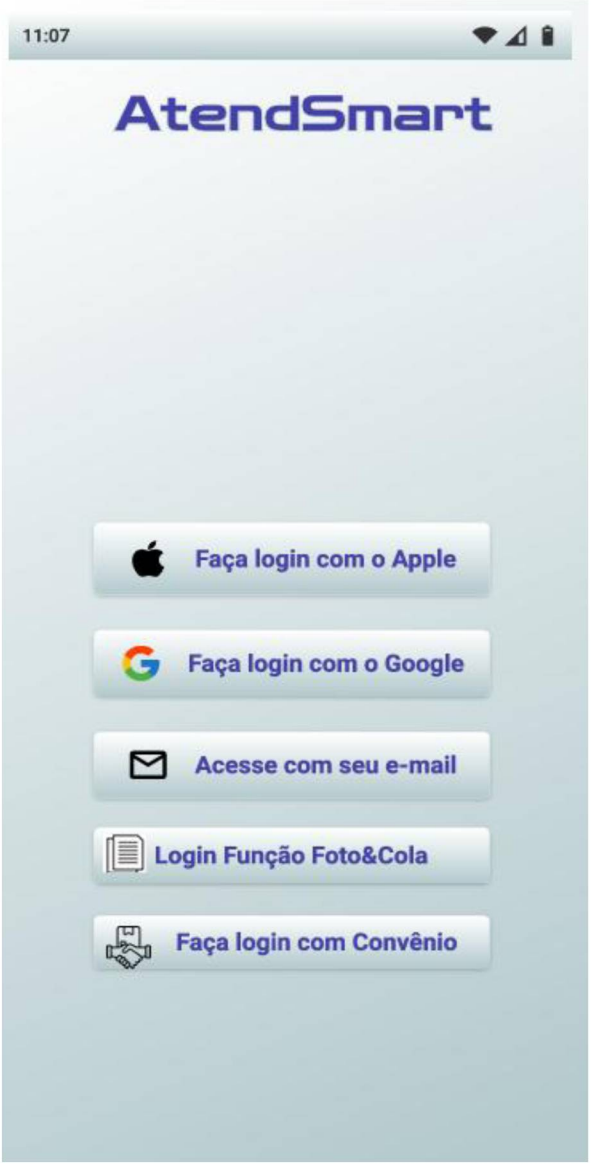
Justificativa: O app foi desenvolvido para responder à demanda crescente por soluções digitais na saúde, especialmente com o avanço da telemedicina e da necessidade de eficiência nos atendimentos médicos. A plataforma busca:

- Reduzir deslocamentos e melhorar o acesso aos serviços de saúde;
- Otimizar o tempo de pacientes e profissionais;
- Aumentar a satisfação dos usuários com atendimento personalizado e digital.

Objetivo: Modernizar a experiência do paciente no sistema de saúde por meio de uma plataforma integrada, ágil e acessível, promovendo:

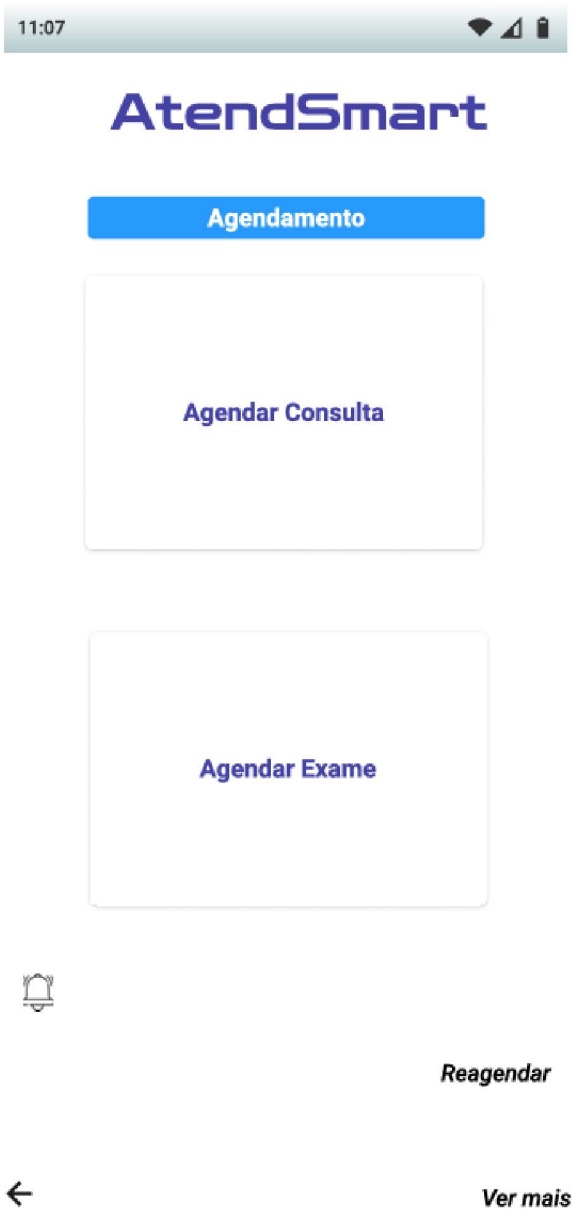
- Maior controle do usuário sobre sua jornada médica;
- Integração entre paciente, profissional e instituição de saúde;
- Redução de filas e melhora na gestão dos serviços.

FIGURA 38 – TELA INICIAL



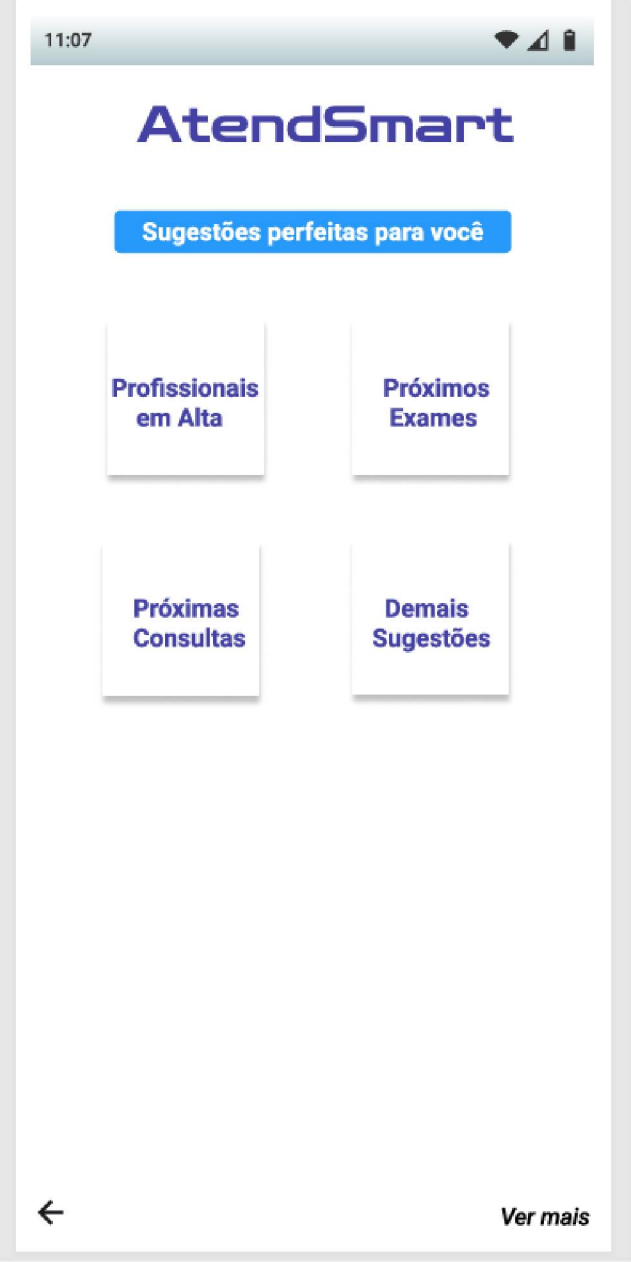
FONTE: O autor (2025).

