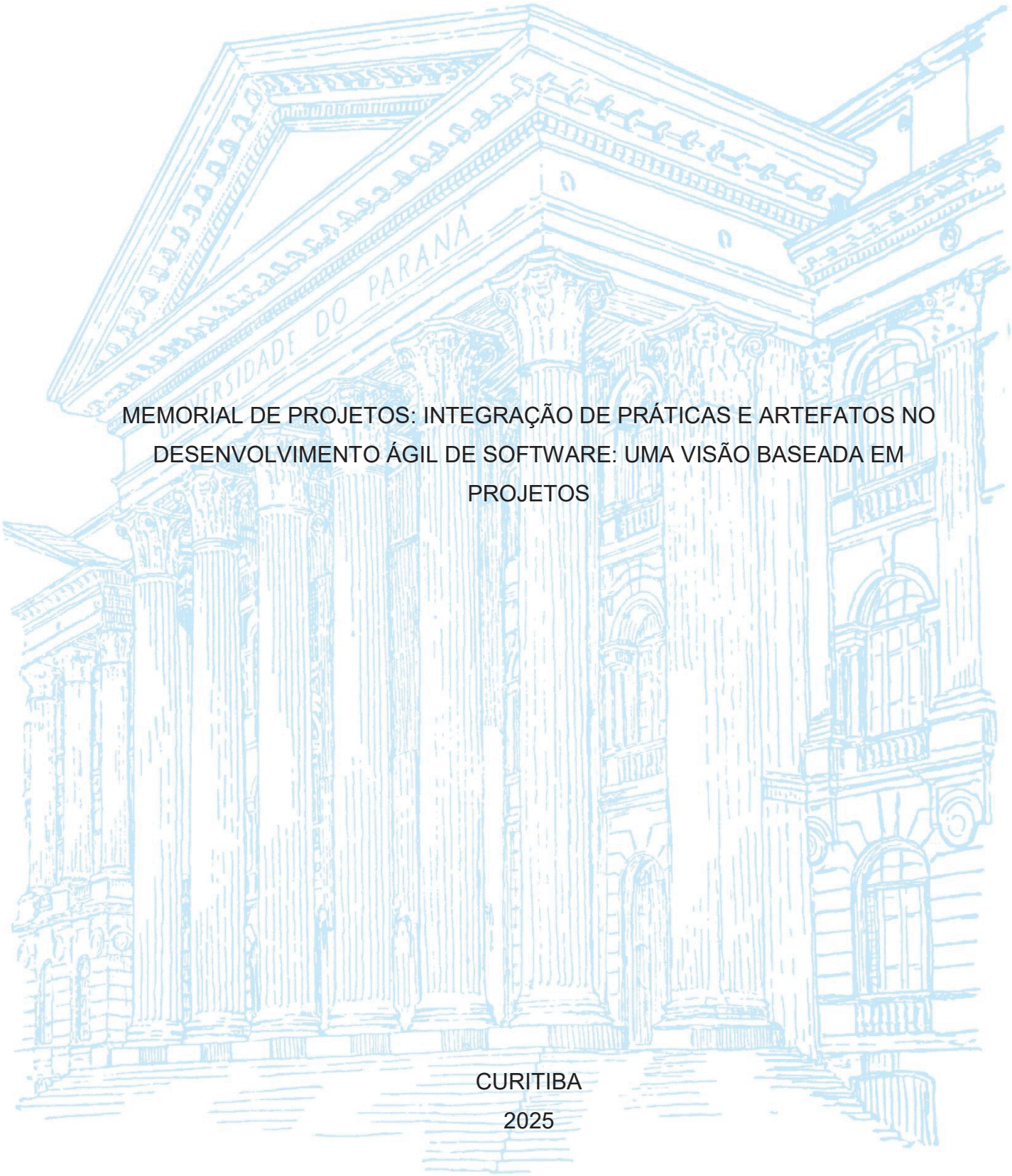
UNIVERSIDADE FEDERAL DO PARANÁ

RAFAEL PEREIRA DE AGUILAR



MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS E ARTEFATOS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA VISÃO BASEADA EM PROJETOS

CURITIBA

2025

RAFAEL PEREIRA DE AGUILAR

MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS E ARTEFATOS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA VISÃO BASEADA EM PROJETOS

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. Razer Anthom Nizer Rojas Montaño

CURITIBA

2025

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO DESENVOLVIMENTO ÁGIL DE SOFTWARE - 40001016398E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Desenvolvimento Ágil de Software da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **RAFAEL PEREIRA DE AGUILAR**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS E ARTEFATOS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA VISÃO BASEADA EM PROJETOS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 22 de Julho de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Rua Doutor Alcides Vieira Arco-verde - Curitiba - Paraná - Brasil
CEP 81520-260 - Tel: (41) 98874-0472 - E-mail: jaimewo@ufpr.br

RESUMO

Este memorial tem como objetivo apresentar, de forma integrada, os principais projetos e artefatos desenvolvidos ao longo do curso de Especialização em Desenvolvimento Ágil de Software. A estrutura do documento contempla a trajetória completa de aprendizado, desde a modelagem de requisitos e análise de processos até a implementação, testes automatizados e práticas de DevOps. Os artefatos apresentados incluem mapas mentais, diagramas UML, histórias de usuário, scripts SQL, protótipos de interfaces, códigos em Java, Angular e TypeScript, além de scripts de automação e configurações com Docker. Cada disciplina contribuiu com conhecimentos específicos que, quando reunidos, reforçam a prática ágil e iterativa no desenvolvimento de software. A combinação entre teoria e prática evidenciou o valor da colaboração, da entrega contínua e da adaptação às mudanças. O uso de ferramentas modernas e abordagens como TDD, Clean Code, CI/CD e prototipação rápida garantiu a aplicabilidade dos conceitos em contextos reais, alinhando o aprendizado às demandas do mercado atual.

Palavras-chave*:* Desenvolvimento Ágil, Modelagem, Testes Automatizados, DevOps.

ABSTRACT

This portfolio aims to present, in an integrated manner, the main projects and artifacts developed throughout the Postgraduate Program in Agile Software Development. The document outlines the complete learning journey, from requirements modeling and process analysis to implementation, automated testing, and DevOps practices. The artifacts include mind maps, UML diagrams, user stories, SQL scripts, interface prototypes, source code in Java, Angular, and TypeScript, as well as automation scripts and Docker configurations. Each subject contributed specific knowledge that, when combined, reinforced the iterative and agile software development process. The integration between theory and practice highlighted the value of collaboration, continuous delivery, and adaptability to change. The use of modern tools and approaches such as TDD, Clean Code, CI/CD, and rapid prototyping ensured the practical applicability of the concepts, aligning the learning process with current industry demands.

Keywords: Agile Development, Modeling, Automated Testing, DevOps.

SUMÁRIO

1 PARECER TÉCNICO

O presente parecer técnico tem como objetivo apresentar uma análise integrada dos projetos desenvolvidos ao longo das disciplinas do curso de Especialização em Desenvolvimento Ágil de Software da UFPR – Turma 2024. A partir da consolidação desses trabalhos, busca-se evidenciar como os conteúdos abordados se complementam e contribuem para a formação de uma visão prática e estratégica sobre o ciclo de desenvolvimento ágil, desde a concepção até a entrega contínua de valor.

As disciplinas iniciais forneceram uma base sólida em modelagem e planejamento de sistemas, com destaque para Métodos Ágeis para Desenvolvimento de Software (MADS) e Modelagem Ágil de Software (MAG1/MAG2). Nessas etapas, foram produzidos artefatos como mapas mentais, diagramas UML, histórias de usuário e modelos de banco de dados, os quais estruturaram a compreensão do escopo e do comportamento dos sistemas. A clareza na definição dos requisitos e o uso de linguagem visual facilitaram a comunicação entre as partes interessadas, alinhando-se aos princípios do Manifesto Ágil (Beck *et al.*, 2001).

No decorrer do curso, disciplinas como Introdução à Programação (INTRO), Desenvolvimento Web 1 e 2 (WEB1/WEB2), Banco de Dados (BD) e Desenvolvimento Mobile 1 e 2 (MOB1/MOB2) proporcionaram a implementação prática dos projetos utilizando tecnologias modernas como Java (Oracle, 2025), Angular (Angular Team, 2025), Spring Boot (Pivotal Software, 2025), PostgreSQL (PostgreSQL Global Development Group, 2025) e SQLite (Hipp *et al*., 2025). Foram criados sistemas completos com funcionalidades de CRUD (*Create, Read, Update, Delete*), organização de dados, integração entre camadas e uso de *frameworks*. Mesmo nas situações em que a implementação não foi realizada, o entendimento arquitetural e conceitual permitiu a simulação de soluções escaláveis e aderentes ao modelo ágil.

A disciplina de UX (*User Experience*) demonstrou como o *design* centrado no usuário influencia diretamente a aceitação e eficiência das soluções desenvolvidas. A criação de protótipos criados com Figma (Figma, 2025), aliados ao uso de cores, gráficos e hierarquia visual, reforçou a importância do *feedback* contínuo no processo iterativo de construção de interfaces. Essa etapa integra-se com as disciplinas técnicas ao destacar que a entrega de software vai além da funcionalidade: envolve também a experiência do usuário final (Knapp, 2016).

Já Aspectos Ágeis de Programação (AAP) e Testes Automatizados (TEST) aprofundaram as boas práticas de programação e a importância da qualidade contínua. O uso de princípios de Clean Code (Martin, 2008), testes automatizados com TypeScript e Selenium, além da adoção do TDD (*Test Driven Development*) (Beck, 2002), fortaleceram a ideia de que o código limpo, modular e bem testado facilita a manutenção e reduz falhas em ambientes de constante mudança. Essas abordagens contribuem diretamente para a agilidade e a confiabilidade do software entregue.

A disciplina de Infraestrutura para Desenvolvimento e Implantação de Software (DevOps) (INFRA) complementou o ciclo com práticas de automação e infraestrutura como código, por meio da utilização de Docker (Docker, 2025), Git (Git, 2025) e simulações com GitLab (GitLab, 2025). A experiência prática com containers, versionamento e integração contínua reforçou a importância da automação e da colaboração entre times de desenvolvimento e operação, pilares do DevOps e essenciais em ambientes ágeis (Shore; Warden, 2021).

Ao analisar os projetos desenvolvidos sob a ótica da integração, é possível perceber que o curso promoveu um ambiente de aprendizagem que simula o ciclo completo de desenvolvimento ágil. Os artefatos criados se conectam entre si, formando uma linha evolutiva que vai da modelagem à entrega, passando pela codificação, testes, prototipação e infraestrutura. O aprendizado não ocorreu de forma compartimentada, mas sim como um processo contínuo de iteração, validação e melhoria — exatamente como preconiza o desenvolvimento ágil moderno (Shore; Warden, 2021).

Em síntese, este memorial mostra que a aplicação de métodos ágeis vai além do uso de ferramentas: trata-se de uma mudança de mentalidade que privilegia a colaboração, a adaptabilidade, a entrega frequente de valor e a melhoria contínua. Os projetos desenvolvidos ao longo do curso, apresentados neste documento, refletem essa abordagem de forma prática e integrada, preparando o aluno para atuar em equipes multidisciplinares e ambientes altamente dinâmicos.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

Na disciplina de Métodos Ágeis para Desenvolvimento de Software, foram explorados os fundamentos que sustentam o desenvolvimento ágil, abordando desde os conceitos tradicionais de processos de software até os princípios e práticas que fundamentam as metodologias ágeis na atualidade. A disciplina iniciou com uma contextualização sobre engenharia de software, destacando sua definição, áreas de conhecimento e a importância dos processos no desenvolvimento de soluções robustas e eficientes.

Foram apresentados os modelos tradicionais de desenvolvimento, como o modelo linear sequencial (cascata), o modelo de prototipação, o incremental, o modelo espiral e o processo unificado (Sommerville, 2011). Essa abordagem permitiu compreender as limitações dos modelos tradicionais, especialmente em cenários com alta incerteza, mudanças constantes e necessidade de entregas rápidas, características típicas do atual mercado VUCA — volátil, incerto, complexo e ambíguo (Bennis; Nanus, 1985).

Nesse contexto, emergem os métodos ágeis como uma resposta às limitações dos modelos tradicionais. A disciplina aprofundou o entendimento sobre o Manifesto Ágil e seus quatro valores e doze princípios, que priorizam a colaboração, a entrega contínua de valor, a capacidade de adaptação às mudanças e a valorização das pessoas sobre processos rígidos (Beck *et al*., 2001).

Além disso, foram explorados os conceitos de *Lean Software Development*, abordando seus sete princípios fundamentais: eliminar desperdícios, construir qualidade, criar conhecimento, adiar decisões, entregar rapidamente, respeitar as pessoas e otimizar o todo (Poppendieck; Poppendieck, 2003). Esses princípios reforçam práticas que visam à eficiência, eficácia e sustentabilidade no desenvolvimento de software.

Também foram analisados modelos de maturidade, como o MPS.BR e o CMMI-DEV, que estruturam níveis de capacidade e desempenho das organizações, demonstrando que melhoria contínua e excelência nos processos são fundamentais, independentemente do modelo adotado (SOFTEX, 2021; CMMI Institute, 2018).

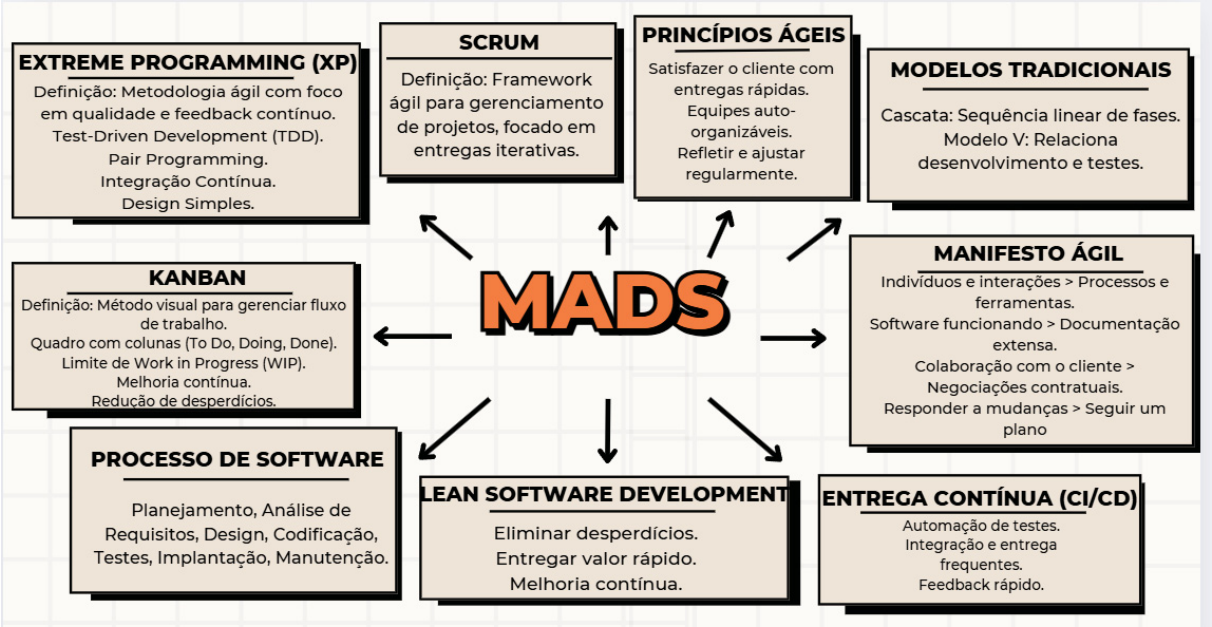
O projeto final dessa disciplina consistiu na elaboração de um mapa mental abrangendo todos os conceitos estudados, estruturando os principais tópicos em

níveis hierárquicos e permitindo uma visão clara das relações entre processos de software, princípios ágeis, métodos como Scrum (Schwaber; Sutherland, 2020), XP – Extreme Programming (Beck, 2004), Kanban (Anderson, 2010) e as práticas de entrega contínua.