FIGURA 39 – TELA AGENDAMENTO



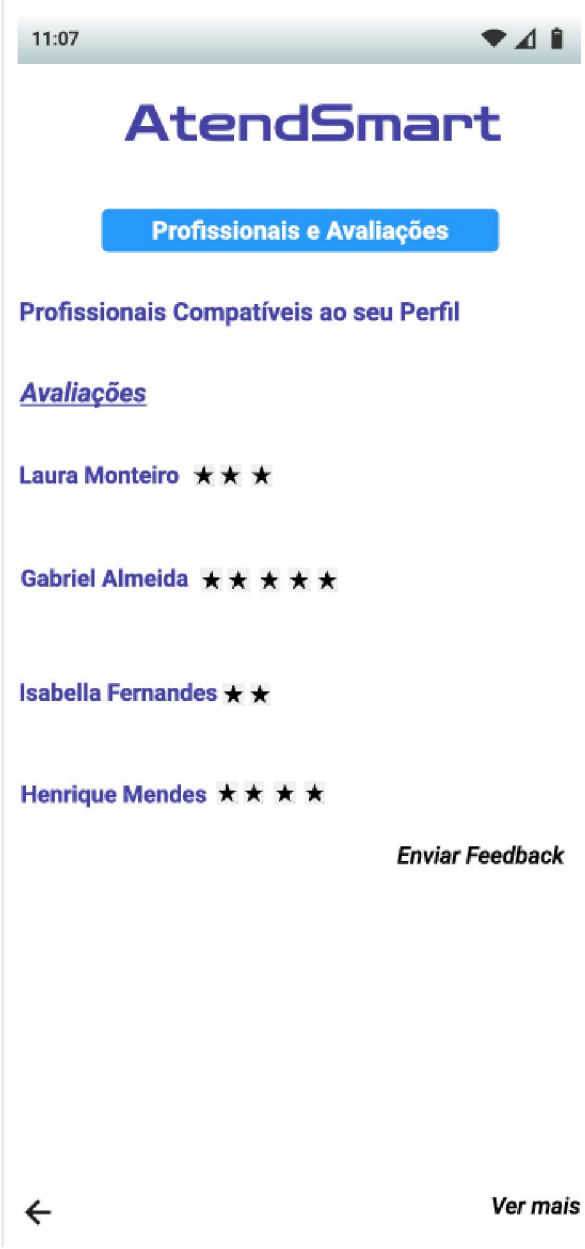
FONTE: O autor (2025).

FIGURA 40 – TELA SUGESTÕES PERFEITAS



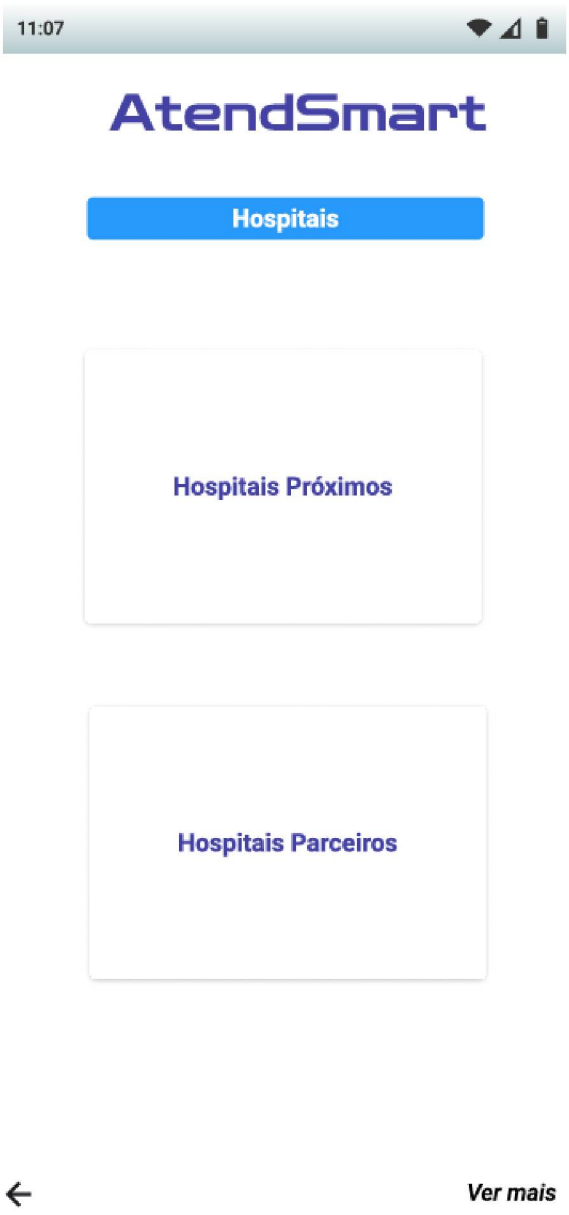
FONTE: O autor (2025).