A disciplina foi fundamental para estabelecer uma base teórica sólida sobre desenvolvimento ágil, permitindo compreender como os métodos, princípios e práticas se conectam e se aplicam ao desenvolvimento de software em ambientes de constante transformação. Esse conhecimento se torna um alicerce indispensável para as demais disciplinas do curso e para atuação no mercado, onde agilidade, qualidade e capacidade de adaptação são competências essenciais.

2.1 ARTEFATOS DO PROJETO

FIGURA 1 – PROCESSOS E PRINCÍPIOS DE DESENVOLVIMENTO DE SOFTWARE



FONTE: Autor (2025)

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

As disciplinas de Modelagem Ágil de Software 1 e 2 foram fundamentais para o entendimento e a prática da modelagem no desenvolvimento de software orientado a objetos, especialmente em ambientes ágeis. O objetivo principal foi proporcionar uma visão clara de como traduzir requisitos em artefatos visuais e textuais, capazes de guiar o desenvolvimento de sistemas de forma eficiente, colaborativa e alinhada aos princípios ágeis.

Durante a disciplina, foram desenvolvidos diversos artefatos que fazem parte do ciclo de desenvolvimento, tais como: Diagramas de Caso de Uso (nível 1 e nível 2), Histórias de Usuário, Diagramas de Classes, Diagramas de Sequência, além da definição de Requisitos Funcionais, Não Funcionais e Regras de Negócio.

A metodologia adotada enfatizou o uso de práticas como o Model Storm (Ambler, 2002), que consiste em reuniões rápidas entre desenvolvedores e *stakeholders* para rascunhar modelos que apoiam o entendimento do problema e das soluções, reforçando que a modelagem em métodos ágeis deve ser leve, colaborativa e focada na entrega de valor.

As Histórias de Usuário foram essenciais na especificação dos requisitos, permitindo transformar funcionalidades em pequenas entregas, priorizadas e bem compreendidas pela equipe. Essas histórias eram complementadas com critérios de aceitação, regras de negócio e observações técnicas, proporcionando clareza tanto para o desenvolvimento quanto para os testes.

O uso dos Diagramas UML (Booch; Rumbaugh; Jacobson, 2005) desempenhou papel estratégico na abstração e documentação dos sistemas. Os Diagramas de Classes permitiram visualizar a estrutura do sistema, seus objetos, atributos, métodos e os relacionamentos entre eles, sendo uma representação fundamental da arquitetura do software. Já os Diagramas de Sequência foram utilizados para descrever o fluxo de mensagens e interações entre os objetos, tornando mais fácil entender os comportamentos dos sistemas modelados.

Além disso, foram explorados outros diagramas complementares, como Diagramas de Atividades e Diagramas de Transição de Estados, especialmente úteis em processos ou fluxos de negócios mais complexos.

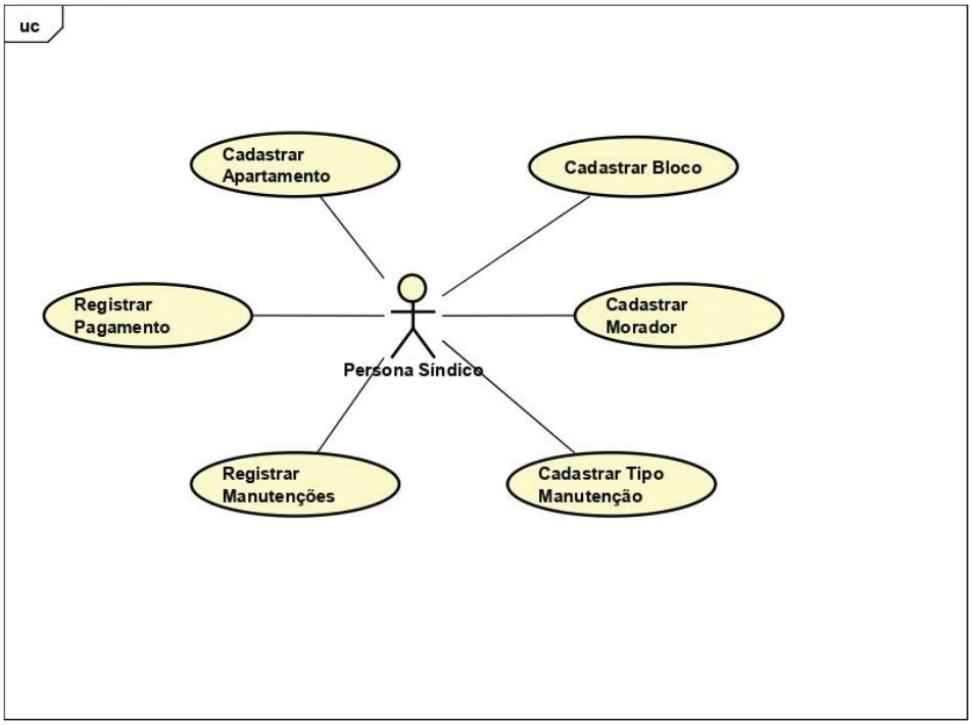
A integração das disciplinas MAG1 e MAG2 com o restante do curso se dá de forma direta, pois a modelagem é a ponte entre os requisitos e a codificação. Ter uma

modelagem bem feita, enxuta e assertiva reduz riscos, melhora a comunicação entre os membros da equipe e gera mais assertividade na implementação, mantendo a aderência aos princípios ágeis.

O conhecimento adquirido nessas disciplinas não só contribuiu para a construção dos projetos práticos do curso, como também é uma competência essencial para atuação no mercado, onde a capacidade de modelar rapidamente e de forma eficiente é um diferencial significativo.

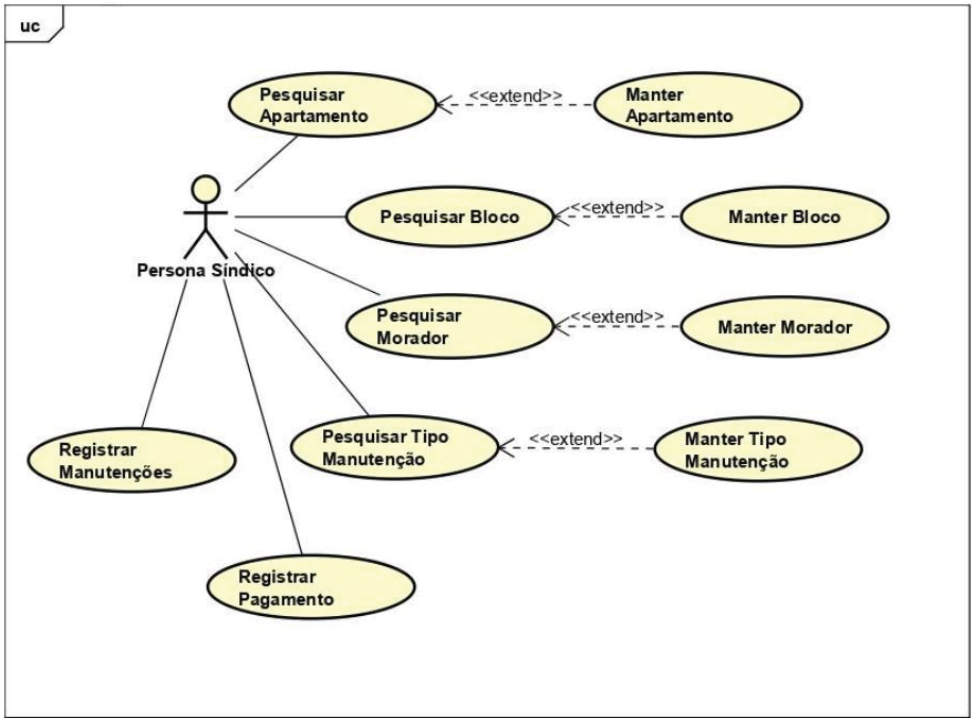
3.1 ARTEFATOS DO PROJETO

FIGURA 1: DIAGRAMA DE CASOS DE USO - NÍVEL 1



FONTE: Autor (2025)

FIGURA 1: DIAGRAMA DE CASOS DE USO - NÍVEL 2



FONTE: Autor (2025)

Histórias de usuários

HU001 – Pesquisar Apartamento

SENDO	Um Síndico
QUERO	Pesquisar apartamentos
PARA	Fazer manutenções nos seus dados

FIGURA 3: DESENHO DA TELA:

FONTE: Autor (2025)

Lista de critérios de aceitação:

1) Deve preencher a tabela com todos os apartamentos cadastrados.
2) Deve permitir incluir um novo apartamento.
3) Deve permitir alterar os dados de um apartamento cadastrado.
4) Deve permitir excluir um apartamento registrado.
5) Deve permitir consultar os dados do apartamento cadastrado.
6) Deve pesquisar os apartamentos na tabela exibida na tela.
7) Deve permitir acesso ao menu para selecionar outra tela.

Critérios de aceitação - detalhamento:

1) Deve preencher a tabela com todos os apartamentos cadastrados.

Dado que:	O síndico acessou a tela de Pesquisar Apartamento
Quando:	A tela é apresentada
Então:	A tabela da tela deve ser preenchida com todos os apartamentos cadastrados.

2) Deve permitir incluir um novo apartamento.

Dado que:	Os apartamentos estão na tabela
Quando:	O botão “Novo” é pressionado
Então:	O sistema chama a HU002 – Manter Apartamento passando o parâmetro “Novo”

3) Deve permitir alterar os dados de um apartamento cadastrado.

Dado que:	Os apartamentos estão na tabela
Quando:	O botão “Editar” é pressionado
Então:	O sistema chama a HU002 – Manter Apartamento passando o parâmetro “Alterar” e os dados do candidato da linha cujo botão foi pressionado.

4) Deve permitir excluir um apartamento registrado.

Dado que:	Os apartamentos estão na tabela
Quando:	O botão “Excluir” é pressionado em uma determinada linha
Então:	O sistema pede uma confirmação, exclui o apartamento da base de dados e refaz a tabela da tela lendo novamente os dados do banco de dados (R1).

5) Deve permitir consultar os dados do apartamento cadastrado.

Dado que:	Os apartamentos estão na tabela
Quando:	O usuário clica em qualquer dado de um apartamento em uma determinada linha
Então:	O sistema chama a HU002 – Manter Apartamento passando o parâmetro “Consultar” e os dados do apartamento da linha selecionada.

6) Deve pesquisar os apartamentos na tabela exibida na tela.

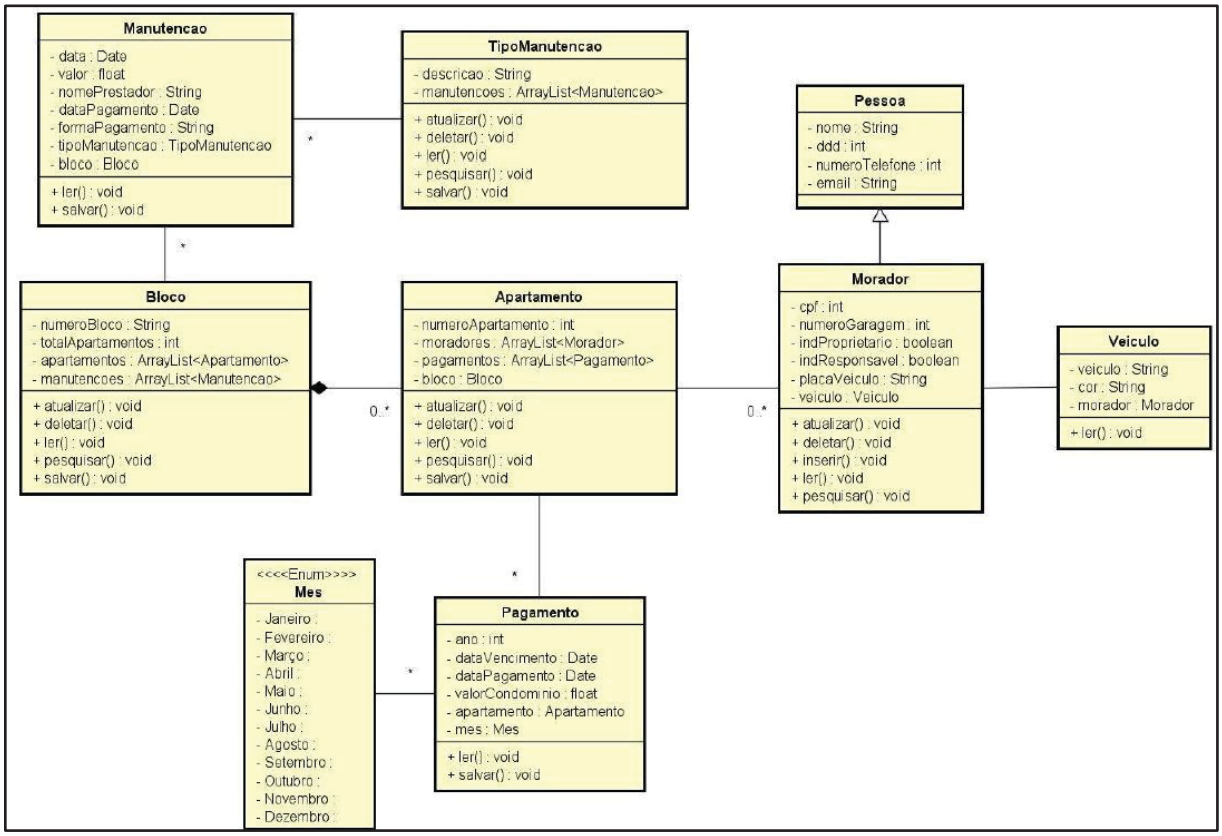
Dado que:	Os apartamentos estão na tabela
Quando:	For informado um argumento de pesquisa
Então:	O sistema filtra e apresenta na tela somente os apartamentos que obedeçam ao critério.

7) Deve permitir acesso ao menu para selecionar outra tela.

Dado que:	A tela “Pesquisar Apartamento” esteja aberta
Quando:	O botão “Menu” for pressionado
Então:	O sistema apresenta as opções de tela do Menu para o usuário abrir.

3.2 ARTEFATOS DO PROJETO - MAG2

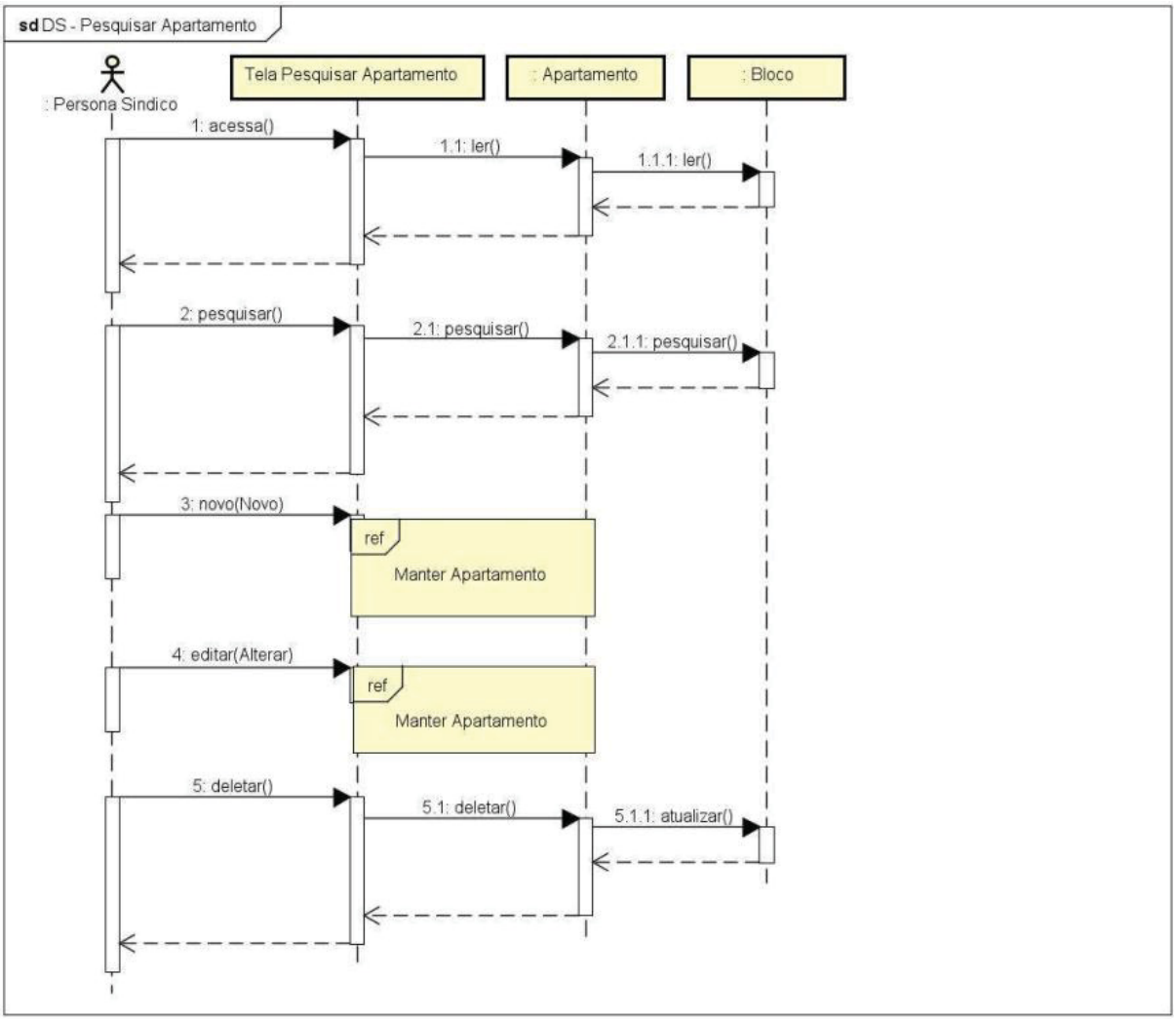
FIGURA 1: DIAGRAMA DE CLASSES



FONTE: Autor (2025)

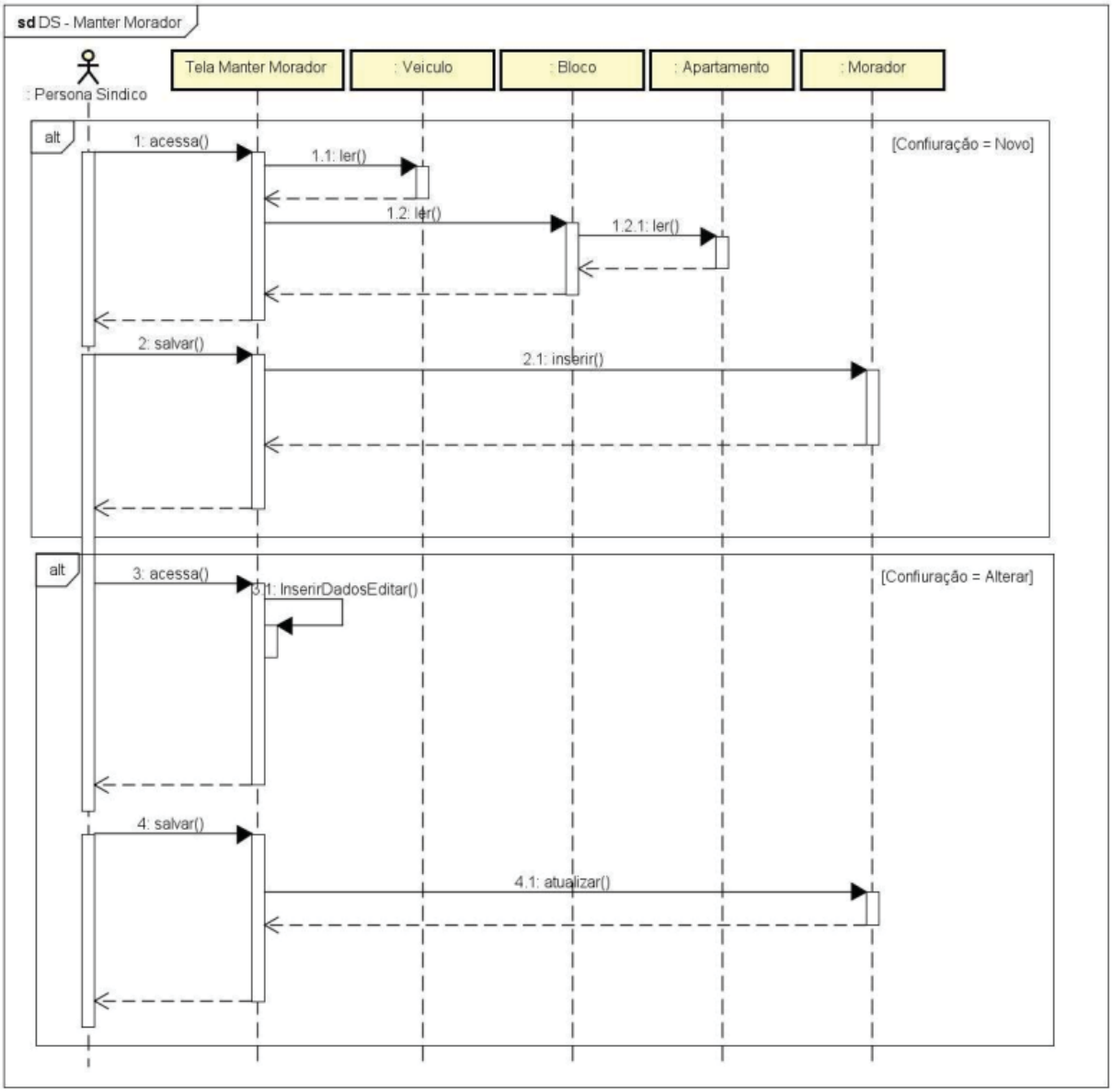
Diagramas de Sequência

FIGURA 2: PESQUISAR APARTAMENTO



FONTE: Autor (2025)

FIGURA 3: MANTER MORADORES



FONTE: Autor (2025)

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

Nas disciplinas de Gerenciamento Ágil de Projetos de Software 1 e 2, foram explorados conceitos fundamentais da gestão de projetos aplicados ao desenvolvimento ágil de software, com foco em frameworks, ferramentas visuais e práticas que garantem transparência, eficiência e entregas contínuas de valor.

Na disciplina de Gerenciamento Ágil de Software 1 (GAP1), os estudos se concentraram na compreensão dos fundamentos do gerenciamento de projetos, tanto na abordagem tradicional quanto na abordagem ágil. Foram apresentados os conceitos do PMBOK 6ª edição, suas áreas de conhecimento, grupos de processos, que orientam a gestão tradicional de projetos (Project Management Institute, 2017). Além disso, foram abordadas metodologias ágeis de concepção e planejamento, como *Design Thinking* (Brown, 2009), *Design Sprint* (Knapp; Zeratsky; Kowitiz, 2016) e *Lean Inception* (Pichler, 2018), que auxiliam na definição de problemas, geração de soluções e estruturação dos projetos de forma colaborativa.

O trabalho final de GAP1 consistiu na elaboração de um Plano de *Release*, que incluiu:

- Cálculo da velocidade da equipe;
- Definição do quadro de release, com as sprints planejadas e suas datas;
- Estimativas das histórias de usuário;
- Priorização e organização das entregas ao longo do ciclo do projeto.

Esse exercício foi essencial para entender como alinhar expectativas, organizar demandas e garantir que as entregas ocorram de maneira contínua, iterativa e incremental, características fundamentais da agilidade.

Já na disciplina de Gerenciamento Ágil de Software 2 (GAP2), o foco esteve na gestão visual de projetos, especialmente através da aplicação prática do Kanban. Foram explorados os conceitos do PMBOK 7ª edição (Project Management Institute, 2021), que adota uma visão mais moderna e alinhada com a agilidade, estruturada em princípios (como foco no valor, pensamento sistêmico, colaboração, adaptabilidade) e domínios de desempenho, como partes interessadas, equipe, ciclo de vida, planejamento, entrega, métricas e incertezas.

O projeto final de GAP2 envolveu a participação em um jogo de simulação online, que representava um fluxo de trabalho utilizando o Kanban. Nele, os alunos

precisaram gerenciar limites de trabalho em progresso (WIP - *Work In Progress*), equilibrar demandas, evitar gargalos e maximizar a eficiência do time. Ao final da simulação, os resultados foram analisados por meio de um Diagrama de Fluxo Cumulativo (Anderson, 2010), ferramenta essencial para visualizar o andamento das tarefas, o tempo de ciclo e possíveis pontos de melhoria no fluxo do projeto.

O aprendizado proporcionado por essas duas disciplinas foi fundamental para consolidar a compreensão sobre como planejar, executar, monitorar e ajustar projetos de software de maneira ágil, utilizando tanto abordagens iterativas (Scrum) quanto fluxos contínuos (Kanban). As práticas desenvolvidas em GAP1 e GAP2 são complementares e se mostram essenciais para garantir não só a organização dos projetos, mas também a entrega de valor constante ao cliente, a mitigação de riscos e a adaptação contínua às mudanças, características centrais no desenvolvimento ágil de software.

4.1 ARTEFATOS DO PROJETO

Cálculo da Velocidade:

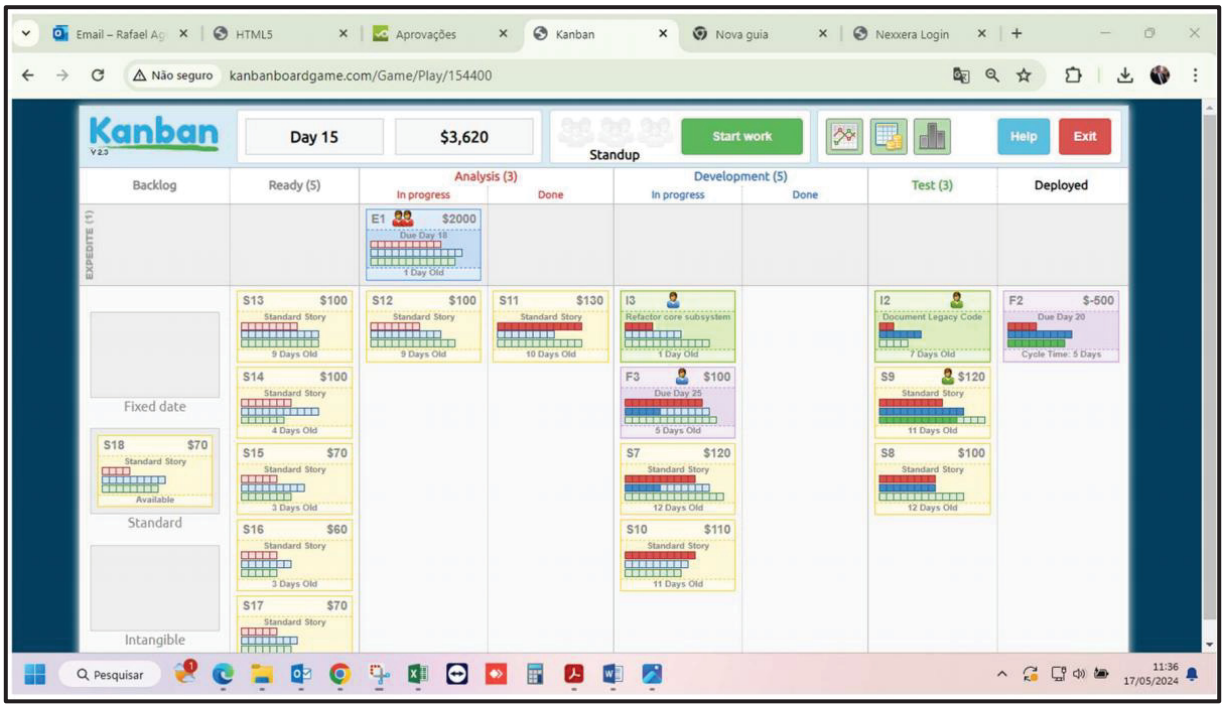
Horas disponíveis por dia:	08 hrs.	Tamanho da Sprint:	1 Semana
Horas disponíveis por Sprint:	40 hrs.	Velocidade:	5 Pontos

Plano de Release: Sistema de cadastros e registros de alunos em um centro de treinamento.