FIGURA 41 - TELA DE AVALIAÇÕES



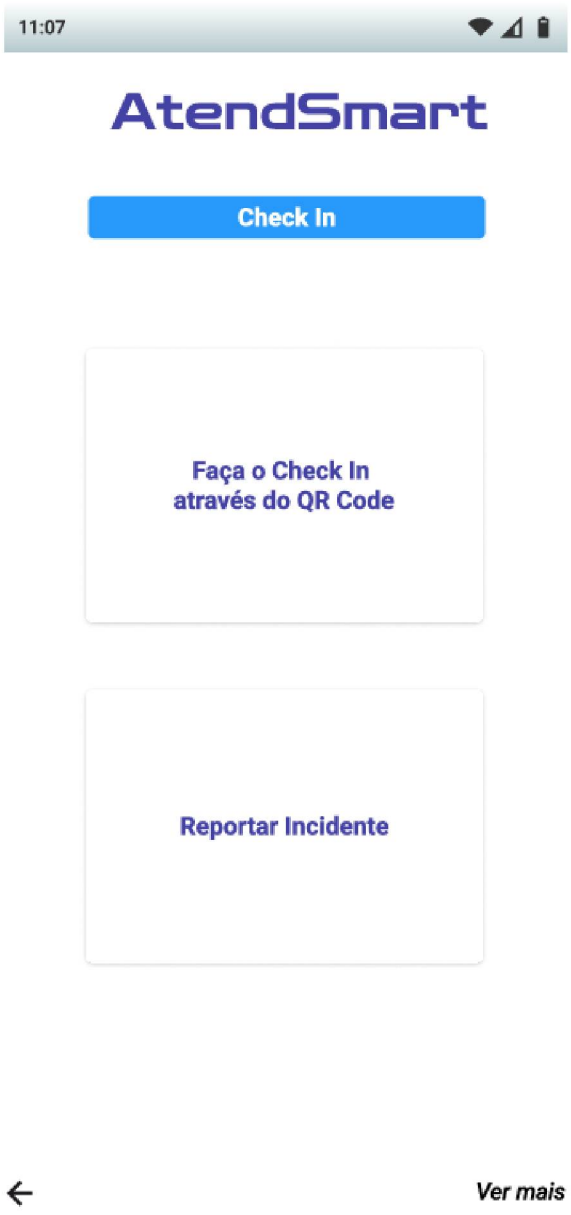
FONTE: O autor (2025).

FIGURA 42 - TELA HOSPITAIS



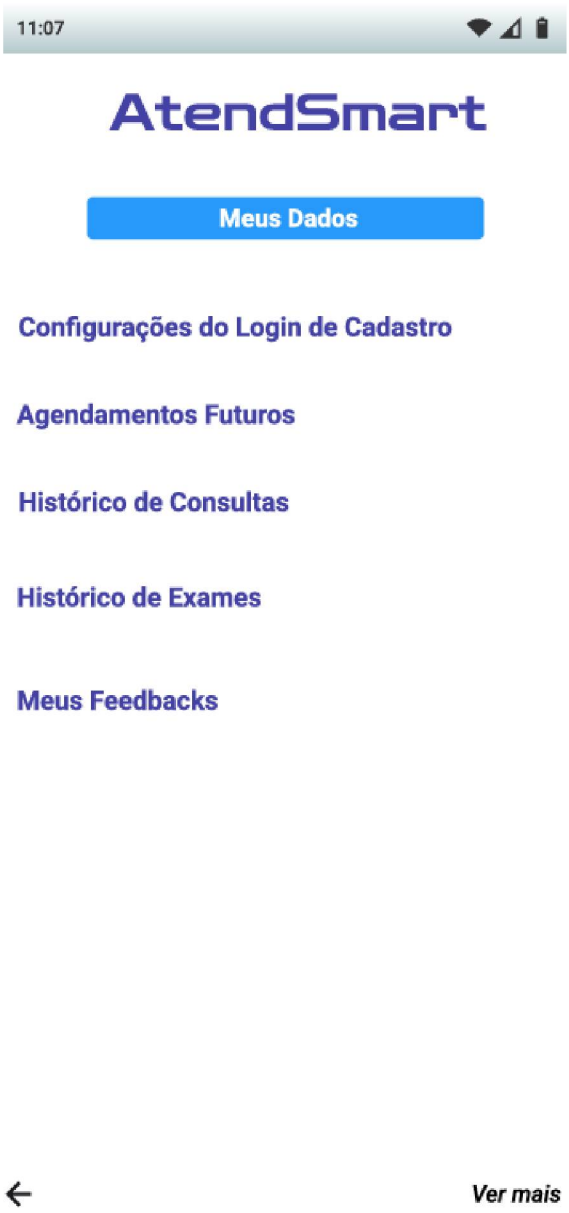
FONTE: O autor (2025).

FIGURA 43 - TELA CHECK IN



FONTE: O autor (2025).

FIGURA 44 – TELA MEUS DADOS

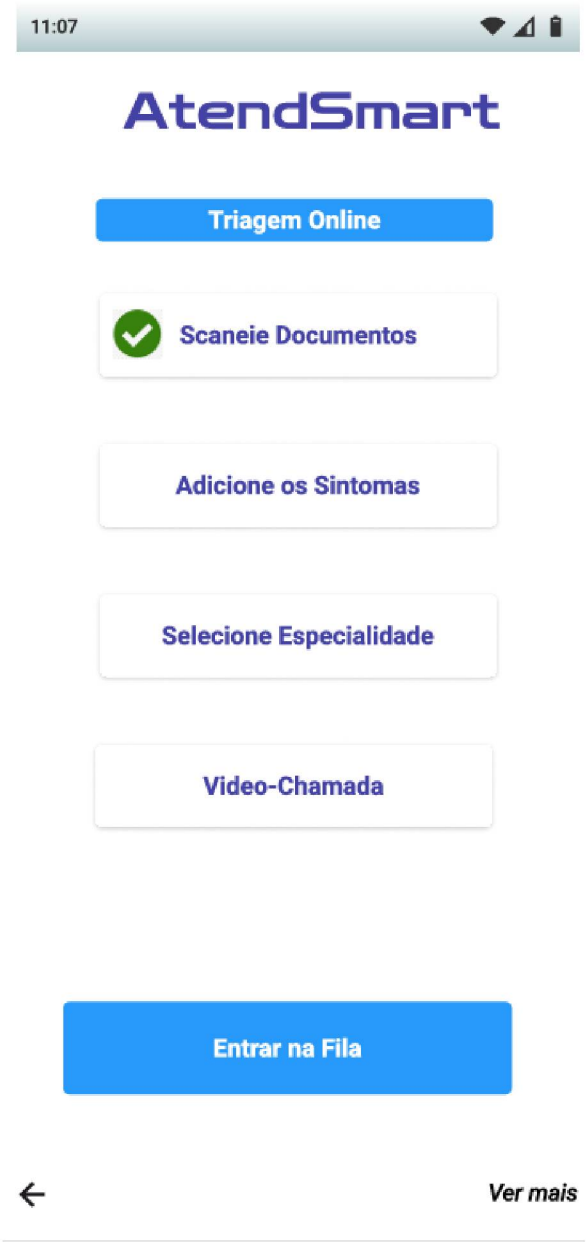


FONTE: O autor (2025).

FIGURA 45 – TELA TEMPO DE ESPERA (FILA)

FONTE: O autor (2025).

FIGURA 46 – TELA TRIAGEM ONLINE



FONTE: O autor (2025).

Cores:

● Escolha de cores claras: Aplicativos de saúde costumam utilizar cores claras (como branco e azul) para transmitir limpeza, confiança e segurança.

● Contraste nas seções de "Fila Online" e "Tempo Estimado": Elementos como o tempo de espera (20 min) destacam-se. Pode ser para chamar a atenção do usuário para informações importantes e urgentes.