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4
Data Início: 01/04/2024	Data Início: 08/04/2024	Data Início: 15/04/2024	Data Início: 22/04/2024
Data Fim: 05/04/2024	Data Fim: 12/04/2024	Data Fim: 19/04/2024	Data Fim: 26/04/2024
< HU001 - Cadastrar Professor> **SENDO**: O Mestre **QUERO**: Cadastrar professor **PARA**: Dar aulas ESTIMATIVA (2)	< HU003 - Cadastrar Aluno> **SENDO**: O Mestre **QUERO**: Cadastrar o aluno (Rafael, Rafael, Maria, Lia, Thiago) **PARA**: Poder registrar nas respectivas aulas ESTIMATIVA (4)	< HU004 - Registrar dias de Aulas > **SENDO**: O Mestre **QUERO**: Registrar os dias de aulas (Seg: 09 hrs Muay Thai - Ter: 10 hrs Boxe - Qua: 09 hrs Muay Thai) **PARA**: Organizar as modalidades nos dias corretos ESTIMATIVA (5)	< HU005 - Registrar Presença> **SENDO**: O Mestre **QUERO**: Registrar as presenças dos alunos (Rafael, Rafael, Maria, Lia, Thiago) **PARA**: Ter o controle da presença dos alunos ESTIMATIVA (5)
< HU002 - Cadastrar Modalidades de Lutas> **SENDO**: O Mestre **QUERO**: Cadastrar modalidades de lutas (Muay Thai, Boxe, Jiu-jitsu) **PARA**: que existam aulas em seu centro de treinamento ESTIMATIVA (3)			

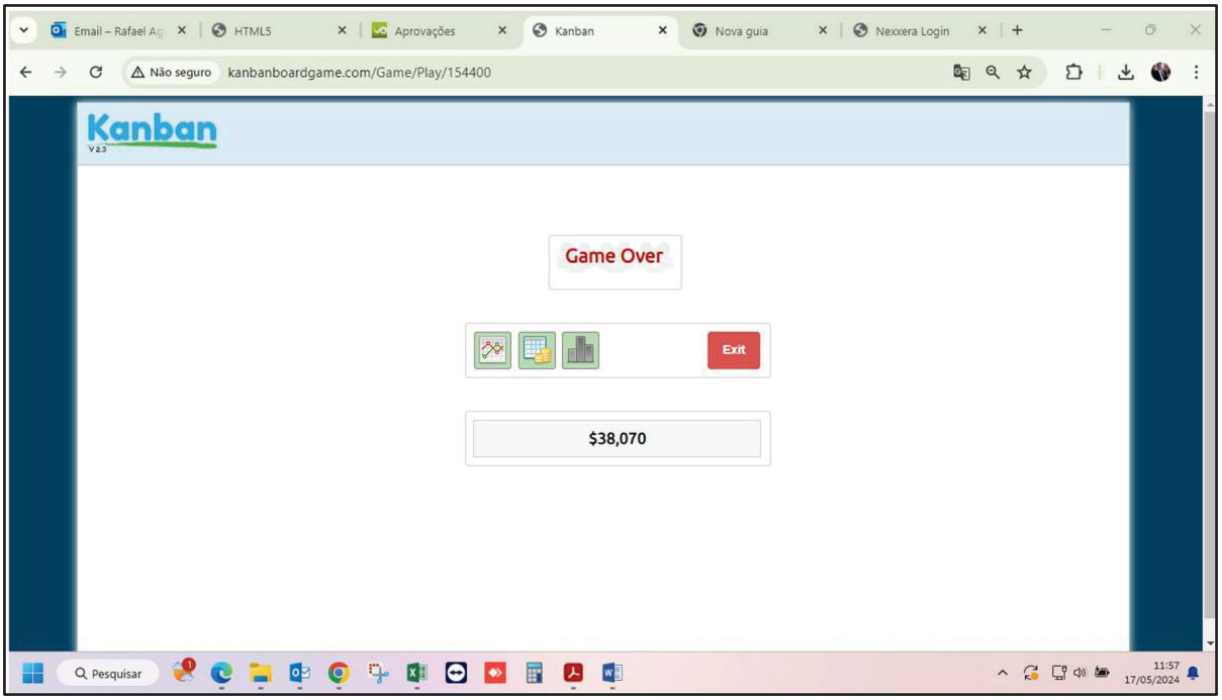
4.2 ARTEFATOS DO PROJETO - GAP 2

FIGURA 1: WIP - *WORK IN PROGRESS*



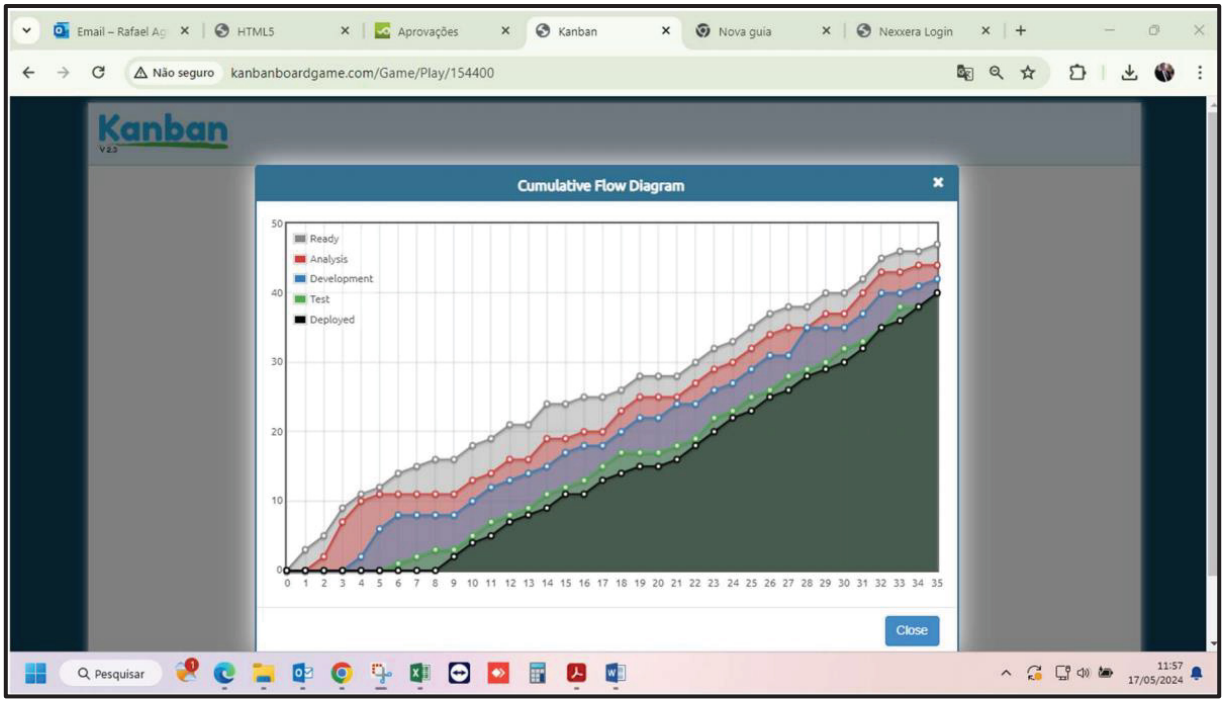
FONTE: Autor (2025)

FIGURA 2: WIP - *WORK IN PROGRESS* - FATURAMENTO



FONTE: Autor (2025)

FIGURA 3: DIAGRAMA DE FLUXO CUMULATIVO



FONTE: Autor (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação foi essencial para o desenvolvimento das competências fundamentais na construção de software, abordando desde os conceitos básicos de programação até a integração com banco de dados. Durante o curso, os principais tópicos envolvendo a sintaxe da linguagem Java (Sierra; Bates, 2009), fundamentos de orientação a objetos, tratamento de exceções, utilização de padrões de projeto como Factory e DAO (Gamma *et al*., 2007; Eckel, 2006), além de práticas de desenvolvimento orientado a testes (TDD – *Test Driven Development*) (Beck, 2003).

O projeto final consistiu na implementação de um sistema bancário simplificado em Java, com funcionalidades como cadastro de clientes, contas-correntes e investimentos. Para apoiar o desenvolvimento, foram disponibilizados:

- Um Diagrama de Classes representando a estrutura do sistema;
- Um projeto pré-configurado no NetBeans, contendo 42 testes unitários utilizando JUnit;
- Um script DDL para criação das tabelas no MySQL (Oracle, 2025).

O desafio proposto exigiu que todas as classes fossem corretamente implementadas, garantindo que, no mínimo, 40 dos 42 testes unitários fossem bem-sucedidos, critério fundamental para a obtenção da nota máxima. Essa abordagem proporcionou uma imersão prática na aplicação do TDD, metodologia ágil que prioriza o desenvolvimento dos testes antes da escrita do código de produção. Com isso, foi possível assegurar maior qualidade, manutenibilidade e robustez no sistema desenvolvido.

Além disso, foram trabalhados conceitos fundamentais da Programação Orientada a Objetos (POO), como:

- Criação e utilização de classes e objetos;
- Encapsulamento, herança, polimorfismo e interfaces;
- Relacionamento entre classes e manipulação de dados.

Outro destaque da disciplina foi o aprendizado sobre integração de aplicações Java com banco de dados MySQL, utilizando a API JDBC (Oracle, 2025) e aplicando os padrões Factory e DAO, que são fundamentais para a separação de responsabilidades, organização do código e facilidade de manutenção.

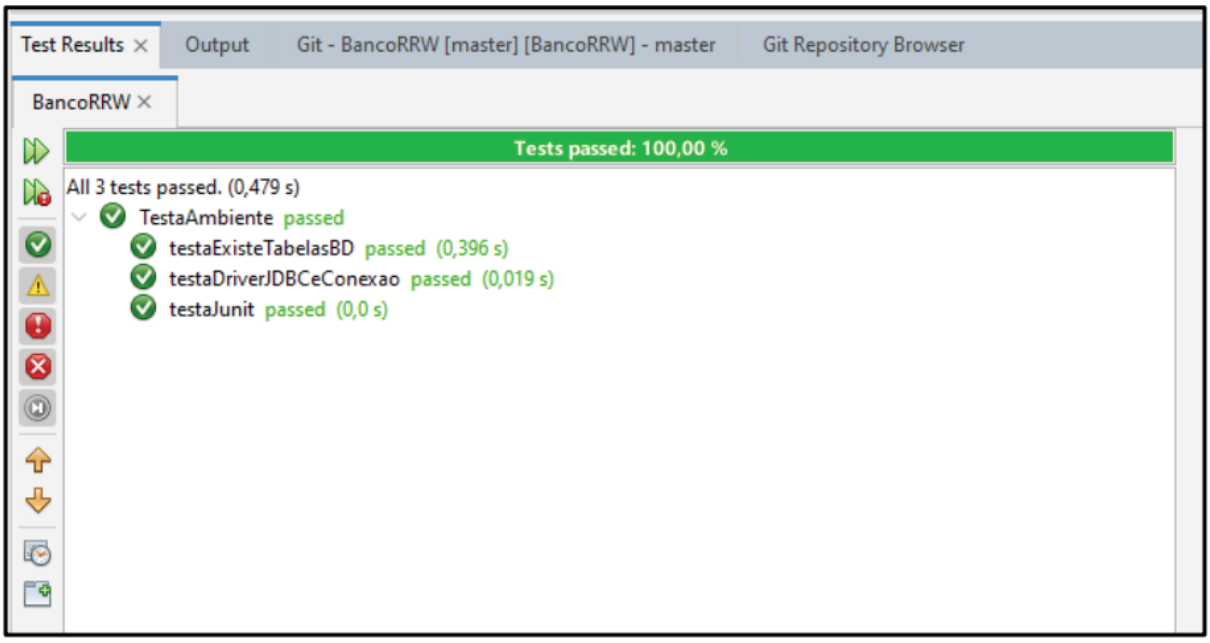
A disciplina também abordou tópicos importantes, como:

- Estrutura de controle, operadores, tratamento de erros e exceções;
- Entrada e saída de dados;
- Configuração de ambientes de desenvolvimento e utilização do NetBeans.

Essa experiência consolidou não apenas os conhecimentos técnicos em programação Java, mas também fortaleceu a compreensão sobre práticas ágeis, qualidade de software e desenvolvimento orientado a testes, pilares fundamentais para uma atuação profissional eficiente no desenvolvimento de software.

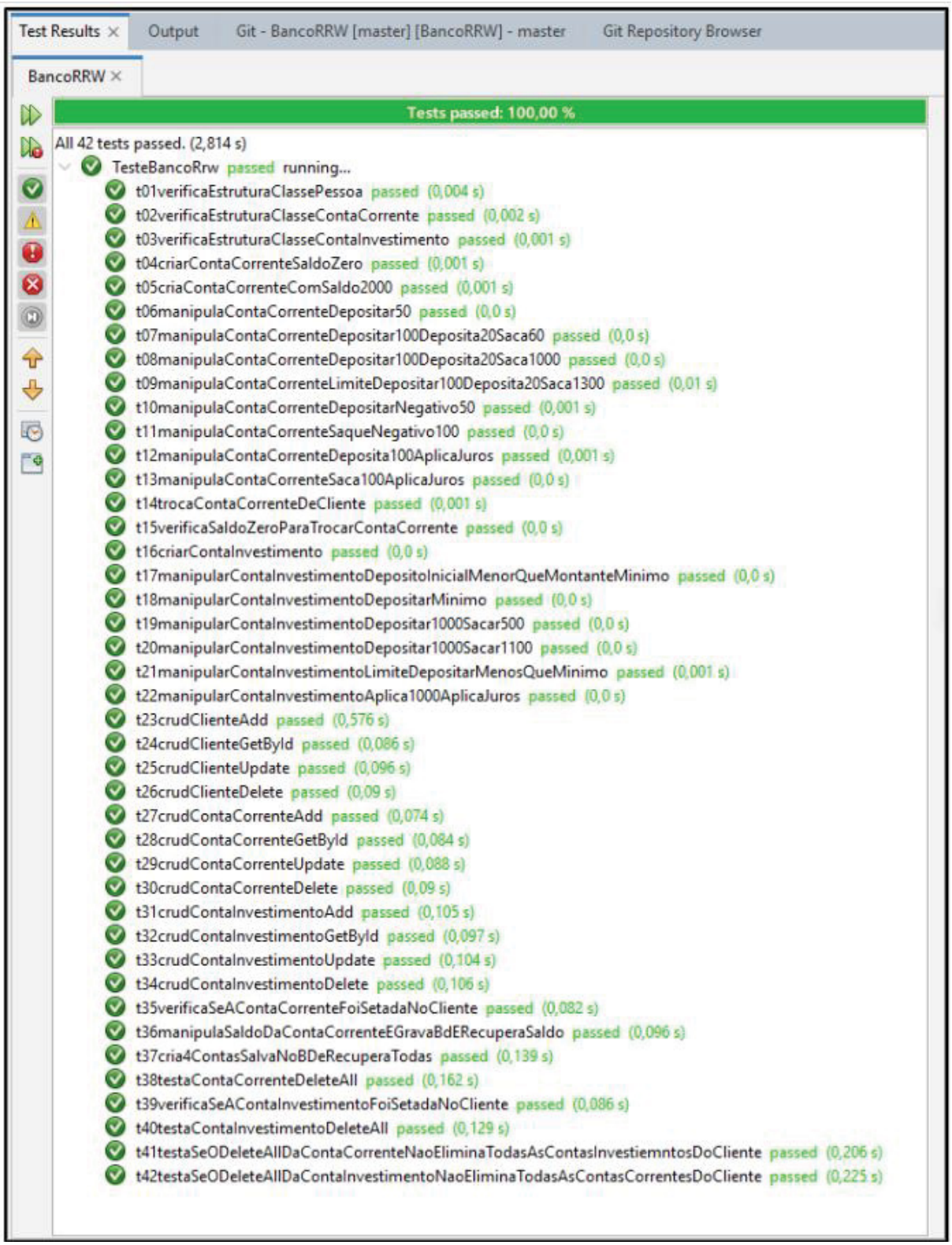
5.1 ARTEFATOS DO PROJETO

FIGURA 1: TESTE BANCO RRW



FONTE: Autor (2025)

FIGURA 2: TESTE BANCO RRW - 42



FONTE: Autor (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

A disciplina de Banco de Dados teve como foco principal desenvolver competências para modelagem, implementação e manipulação de dados, essenciais no desenvolvimento de sistemas robustos e alinhados às práticas ágeis.

O projeto foi dividido em duas etapas. Na primeira, foi elaborado um Modelo Entidade-Relacionamento (MER) no nível conceitual (Elmasri; Navathe, 2017). Posteriormente, foi construído o Diagrama Entidade-Relacionamento (DER) no modelo lógico e preparada sua implementação física em um SGBD (Date, 2004).

Na segunda etapa, o desafio envolveu um sistema de gestão de informações sobre filmes, gêneros e diretores, com a criação do modelo conceitual, seu respectivo modelo lógico e, por fim, o desenvolvimento de um *script* SQL para:

- Criação do banco de dados;
- Definição das tabelas;
- Inserção de registros de exemplo;
- Realização de consultas SQL capazes de demonstrar os diferentes tipos de relacionamentos existentes no modelo.

Durante a disciplina, foram abordados conceitos fundamentais como:

- Modelagem de dados, utilizando o MER e o DER;
- Normalização de dados, aplicando as primeiras formas normais (1FN, 2FN e 3FN) (Elmasri; Navathe, 2017) para garantir integridade e reduzir redundâncias;
- Criação e manipulação de bancos de dados relacionais, utilizando comandos SQL de definição (DDL), manipulação (DML) e consulta (DQL) (Oracle, 2025);
- Compreensão dos princípios das transações, com ênfase nas propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) (Date, 2004).

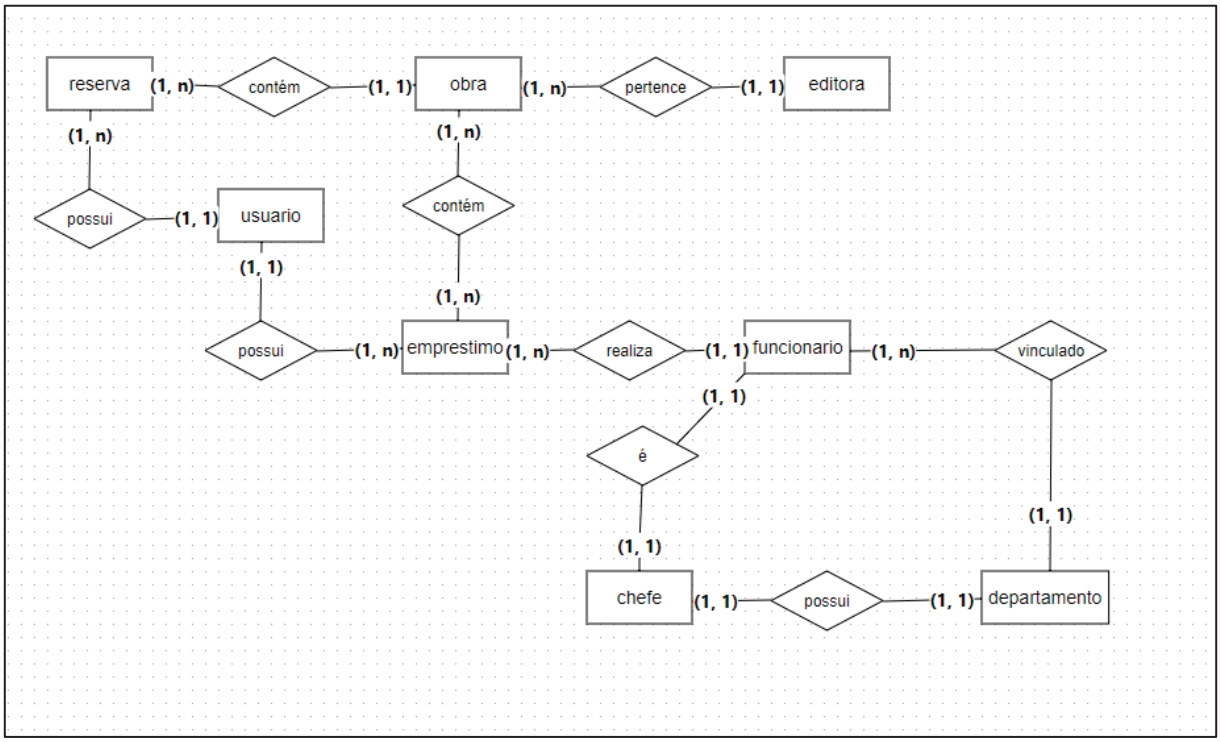
A prática da disciplina reforçou a importância de uma boa modelagem para garantir a integridade e escalabilidade dos sistemas. Foi possível compreender como um projeto mal estruturado pode impactar diretamente na manutenção, evolução e desempenho dos sistemas, elevando custos e riscos.

O conhecimento adquirido em Banco de Dados se conecta diretamente com outras disciplinas do curso, especialmente aquelas que envolvem desenvolvimento backend, frontend e mobile, visto que praticamente toda aplicação robusta necessita

de um banco de dados bem estruturado para garantir o correto funcionamento do sistema e a entrega de valor ao cliente.

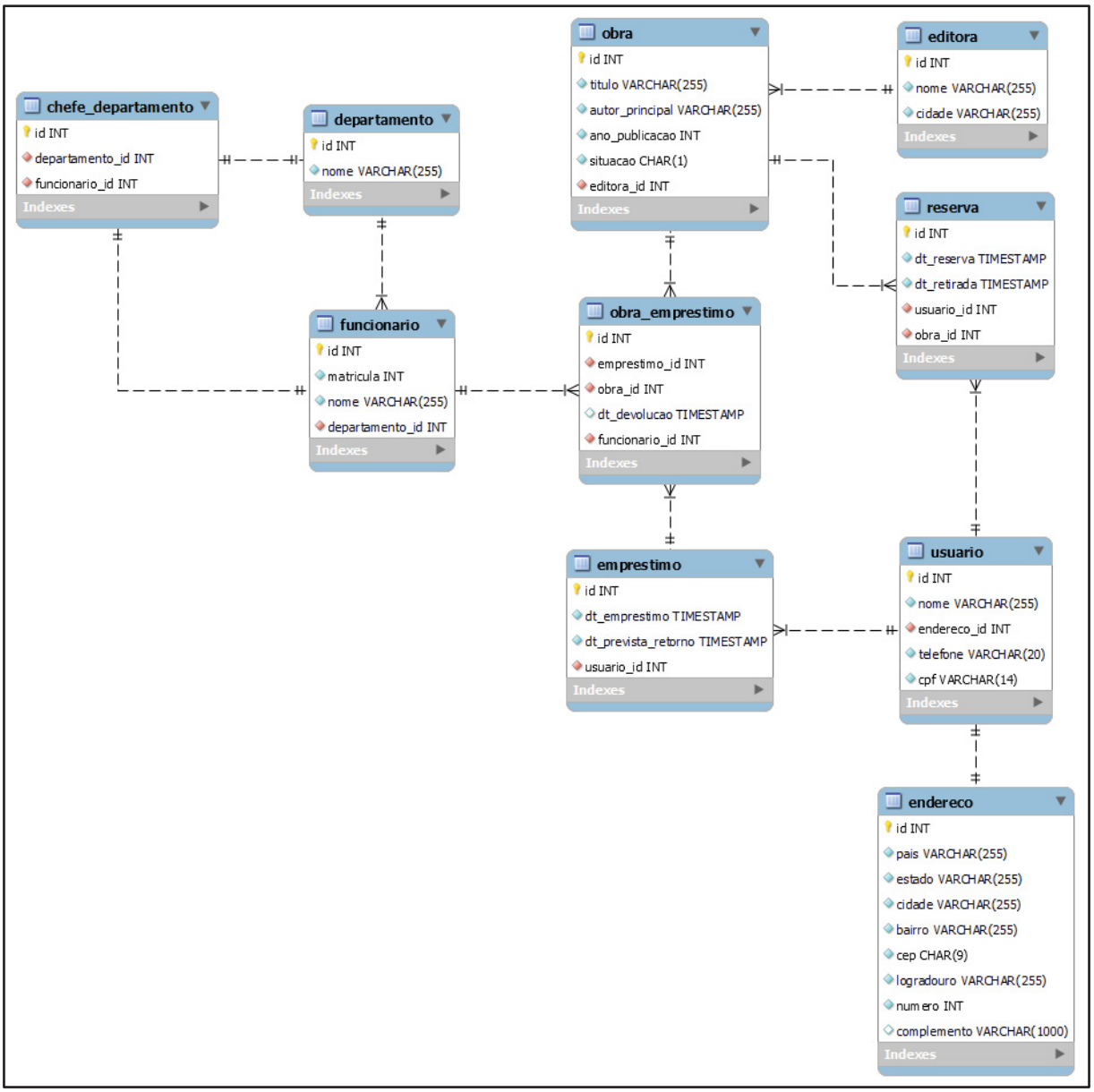
6.1 ARTEFATOS DO PROJETO

FIGURA 1: MODELO CONCEITUAL



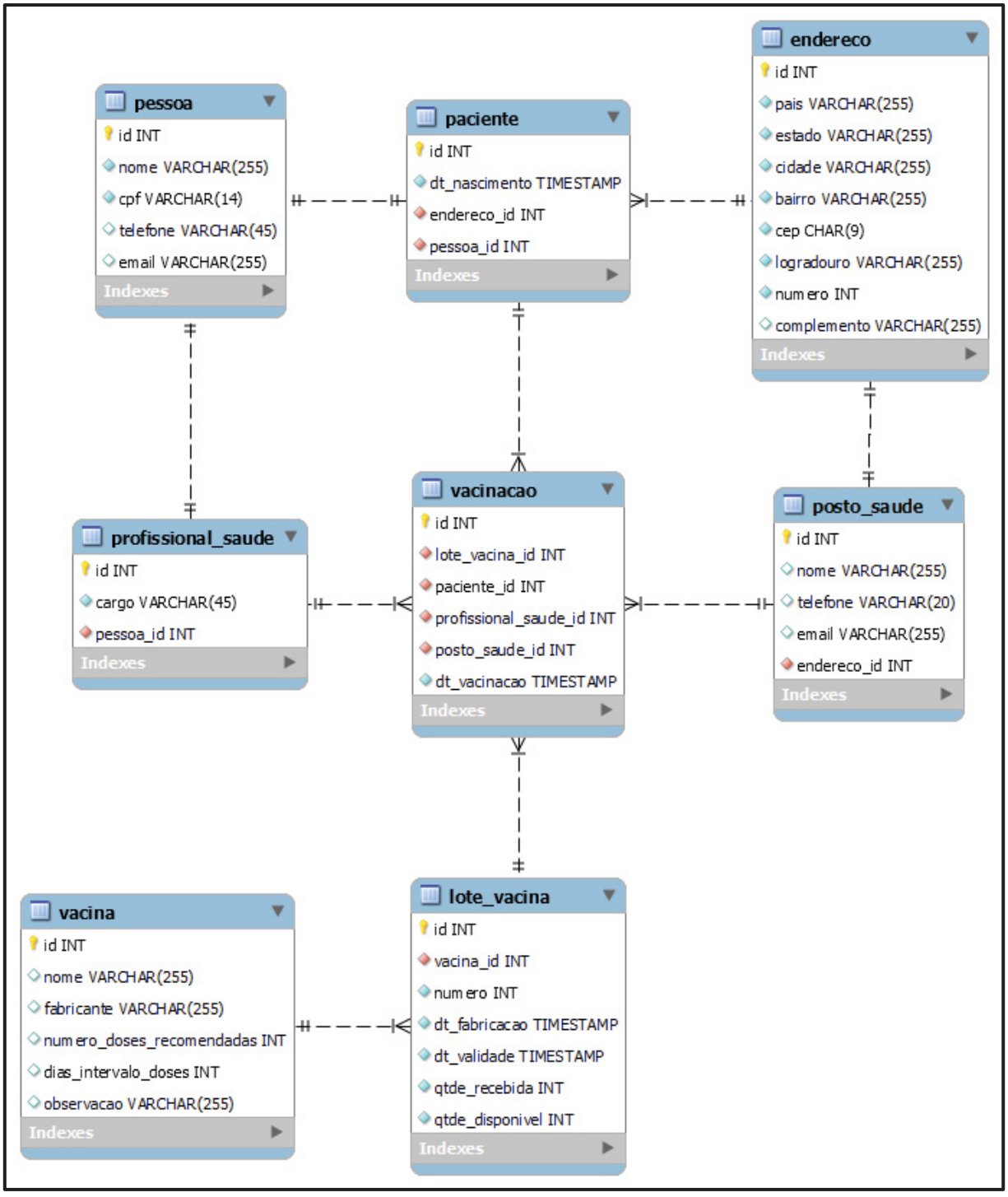
FONTE: Autor (2025)

FIGURA 2: MODELO LÓGICO



FONTE: Autor (2025)