Layout e Estrutura:

● Divisão clara por blocos: Cada funcionalidade tem sua seção dedicada, como "Agendar Consulta", "Meus Dados", "Hospitais Próximos" e "Profissionais em Alta".

● Agrupamento funcional: Por exemplo, os serviços de login aparecem juntos, sugerindo que a prioridade inicial é o acesso rápido. Em seguida, o foco muda para funcionalidades úteis como Agendamento.

● Uso do botão “Ver mais”: Ele indica a intenção de manter a interface principal limpa e simples, evitando sobrecarga de informações e fornecendo acesso gradual conforme necessário.

Fontes:

● Tipografia minimalista: Os exemplos apresentados mostram que a escolha foi por uma fonte sem serifa, comum em interfaces digitais por ser mais legível em telas. Fontes desse tipo transmitem modernidade e acessibilidade.

● Tamanhos diferenciados: Destaca-se a hierarquia visual, por exemplo, com títulos e informações críticas (como o tempo de espera) em tamanho maior.

Interatividade e Navegação:

● Ações destacadas como “Agendar” e “Check-in”: Essas funcionalidades são priorizadas para facilitar o uso imediato, sugerindo uma experiência voltada para a agilidade.

● QR Code para check-in: Essa escolha facilita o processo, alinhando-se com a tendência de digitalização e redução de papel em serviços de saúde.

Retorno do Usuário e Feedback:

Usuário: Aline Aires TeixeiraFeedback: A sétima tela deveria estar após a primeira tela.Retorno: A prioridade foi dada à tela de agendamento para reduzir a fila de espera.

10 DISCIPLINA: MOB1 E MOB2 – DESENVOLVIMENTO MOBILE 1 E 2

Escrever As disciplinas de Desenvolvimento Mobile 1 e 2 tiveram como objetivo capacitar os alunos na criação de aplicativos móveis nativos utilizando a linguagem Kotlin e a IDE Android Studio, com foco na aplicação prática de conceitos fundamentais do desenvolvimento ágil. Nos projetos desenvolvidos, os estudantes foram desafiados a criar aplicativos funcionais com diferentes níveis de complexidade, incluindo interações com APIs externas, armazenamento em memória e navegação estruturada entre telas.

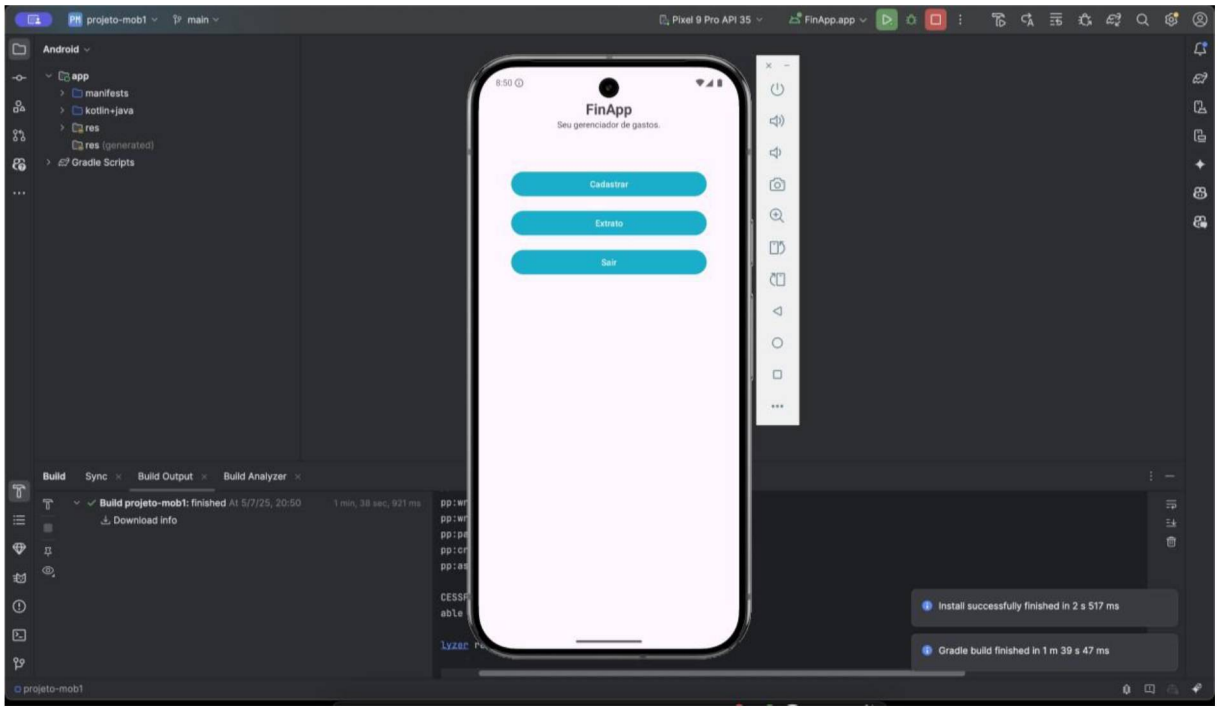
Na disciplina MOB1, foi desenvolvido o aplicativo *FinApp*, uma ferramenta de controle financeiro pessoal que permite o cadastro de transações de débito e crédito. O projeto teve como foco a implementação de funcionalidades básicas, utilizando estruturas de dados em memória e promovendo a experiência com a lógica de negócios e a interface do usuário. Já em MOB2, o desafio foi integrar o aplicativo a uma API externa a HP-AP para realizar consultas e exibir dados da franquia Harry Potter. Com isso, os alunos puderam compreender o consumo de web services e o uso de corrotinas para chamadas assíncronas, práticas essenciais no contexto de aplicações reais.

A importância dessas disciplinas no desenvolvimento ágil está diretamente ligada à entrega contínua e incremental de funcionalidades. Os ciclos de desenvolvimento curtos, o feedback constante e o aprendizado iterativo permitiram maior adaptação às mudanças, reforçando os valores do Manifesto Ágil. Além disso, a aplicação prática dos conhecimentos fortaleceu a integração com outras disciplinas do curso, como UX (experiência do usuário), BD (banco de dados), INFRA (DevOps) e TEST (testes automatizados). O desenvolvimento mobile ampliou a visão do aluno sobre a construção de soluções completas e modernas, promovendo habilidades essenciais para o mercado atual.