FIGURA 2: MODELO LÓGICO



FONTE: Autor (2025)

QUADRO 1 – SCRIPT DDL DE CRIAÇÃO DO BANCO DE DADOS

```
CREATE TABLE IF NOT EXISTS `trabalhobd`.`endereco` (
  `id` INT NOT NULL auto_increment,
  `pais` VARCHAR(255) NOT NULL,
  `estado` VARCHAR(255) NOT NULL,
  `cidade` VARCHAR(255) NOT NULL,
  `bairro` VARCHAR(255) NOT NULL,
  `cep` char(9) NOT NULL,
  `logradouro` VARCHAR(255) NOT NULL,
  `numero` INT NOT NULL,
  `complemento` VARCHAR(255) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `trabalhobd`.`pessoa` (
  `id` INT NOT NULL auto_increment,
  `nome` VARCHAR(255) NOT NULL,
  `cpf` VARCHAR(14) NOT NULL,
  `telefone` VARCHAR(45) NULL,
  `email` VARCHAR(255) NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `uk_cpf` (`cpf` ASC) VISIBLE)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `trabalhobd`.`paciente` (
  `id` INT NOT NULL auto_increment,
  `dt_nascimento` TIMESTAMP NOT NULL,
  `endereco_id` INT NOT NULL,
  `pessoa_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_paciente_endereco`
    FOREIGN KEY (`endereco_id`)
    REFERENCES `trabalhobd`.`endereco` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
    CONSTRAINT `fk_paciente_pessoa`
    FOREIGN KEY (`pessoa_id`)
    REFERENCES `trabalhobd`.`pessoa` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Exemplos dos dados em tabela com a representação dos relacionamentos

QUADRO 2 – TABELA DE PESSOAS

Tabela: pessoa				
PK				
id	nome	cpf	telefone	email
1	Valmir Furtunato Almeida	757.743.949-56	(48) 99217-7041	valmir.almeida@gmail.com
2	Salvador Cruz Cocelo	281.663.569-45	(47) 97476-2052	salvador.cocelo@gmail.com
3	Dinea Pedroso Gonçalves	264.712.069-25	(48) 99584-8576	dinea.goncalves@gmail.com
4	Lurdes Saraiva Nogueira	972.656.389-56	(49) 98163-7430	lurdes.nogueira@gmail.com
5	Roseanne Ascar Bon	462.015.659-06	(49) 98439-3451	roseanne.bon@outlook.com
6	Gabriel Matta Inácio	965.711.859-03	(48) 98080-8996	gabriel.inacio@gmail.com
7	Sandro de Padua Estevam	505.431.119-03	(48) 97326-4673	sandro.estevam@gmail.com
8	Anna Pessoa Frotté	824.255.689-05	(48) 99552-2845	anna.frotte@outlook.com

QUADRO 3 – PROFISSÕES

Tabela: profissional_saude		
PK		FK
id	cargo	pessoa_id
1	Enfermeiro	1
2	Clínico Geral	2
3	Enfermeiro	3
4	Farmaceutico	4
5	Clínico Geral	5

QUADRO 4 – ENDEREÇOS

Tabela: endereco								
PK								
id	pais	estado	cidade	bairro	cep	logradouro	numero	complemento
1	Brasil	SC	Mafra	Jardim das Américas	89300-393	Rua Ruy Barbosa	50	Ao lado do posto de gasolina
2	Brasil	SC	Gaspar	Batel	89110-091	Rua Douglas Alexandre	91	Ao lado da igreja
3	Brasil	PE	Camaragibe	Cajuru	54762-837	Rua Pedro Marques	137	
4	Brasil	BA	Juazeiro	Uberaba	48907-237	Rua H	63	
5	Brasil	ES	Vitória	Mercês	29055-075	Rua Ruy Barbosa	12	
6	Brasil	SC	Mafra	Roça Grande	89300-393	Rua Ruy Barbosa	98	
7	Brasil	SC	Gaspar	Vila Zumbi	89110-091	Rua Douglas Alexandre	102	
8	Brasil	PE	Camaragibe	São Gabriel	54762-837	Rua Pedro Marques	180	
9	Brasil	BA	Juazeiro	Maracanã	48907-237	Rua H	5	
10	Brasil	ES	Vitória	Monza	29055-075	Rua Ruy Barbosa	130	

QUADRO 5 – PACIENTES

Tabela: paciente			
PK		FK	FK
id	dt_nascimento	endereco_id	pessoa_id
1	17/03/1978 00:00	1	4
2	23/06/1985 00:00	2	5
3	07/12/1989 00:00	3	6
4	15/02/1993 00:00	4	7
5	19/07/1972 00:00	5	8

QUADRO 6 – UNIDADES DE SAÚDE

Tabela: posto_saude				
PK				FK
id	nome	telefone	email	endereco_id
1	Unidade Saúde Mafra	(69) 3115-6621	unidadesaudemafra@sus.br	6
2	Unidade Saúde Gaspar	(98) 2759-6137	unidadesaudegaspar@sus.br	7
3	Unidade Saúde Camaragibe	(96) 2914-1842	unidadesaudecamaragibe@sus.br	8
4	Unidade Saúde Juazeiro	(93) 3148-3348	unidadesaudejuazeiro@sus.br	9
5	Unidade Saúde Vitória	(28) 2286-0618	unidadesaudevitoria@sus.br	10

QUADRO 7 – VACINAS

Tabela: vacina				
PK				
id	nome	fabricante	numero_doses_recomendadas	dias_intervalo_doses
1	Tríplice viral	Sinovac	3	30
2	Pneumocócica 23 Valente	Pfizer	1	0
3	Hepatite B	Sinovac	3	180
4	Febre Amarela	Butantan	1	0
5	Meningocócica C - reforço	Butantan	3	0

QUADRO 8 – LOTES DE VACINAS

Tabela: lote_vacina						
PK	FK					
id	vacina_id	numero	dt_fabricacao	dt_validade	qtde_recebida	qtde_disponivel
1	1	1001	30/04/2024 09:44	29/07/2024 09:44	15	14
2	2	1002	09/06/2024 09:44	14/07/2024 09:44	5	4
3	3	1004	19/06/2024 09:44	24/07/2024 09:44	7	5
4	4	1003	30/05/2024 09:44	28/08/2024 09:44	19	18
5	5	1005	14/06/2024 09:44	19/07/2024 09:44	11	10

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como foco a aplicação prática de boas práticas de codificação, baseadas no conceito de Clean Code (Martin, 2009), visando a construção de software legível, manutenível e de alta qualidade — características indispensáveis no contexto ágil.

O projeto final consistiu na refatoração do algoritmo de ordenação Bubble Sort (Cormen *et al*., 2012), aplicando no mínimo três melhorias significativas no código original. Dentre as alterações propostas estavam:

- A atribuição de nomes significativos às variáveis e métodos;
- A remoção de comentários desnecessários;
- A criação de métodos com responsabilidade única, reforçando o princípio do SRP (Single Responsibility Principle) (Martin, 2002).

Durante a disciplina, foram trabalhados diversos temas fundamentais, como:

- Nomes significativos para variáveis, funções e classes;
- Organização de funções pequenas e específicas;
- Comentários úteis e objetivos, apenas quando realmente necessários;
- Formatação padronizada para facilitar a leitura em equipe;
- Encapsulamento, coesão, acoplamento fraco e aplicação da Lei de Demeter (Martin, 2002);
- Princípios de refinamento sucessivo e a filosofia do "*boy scout rule*": deixar o código melhor do que encontrou;
- Boas práticas de tratamento de erros, minimização de efeitos colaterais e uso apropriado de exceções.

Além disso, a disciplina introduziu práticas colaborativas, como o *Pair Programming* e o *Mob Programming*, detalhando estilos, papéis e vantagens dessas abordagens no ambiente ágil. Também foi discutido o uso de ferramentas e técnicas para ambientes remotos e presenciais, promovendo a colaboração eficaz entre os membros do time de desenvolvimento.

A metodologia de ensino incluiu demonstrações de refatoração com aplicação de técnicas como:

- Extração de métodos e variáveis;
- Substituição de código duplicado por abstrações reutilizáveis;
- Separação de responsabilidades;

- Aplicação da Regra do Passo Descendente, onde o código é organizado de forma a contar uma história de cima para baixo (Martin, 2009).

A aplicação prática dos conceitos de *Clean Code* reforçou a importância de um código limpo, legível e sustentável no desenvolvimento ágil. Essa base teve impacto direto nas demais disciplinas do curso, como programação, testes automatizados e desenvolvimento web/mobile, promovendo soluções mais bem estruturadas e compreensíveis.

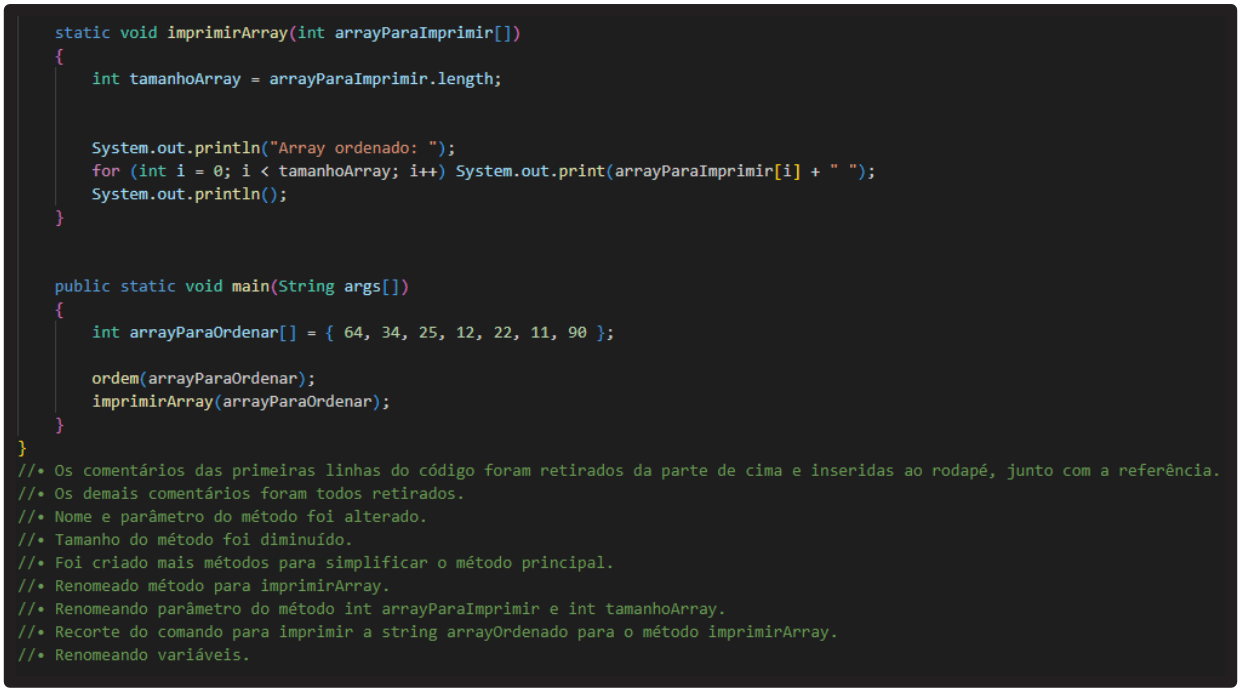
7.1 ARTEFATOS DO PROJETO

FIGURA 1: *CLEAN CODE*

```java
package trabalho.aap;

import java.io.*;

class BubbleSort {


    static void ordem(int arr[])
    {
        int n = arr.length;
        int i;
        boolean trocado;
        for (i = 0; i < n - 1; i++) {
            trocado = false;
            trocado = percorrendoItensParaAtualizarValores(n, i, arr, trocado);


            if (trocado == false)
                break;
        }
    }

    public static boolean percorrendoItensParaAtualizarValores(int n, int i, int[] arr, boolean trocado) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {


                trocado = trocarValor(arr, j);
            }
        }
        return trocado;
    }

    public static boolean trocarValor(int[] arr, int j) {
        int valorAtual;
        boolean trocado;
        valorAtual = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = valorAtual;
        trocado = true;
        return trocado;
    }
```

FONTE: Autor (2025)

FIGURA 2: *CLEAN CODE* E COMENTÁRIOS



```
    static void imprimirArray(int arrayParaImprimir[])
    {
        int tamanhoArray = arrayParaImprimir.length;


        System.out.println("Array ordenado: ");
        for (int i = 0; i < tamanhoArray; i++) System.out.print(arrayParaImprimir[i] + " ");
        System.out.println();
    }


    public static void main(String args[])
    {
        int arrayParaOrdenar[] = { 64, 34, 25, 12, 22, 11, 90 };