10.1 ARTEFATOS DO PROJETO

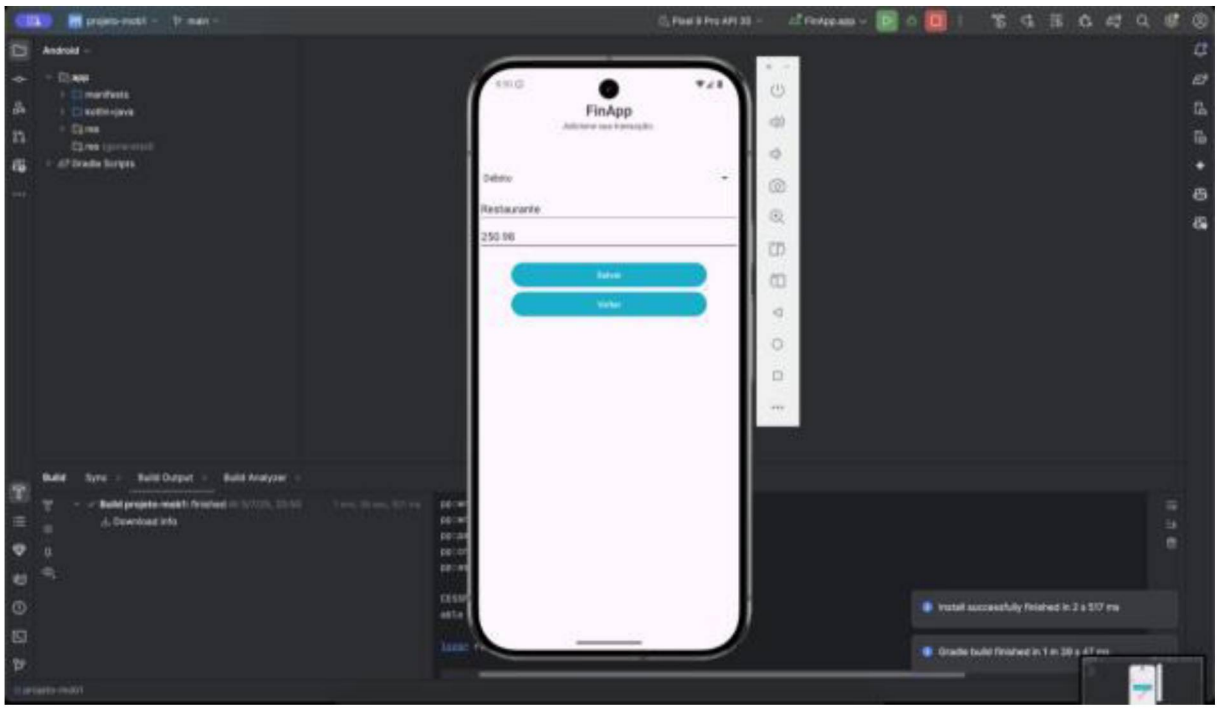
MOB_01: Aplicativo "FinApp" para cadastro de operações financeiras e extrato com a utilização de dados internos acesso a API externa.

FIGURA 47 – TELA CADASTRO



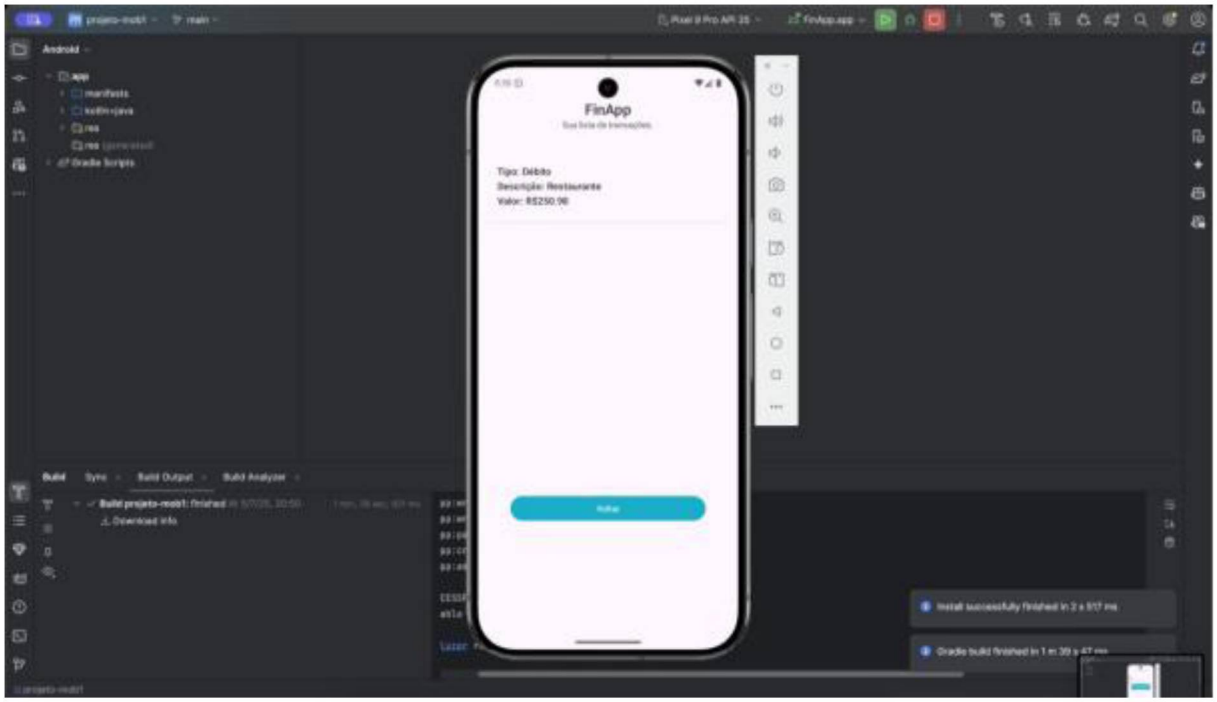
FONTE: O autor (2025).

FIGURA 48– TELA ADICIONAR TRANSAÇÃO



FONTE: O autor (2025).

FIGURA 49 – TELA LISTA DE TRANSAÇÕES

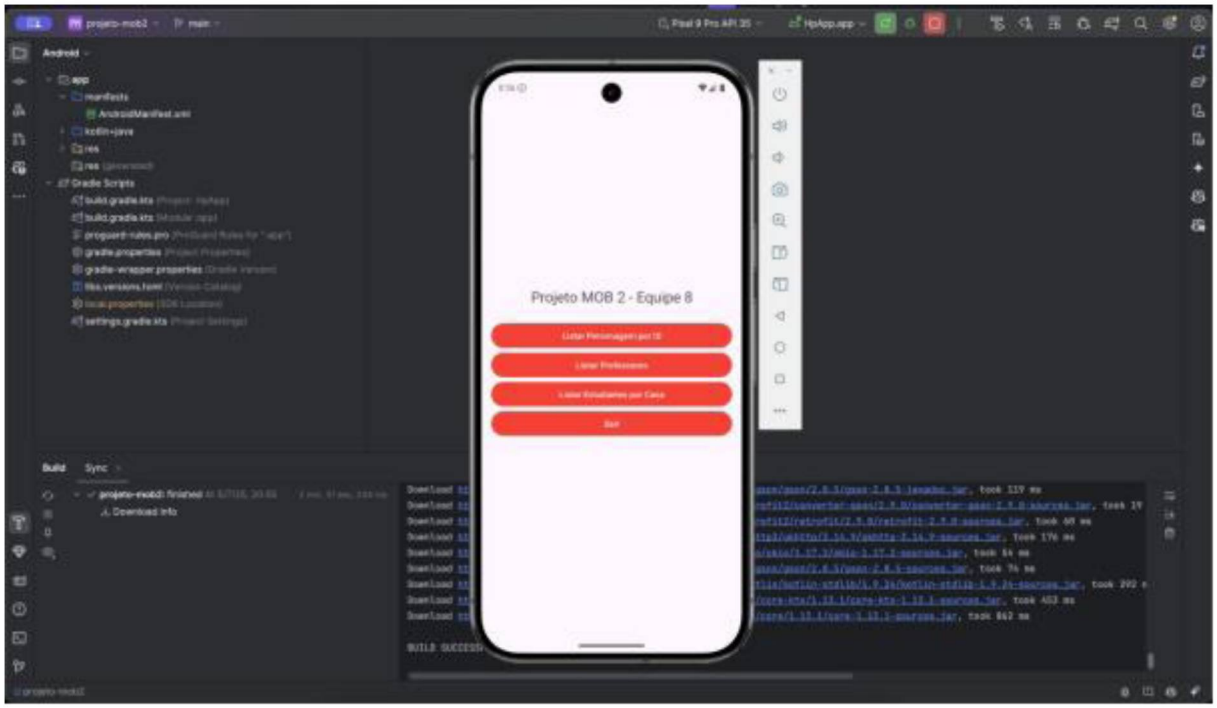


FONTE: O autor (2025).

Segue link do gitHub: https://github.com/enzofurlan/projeto-mob1

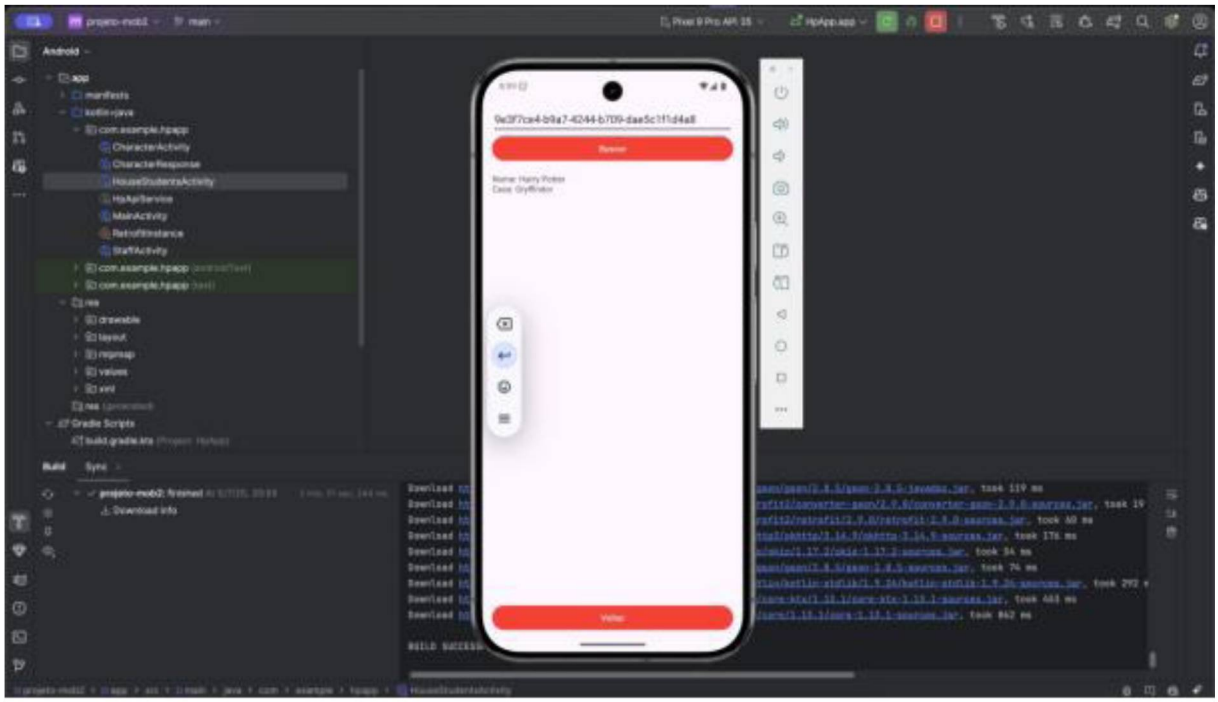
MOB_02: Aplicativo mobile em plataforma "Android" que acessa dados de API externa e imprime na tela os dados da Saga Harry Potter.

FIGURA 50 – TELA LISTAGEM DO PROJETO



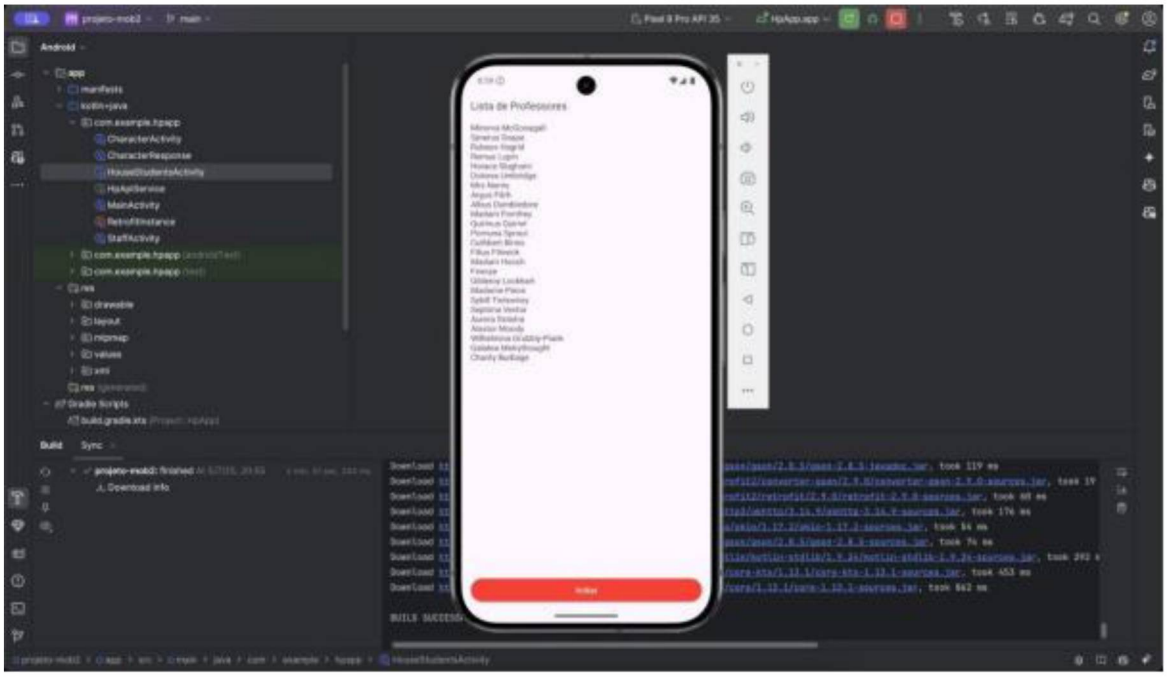
FONTE: O autor (2025).