        ordem(arrayParaOrdenar);
        imprimirArray(arrayParaOrdenar);
    }
}
//• Os comentários das primeiras linhas do código foram retirados da parte de cima e inseridas ao rodapé, junto com a referência.
//• Os demais comentários foram todos retirados.
//• Nome e parâmetro do método foi alterado.
//• Tamanho do método foi diminuído.
//• Foi criado mais métodos para simplificar o método principal.
//• Renomeado método para imprimirArray.
//• Renomeando parâmetro do método int arrayParaImprimir e int tamanhoArray.
//• Recorte do comando para imprimir a string arrayOrdenado para o método imprimirArray.
//• Renomeando variáveis.
```

FONTE: Autor (2025)

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

As disciplinas de Desenvolvimento Web 1 e 2 proporcionaram uma imersão nas práticas de criação de aplicações web modernas, utilizando tecnologias amplamente adotadas no mercado como Angular, Spring Boot e PostgreSQL. Os projetos e conteúdos abordaram desde os fundamentos do *frontend* até a integração com o *backend*, promovendo uma visão completa do desenvolvimento *full-stack*.

Em Desenvolvimento Web 1 (WEB1), o projeto final consistiu na criação de dois CRUDs para as entidades Aluno e Curso, utilizando o *framework* Angular, com persistência dos dados em *Local Storage*. O desenvolvimento não exigia *backend*, o que permitiu um foco maior na lógica de *frontend*, modelagem dos dados no cliente e construção de interfaces dinâmicas. A estrutura do projeto envolveu a utilização de:

- Componentes Angular para criação das telas;
- *Bootstrap* para estilização das interfaces;
- *NgModel* para o *binding* de dados;
- Diretivas personalizadas, como validações de campos;
- Serviços Angular para gerenciamento da lógica de negócios e acesso aos dados.

Além disso, as aulas abordaram os fundamentos de TypeScript (Hejlsberg *et al.*, 2022), HTML5 (Duckett, 2011), modularização de aplicações Angular e boas práticas no desenvolvimento de aplicações SPA (Single Page Applications) (Freeman; Sanderson, 2022).

Já em Desenvolvimento Web 2 (WEB2), o foco se voltou para o backend. O projeto proposto envolvia a implementação de um servidor REST em Spring Boot, com banco de dados relacional PostgreSQL, para dar suporte aos CRUDs desenvolvidos no *frontend*. Apesar de não ter sido realizada a implementação prática, os conteúdos abordados em aula permitiram o entendimento aprofundado sobre:

- Criação de APIs RESTful com Spring Boot (Walls, 2016);
- Utilização do Spring Data JPA para acesso a dados (Schmidt, 2020);
- Mapeamento objeto-relacional (ORM) com JPA;
- Configuração e conexão com banco de dados PostgreSQL;
- Consumo de APIs via Angular utilizando *HTTPClient* e *observables*.

Também foram explorados temas como programação reativa, autenticação de usuários (*login*), e validação de formulários com diretivas personalizadas, fortalecendo o conhecimento prático necessário para o desenvolvimento web completo.

A integração entre essas duas disciplinas evidenciou o papel central do desenvolvimento web no ciclo ágil. A capacidade de rapidamente criar e adaptar interfaces, junto à conexão eficiente com o *backend*, promove entregas contínuas de valor, com fácil adaptação às mudanças de escopo.

Mesmo que a implementação prática do *backend* não tenha sido realizada, a compreensão dos conceitos arquiteturais, das tecnologias envolvidas e das boas práticas discutidas em aula contribuiu significativamente para a formação como desenvolvedor ágil, preparado para atuar em ambientes modernos de desenvolvimento *full-stack*.

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

A disciplina de UX no Desenvolvimento Ágil de Software destacou a importância da experiência do usuário como elemento central no desenvolvimento de soluções digitais. A proposta do projeto final envolveu a criação de telas para um aplicativo móvel de controle de investimentos pessoais, com foco em uma interface funcional, intuitiva e alinhada às necessidades reais dos usuários.

O processo iniciou com a justificativa da escolha do tema, considerando a relevância de ferramentas que auxiliem no controle financeiro individual e os benefícios associados à autonomia na gestão de investimentos. A partir dessa definição, o projeto evoluiu para a elaboração das interfaces no Figma, utilizando os princípios de design visual, acessibilidade e usabilidade discutidos ao longo da disciplina.

Durante o desenvolvimento, foram aplicados conceitos como:

- Design emocional, para criar conexões positivas com o usuário;
- Hierarquia visual, promovendo clareza e foco nas informações;
- Prototipagem de alta fidelidade, para simular com precisão a experiência real do sistema;
- Design acessível, utilizando contraste adequado, elementos compreensíveis e navegação intuitiva.

A escolha das cores, tipografia e elementos gráficos buscou não apenas a estética, mas também a clareza na comunicação das informações. Em particular, os gráficos financeiros foram projetados com cores diferenciadas para representar variações de desempenho, otimizando a interpretação visual dos dados.

A ferramenta Figma foi essencial no processo de prototipação, permitindo iteração rápida e colaboração em tempo real. Essa agilidade se alinha diretamente aos valores do desenvolvimento ágil, que preza por entregas incrementais e adaptações contínuas com base em feedback.

Além disso, os testes de usabilidade realizados com os protótipos permitiram a identificação de melhorias e ajustes antes mesmo da implementação, reforçando a prática de validação contínua e evitando retrabalho em etapas posteriores.

A disciplina reforçou que a experiência do usuário é um componente estratégico no desenvolvimento de software. Em ambientes ágeis, onde as mudanças são constantes e o tempo para entrega é curto, projetar com foco no usuário é

essencial para garantir a aceitação do produto, satisfação do cliente e eficiência na evolução da solução.

9.1 ARTEFATOS DO PROJETO

FIGURA 1: *HOME*

FONTE: Autor (2025)

FIGURA 2: CRIAÇÃO DE USUÁRIO

FONTE: Autor (2025)

FIGURA 3: INFORMAÇÕES DO AVISO DE RECEBIMENTO

FONTE: Autor (2025)

FIGURA 4: PROJETOS DA CONSTRUTORA

FONTE: Autor (2025)

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

As disciplinas de Desenvolvimento Mobile 1 e 2 tiveram como objetivo principal proporcionar uma base sólida no desenvolvimento de aplicativos Android, aliando teoria e prática com foco nas demandas reais do mercado e nos princípios do desenvolvimento ágil.

Em Desenvolvimento Mobile (MOB1), o projeto propôs o desenvolvimento de um aplicativo Android voltado para a gestão financeira pessoal, permitindo o cadastro de receitas e despesas. A aplicação visava facilitar a organização e categorização dos gastos dos usuários, contribuindo para um melhor controle das finanças. Os conceitos explorados ao longo da disciplina incluíram:

- Introdução à IDE Android Studio (Phillips *et al*., 2022);
- Componentes de UI (TextView, EditText, Button, SeekBar) (Google, 2025);
- Manipulação de eventos com métodos como *onClick* (Google, 2025);
- *Intents* e navegação entre múltiplas *activities* (Google, 2025);
- Padrão de projeto MVC/MVVM (Freeman; Robson, 2008);
- Uso de layouts responsivos com *ConstraintLayout* (GoogleE, 2025);
- Criação de apps simples como: *Frase do Dia*, *CambioApp*, *Churrascometro*, entre outros (Phillips *et al*., 2022).

Já em MOB2, o foco avançou para integração com web services e persistência local de dados. O projeto sugerido consistia na criação do HarryPotterDBApp, um aplicativo que consumia uma API ou banco local SQLite para inserir, atualizar, deletar e listar personagens da saga Harry Potter. Os principais conteúdos abordados incluíram:

- RecyclerView e customização de listas com Adapter e ViewHolder;
- SQLite, DBHelper e DAO para persistência local;
- Acesso a dados com ContentValues, cursosres e comandos SQL no Android;
- Lambda functions para escuta de eventos e simplificação de código;
- Estruturação de projetos com os pacotes model, dao, db e controller;
- Persistência de dados usando arquivos internos (*openFileOutput, openFileInput*) como alternativa ao SQLite.

Apesar de os projetos de MOB1 e MOB2 serem opcionais, o estudo dos materiais, vídeos, questionários e exemplos práticos apresentados em aula possibilitou uma compreensão clara do ciclo completo de desenvolvimento mobile.

Isso inclui desde a concepção da ideia e criação das interfaces, até o armazenamento local dos dados e a manipulação de listas dinâmicas.

As disciplinas reforçaram a importância dos aplicativos móveis no ecossistema ágil, possibilitando:

- Entregas rápidas e incrementais;
- Validação contínua com o usuário;
- Ajustes dinâmicos às necessidades do mercado;
- Distribuição eficiente de soluções com alto potencial de impacto social.

Por fim, a conexão com outras disciplinas do curso, como UX, BD, INTRO e WEB, evidenciou o caráter interdisciplinar do desenvolvimento mobile, que exige domínio de diversas tecnologias para entregar aplicações completas, escaláveis e centradas no usuário.