FIGURA 51 – TELA BUSCAR



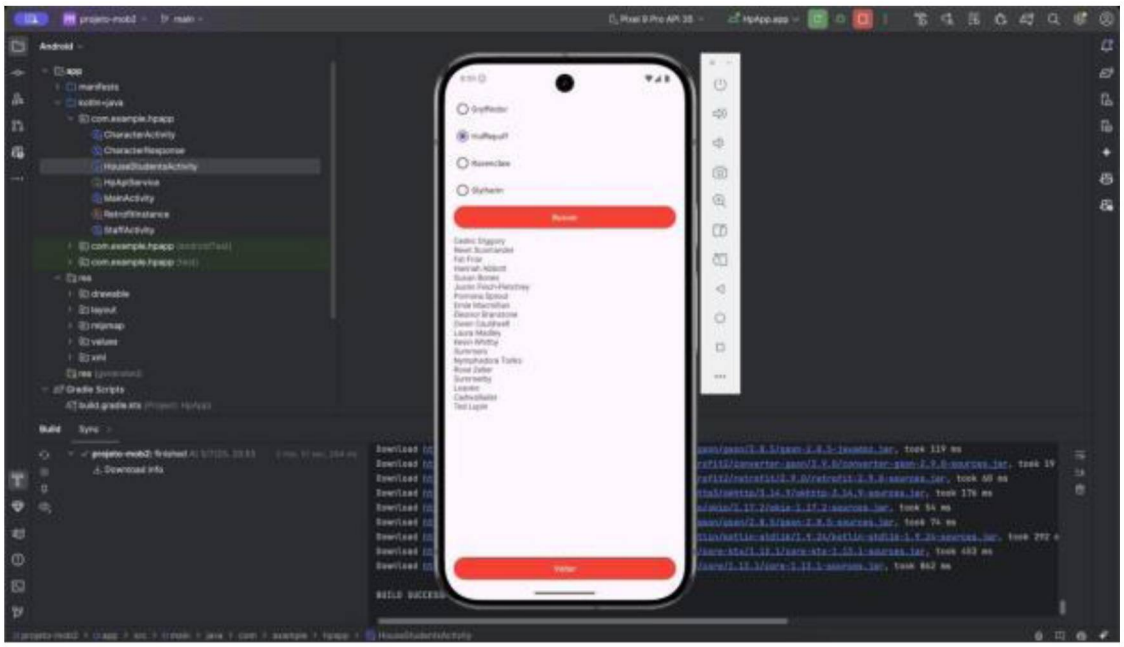
FONTE: O autor (2025).

FIGURA 52 – TELA LISTA DE PROFESSORES



FONTE: O autor (2025).

FIGURA 53 – TELA DE BUSCA (CATEGORIA)



FONTE: O autor (2025).

Segue link do gitHub: https://github.com/enzofurlan/projeto-mob2

11 DISCIPLINA: INFRA - INFRAESTRUTURA PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SOFTWARE (DEVOPS)

A disciplina de Infraestrutura para Desenvolvimento e Implantação de Software (DevOps) teve como foco principal apresentar aos alunos os conceitos e ferramentas essenciais para automação, integração contínua e entrega contínua de aplicações. O projeto proposto consistiu na utilização prática do **Docker** e do **GitLab**, ferramentas amplamente utilizadas no mercado para orquestração de containers e gerenciamento de repositórios de código com pipelines automatizados.

Durante a atividade, os alunos executaram a configuração de um ambiente virtualizado por meio do Docker, utilizando uma imagem específica do GitLab Jenkins. Essa configuração incluiu a publicação de portas estratégicas (22, 80, 443 e 9091), login no sistema como usuário root, acesso ao arquivo de senha no container e a realização de operações de versionamento com commit e push no repositório Git. Essa experiência permitiu a compreensão prática de ambientes de homologação e produção, reforçando a importância da infraestrutura como parte integrante do ciclo de vida de software.

No contexto do desenvolvimento ágil, a disciplina INFRA se mostrou fundamental para garantir entregas frequentes, seguras e escaláveis. A abordagem DevOps promove a colaboração entre os times de desenvolvimento e operações, reduz o tempo de implantação e aumenta a confiabilidade dos sistemas entregues. Essa prática se integra de forma direta com disciplinas como TEST (testes automatizados), WEB/MOB (desenvolvimento web e mobile), e BD (banco de dados), viabilizando uma infraestrutura que suporta a automação dos testes, a integração de APIs, e o gerenciamento eficiente de versões.

Neste estágio do curso, tornou-se essencial compreender a infraestrutura de um sistema operando em um ambiente real. Para isso, utilizamos o Docker, que representou o uso de containers, e o GitLab, como exemplo de repositório virtual sincronizado com diretórios locais. O objetivo do trabalho foi fornecer um conjunto de instruções para baixar e executar uma imagem do GitLab Jenkins, bem como realizar a modificação de um arquivo em um ambiente de homologação.
Instruções:

1. Baixar e executar a imagem dfwandarti/gitlab_jenkins:3 no Docker:1.1. O container deve ser nomeado com sua matrícula.1.2. Certifique-se de publicar as portas 22, 80, 443 e 9091.
2. Acessar o GitLab:
   2.1. O nome de usuário é root.
   2.2. A senha pode ser encontrada dentro do container, no arquivo:
      1. bash
      1. CopiarEditar
      1. /etc/gitlab/initial_root_password
3. Modificação no projeto:

- Faça uma alteração no projeto que está dentro do Git.
- Em seguida, realize um commit e um push para enviar as mudanças ao repositório.

11.1 ARTEFATOS DO PROJETO

FIGURA 54 – TELA DOCKER

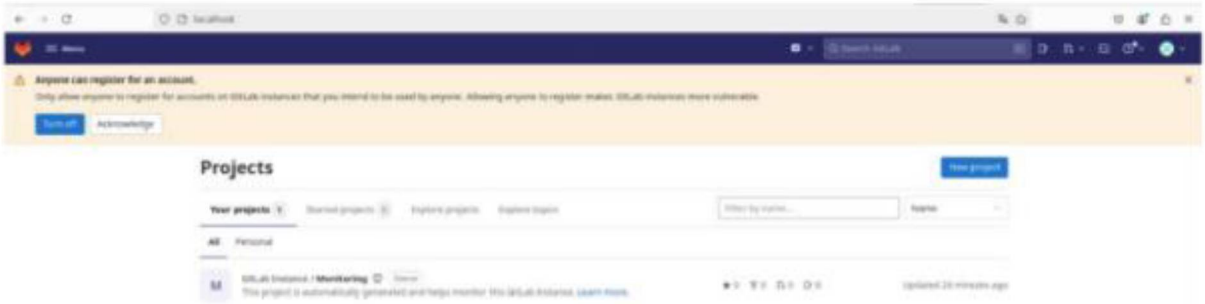
1. Baixe e rode no docker a imagem dfwandarti/gitlab_jenkins:3
   a. O container deve ter o mesmo nome que a matrícula.
   b. Publique as portas 22, 80, 443 e 9091.
   c. Tire um print screen do comando e do container rodando.

2. Logue no gitlab.
   a. O usuário é root.
   b. A senha está dentro do container no arquivo /etc/gitlab/initial_root_password

FONTE: O autor (2025).

FIGURA 55 – TELA LOG DO GIT

c. Tire um print screen do browser.



3. Faça um commit e push no projeto que está dentro do git.
   a. Tire um print screen do log do git com o commit.

```
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~$ git clone http://localhost/gitlab-instance-5f1e789c/Monitoring.git
Cloning into 'Monitoring'...
Username for 'http://localhost': root
Password for 'http://root@localhost':
warning: You appear to have cloned an empty repository.
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~$ cd Monitoring
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$ git switch -c main
Switched to a new branch 'main'
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$ touch README.md
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$ git add README.md
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$ git commit -m "add README"
[main (root-commit) f1b24de] add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```

```
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$ git push -u origin main
Username for 'http://localhost': root
Password for 'http://root@localhost':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 217 bytes | 217.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To http://localhost/gitlab-instance-5f1e789c/Monitoring.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$ git log
commit f1b24de714e341f2ec1a5cd15c8a1934f0e83fe8 (HEAD -> main, origin/main)
Author: emanuel-nunes <enreis.18@gmail.com>
Date:   Mon Jan 27 10:51:13 2025 -0300

    add README
emanuel-nunes@emanuel-nunes-OptiPlex-7050:~/Monitoring$
```

FONTE: O autor (2025).

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados teve como principal objetivo apresentar aos alunos os fundamentos da qualidade de software, com ênfase na importância da testabilidade em ambientes de desenvolvimento ágil. Foram explorados diversos frameworks e ferramentas voltadas para automação de testes, como o JUnit, utilizado para testes unitários em Java, e o Playwright, uma biblioteca de código aberto que permite automatizar testes em navegadores web de forma eficiente e moderna.

A atividade prática da disciplina consistiu na criação de um script automatizado capaz de abrir o Bloco de Notas (Notepad), inserir o título *"Entrega trabalho TEST DAS 2024"* e registrar o nome completo do aluno juntamente com sua matrícula no corpo da nota. Esse exercício teve como finalidade consolidar os conceitos de automação e demonstrar, de forma simples e prática, como um teste pode simular ações do usuário e verificar se uma funcionalidade está sendo executada conforme o esperado.

No contexto do desenvolvimento ágil de software, os testes automatizados são fundamentais para garantir a entrega contínua e a confiabilidade das soluções. Eles permitem detectar falhas rapidamente, facilitam a refatoração de código e asseguram que as alterações realizadas não comprometam funcionalidades já existentes. A disciplina se integra diretamente com outras áreas do curso, como Desenvolvimento Web (WEB), Mobile (MOB) e DevOps (INFRA), promovendo a cultura de testes desde o início do ciclo de desenvolvimento até o momento da entrega e implantação.

Assim, a unidade de Testes Automatizados reforça o compromisso com a qualidade, agilidade e segurança nas entregas, pilares essenciais no cenário atual de desenvolvimento de software ágil.

12.1 ARTEFATOS DO PROJETO

FIGURA 56 – TELA DO TESTE

```
import { test } from '@playwright/test';


test('test', async ({ page }) => {
  await page.goto('https://pt.anotepad.com/', {waitUntil: 'domcontentloaded'});
  await page.getByRole('textbox', { name: 'Título da Nota' }).fill('Entrega trabalho TEST
DAS 2024');
  await page.getByRole('textbox', { name: 'Conteúdo da Nota' }).fill('Henrique Antonio
Merlin Junior - 202400184875\nEmanuel Nunes Reis - 202400184596\nJoão Lucas
Della Coletta - 202400184580\nLuis Felipe Ortega Lyng - 202100146362\nRahuana
Ribeiro Fujioka - 202400184573\n');
  await page.getByRole('button', { name: 'Salvar' }).click();
});
```

FONTE: O autor (2025).

13 CONCLUSÃO

Este memorial sintetizou a trajetória formativa no curso de Especialização em Desenvolvimento Ágil de Software da UFPR, destacando a integração entre teoria e prática por meio do desenvolvimento de projetos que cobriram todas as etapas do ciclo full-stack. As disciplinas foram estrategicamente encadeadas, proporcionando uma formação progressiva e consistente desde os fundamentos dos métodos ágeis até a entrega de sistemas funcionais com testes automatizados e integração contínua.

A estruturação dos artefatos, como histórias de usuário, diagramas UML, protótipos de telas, aplicações web/mobile, pipelines de DevOps e scripts de testes, evidencia a aplicação prática dos princípios do Manifesto Ágil. A formação foi pautada na entrega incremental de valor, na colaboração contínua entre áreas e na adaptação constante às mudanças, características centrais da cultura ágil.

Entre os principais desafios enfrentados ao longo dessa jornada, destacam-se a mudança de mentalidade necessária para abandonar abordagens tradicionais em favor da agilidade, a conciliação entre modelagem leve e entrega rápida, e a adoção de ferramentas modernas (como Docker, GitLab, Spring Boot, Angular e Playwright) em um curto espaço de tempo. Superar essas barreiras exigiu disciplina, autonomia e forte comprometimento com a melhoria contínua.

Ao final do curso, o resultado é a consolidação de uma visão sistêmica sobre o desenvolvimento de software, capacitando o profissional a atuar em todas as camadas da concepção à implantação com foco em qualidade, usabilidade e valor para o usuário.

14 REFERÊNCIAS

AMBLER, S. W. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. Wiley, 2002.

BECK, K. et al. Manifesto for Agile Software Development. 2001. Disponível em: https://agilemanifesto.org/

BROWN, T. Design Thinking: Uma metodologia poderosa para decretar o fim das velhas ideias. Rio de Janeiro: Elsevier, 2009.

FOWLER, M. Refactoring: Improving the Design of Existing Code. 2. ed. Boston: Addison-Wesley, 2018.

HUMBLE, J.; FARLEY, D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston: Addison-Wesley, 2010.

LARMAN, C. Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos e ao Desenvolvimento Iterativo. 3. ed. Porto Alegre: Bookman, 2007.

PRESSMAN, R. S.; MAXIM, B. R. Engenharia de Software: Uma Abordagem Profissional. 9. ed. Porto Alegre: AMGH, 2016.

SHORE, J.; WARDEN, S. The Art of Agile Development. 2. ed. Sebastopol: O'Reilly Media, 2021.

SILVA, E. F. Testes de Software: Fundamentos e Aplicações. São Paulo: Novatec, 2014.

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