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 o aprofundamento em práticas modernas de automação, versionamento de código e implantação contínua de software. O projeto final envolveu uma série de atividades práticas utilizando ferramentas amplamente adotadas na indústria, como Docker (Merkel, 2014) e GitLab (GitLab, 2025), reforçando os pilares de automação, integração e entrega contínua no desenvolvimento ágil.

O trabalho consistiu nas seguintes etapas:

- Criação de um container Docker a partir da imagem `dfwandarti/gitlab_jenkins:3`, nomeando-o com a matrícula do aluno e publicando as portas 22, 80, 443 e 9091;
- Acesso ao container via terminal e extração da senha do GitLab para fins de autenticação;
- Execução de comandos Git fundamentais para controle de versão, como *clone, add, commit* e *push*.

Um diferencial na abordagem deste aluno foi o uso tanto da linha de comando quanto de interface visual para manipulação dos containers Docker. Essa prática ampliou a compreensão sobre a flexibilidade da ferramenta e demonstrou a aplicabilidade em diferentes contextos e níveis de experiência.

A disciplina também explorou conceitos fundamentais do ciclo de vida de software (SDLC) sob a ótica do DevOps, abordando:

- Integração e entrega contínuas (CI/CD) (Humble; Farley, 2011);
- Infraestrutura como código (IaC) (Morris; Smith, 2020);
- Gerenciamento de ambientes com Docker e Kubernetes (Merkel, 2014; Hightower; Burns; Beda, 2019);
- Segurança integrada (DevSecOps) (Munshi, 2021);
- Observabilidade e métricas com foco em confiabilidade e desempenho (DORA *Metrics*: DF, MLTC, CFR, MTTR) (DORA Metrics) *(*Forsgren; Humble; Kim*, 2018)*;

- Versionamento com Git (Chacon; Straub, 2014), incluindo *branching*, *merge*, *rebase*, *cherry-pick*, *squash* e resolução de conflitos.

A prática com Dockerfiles, containers personalizados e rede de containers demonstrou na prática como ambientes isolados e reproduzíveis podem ser facilmente construídos, reduzindo erros de configuração e acelerando o processo de deploy. A criação de imagens otimizadas e o uso de boas práticas de segurança em containers também foram enfatizados.

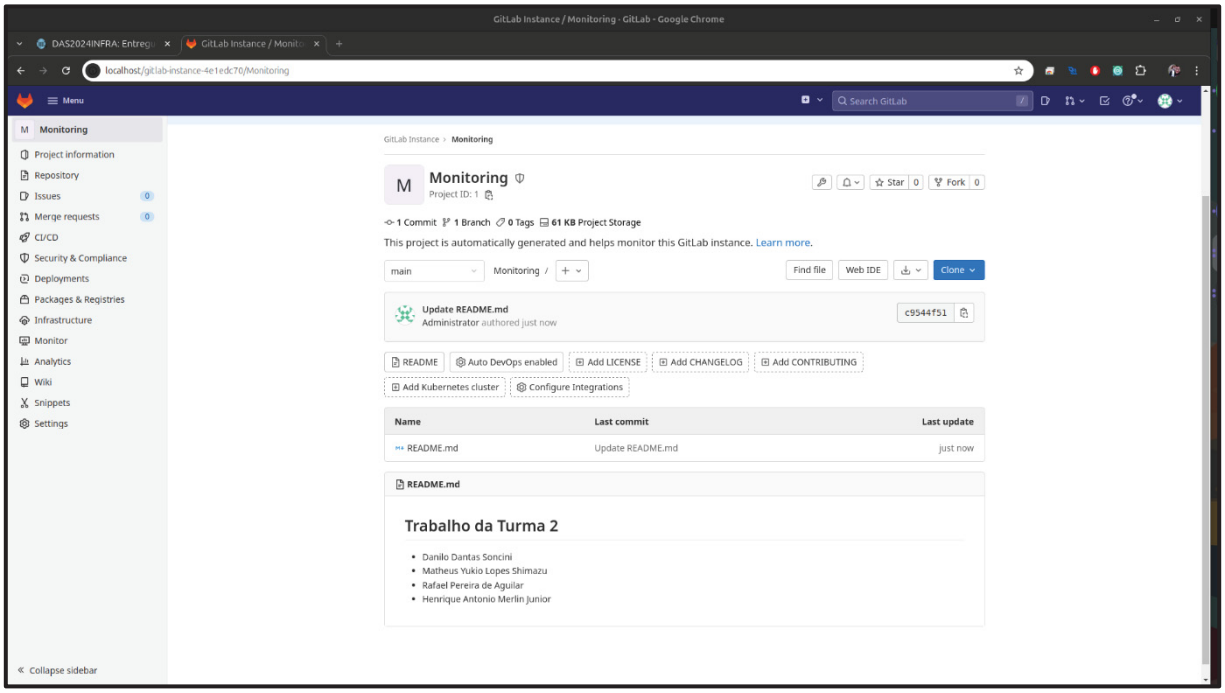
Esses aprendizados são altamente relevantes dentro do contexto do desenvolvimento ágil, pois DevOps promove:

- Entrega contínua de valor;
- Respostas rápidas a mudanças;
- Redução de falhas em produção;
- Melhoria contínua dos processos;
- Colaboração entre times de desenvolvimento, operação e segurança.

A experiência proporcionada pela disciplina de Infraestrutura serviu como uma ponte entre os conhecimentos técnicos adquiridos nas demais disciplinas e sua operacionalização no mundo real, reforçando a visão de um ciclo de desenvolvimento completo, eficiente e alinhado às exigências do mercado.

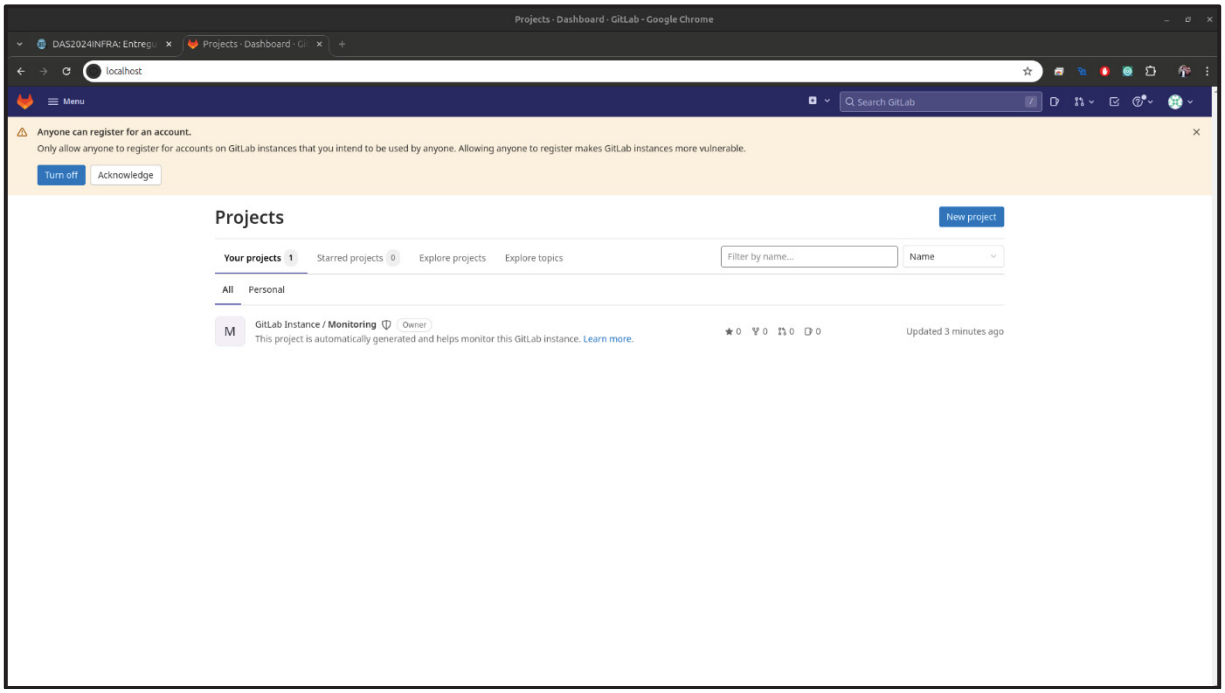
11.1 ARTEFATOS DO PROJETO

FIGURA 1: REPOSITÓRIO



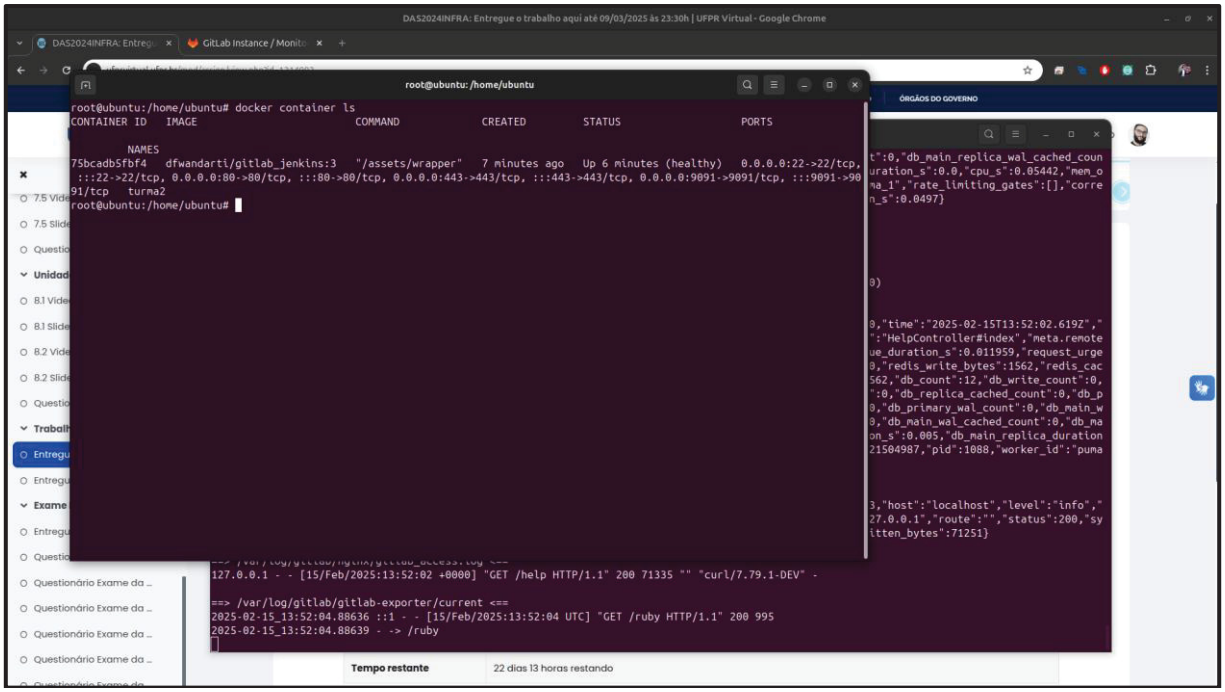
FONTE: Autor (2025)

FIGURA 2: GITLAB



FONTE: Autor (2025)

FIGURA 3: *CONTAINER*



FONTE: Autor (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados encerrou o ciclo de especialização com a consolidação de diversos conceitos abordados ao longo do curso, focando na qualidade do software e na construção de testes robustos, alinhados às práticas ágeis de desenvolvimento.

O projeto final consistiu na criação de um teste automatizado em TypeScript (Hejlsberg *et al*., 2022), utilizando o *framework* Selenium (Burns; McGregor, 2022) em conjunto com o navegador Microsoft Edge (Microsoft, 2025). O script acessava o site `https://pt.anotepad.com`, preenchia automaticamente o campo de título com "Entrega trabalho TEST DAS 2024" e, no corpo da nota, inseria o nome e a matrícula do aluno. Essa automação demonstrou um cenário real de teste end-to-end (E2E), no qual a interação simulada do usuário é validada de forma automatizada.

Durante a disciplina, foram abordados diversos conceitos fundamentais para a prática de testes em ambientes ágeis, tais como:

- Diferenças entre verificação e validação;
- Tipos de testes: unitários, integração, sistema, regressão, aceitação e testes de caixa preta/branca;
- Princípios como o "Shift-Left", que promove a antecipação dos testes ainda nas fases iniciais do desenvolvimento;
- Estratégias para aumentar a cobertura de código, respeitando os limites da eficácia dos testes;
- Utilização de ferramentas como JUnit (Gammelgaard, 2017), Playwright (Microsoft, 2025), Mockito (Kaczmarzyk, 2014) e práticas como Testes Parametrizados, Stubs, Mocks, Assumptions, Page Object Models e CI/CD com testes automatizados (Meszaros, 2007).

A disciplina também enfatizou o papel de qualidade contínua dentro do ciclo ágil, explorando a Pirâmide de Testes, que sugere uma base forte de testes unitários, suporte com testes de integração e cobertura complementar com testes de sistema e testes manuais exploratórios.

O exercício final, embora simples em sua essência, envolveu conceitos complexos de:

- Localização de elementos DOM via seletores;

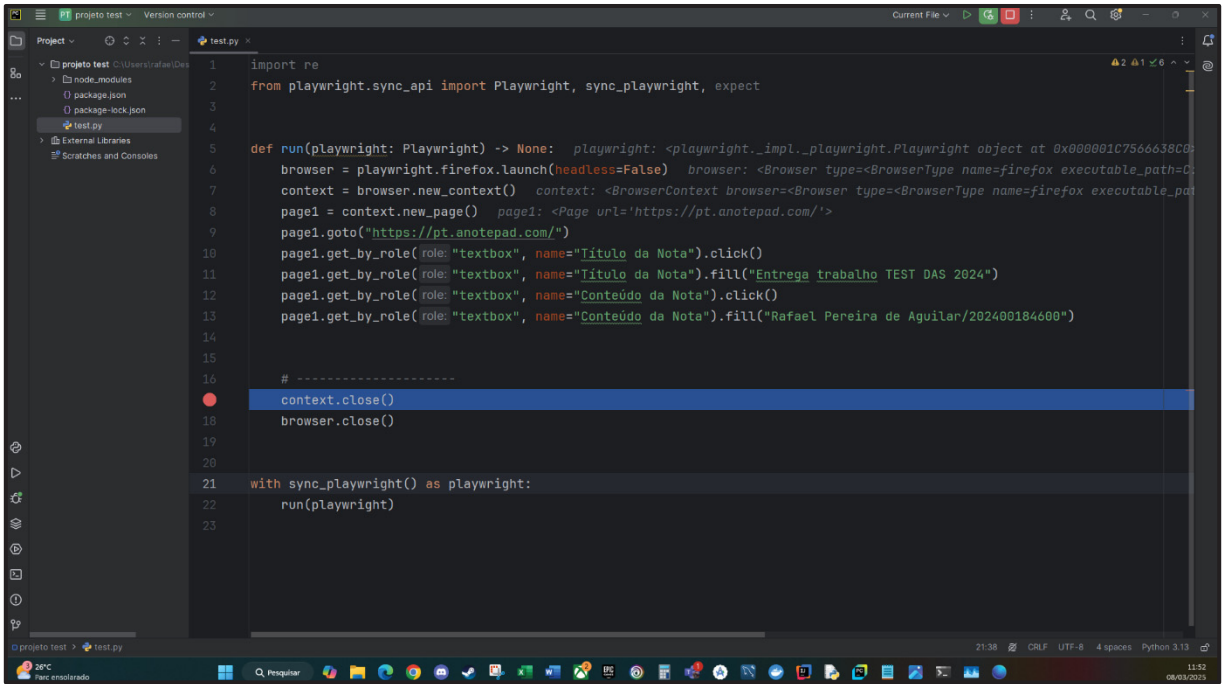
- Manipulação de inputs e ações de clique;
- Execução e verificação de comportamento esperado via assertivas;
- Execução assíncrona e sincronização de elementos da interface.

Essa experiência reforçou a importância da automação de testes como aliada da entrega contínua de valor ao cliente, promovendo segurança nas alterações de código, agilidade no ciclo de desenvolvimento e confiabilidade na validação de funcionalidades.

A disciplina de Testes Automatizados completou o ciclo de formação, estabelecendo uma base sólida para a prática da qualidade como parte integrante do desenvolvimento, e não como uma etapa posterior. Em contextos ágeis, essa mentalidade é essencial para garantir a escalabilidade, sustentabilidade e adaptabilidade das soluções desenvolvidas.

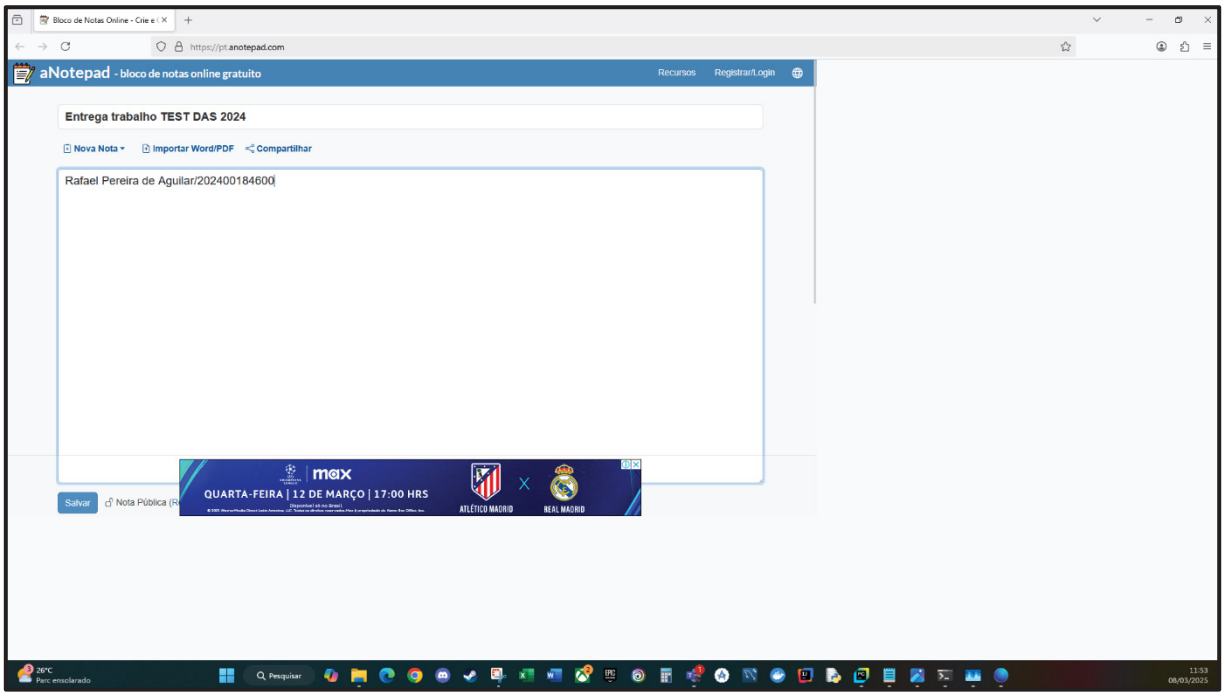
12.1 ARTEFATOS DO PROJETO

FIGURA 1: CÓDIGOS



FONTE: Autor (2025)

FIGURA 1: CÓDIGO RODANDO



FONTE: Autor (2025)

13 CONCLUSÃO

Ao longo deste memorial, foi possível consolidar uma jornada de aprendizado baseada em práticas modernas de desenvolvimento de software, com ênfase na adoção de métodos ágeis em todas as etapas do processo. Cada disciplina contribuiu de forma específica para a formação de uma visão integrada, que vai desde a concepção e modelagem de sistemas até sua implementação, validação e implantação.

As etapas iniciais destacaram a importância do planejamento estruturado por meio da modelagem de requisitos, utilização de histórias de usuário e representação visual de sistemas. Em seguida, as disciplinas técnicas aprofundaram o uso de linguagens de programação, frameworks e ferramentas como Angular, Spring Boot, PostgreSQL, Docker e Selenium, permitindo o contato direto com soluções amplamente utilizadas no mercado.

Também foi possível compreender o impacto do design centrado no usuário, evidenciado na construção de interfaces intuitivas e funcionais com auxílio de ferramentas como o Figma. Além disso, as práticas de DevOps e testes automatizados demonstraram a relevância da automação e da integração contínua como estratégias para manter a qualidade e a agilidade em projetos reais.

Apesar dos avanços promovidos pela aplicação dos princípios ágeis, alguns desafios se mostraram recorrentes ao longo da experiência:

- Conciliar múltiplas ferramentas em um fluxo eficiente, exigindo conhecimento técnico e organização;
- Garantir a manutenção e a escalabilidade das soluções, o que demanda disciplina na escrita do código e boas práticas de arquitetura;
- Promover qualidade contínua, por meio de testes automatizados e processos que assegurem confiabilidade sem comprometer a velocidade de entrega.

Portanto, mais do que um conjunto de disciplinas isoladas, o curso representou uma formação voltada para a prática ágil e colaborativa do desenvolvimento de software. A integração entre teoria e prática, aliada ao uso de ferramentas modernas e à valorização da experiência do usuário, resultou em uma base sólida para enfrentar os desafios do mercado e para contribuir com projetos que priorizem valor, adaptabilidade e qualidade.

REFERÊNCIAS

AMBLER, Scott W. Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. New York: Wiley, 2002.

ANDERSON, David J. Kanban: Successful Evolutionary Change for Your Technology Business. Sequim: Blue Hole Press, 2010.

ANGULAR TEAM. Angular Documentation. Disponível em: https://angular.io/docs. Acesso em: 18 jul. 2025.

BECK, Kent. Extreme Programming Explained: Embrace Change. 2nd ed. Boston: Addison-Wesley, 2004.

BECK, Kent. Test-Driven Development: By Example. Boston: Addison-Wesley, 2003.

BECK, Kent et al. Manifesto Ágil para Desenvolvimento de Software. 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 18 jul. 2025.

BENNIS, Warren; NANUS, Burt. Leaders: Strategies for Taking Charge. New York: HarperCollins, 1985.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML – Guia do Usuário. 2. ed. Rio de Janeiro: Elsevier, 2005.

BURNS, Simon; MCGREGOR, Alex. Selenium WebDriver 4: New Features and Improvements. Birmingham: Packt Publishing, 2022.

CHACON, Scott; STRAUB, Ben. Pro Git. 2. ed. Berkeley: Apress, 2014. Disponível em: https://git-scm.com/book/en/v2. Acesso em: 18 jul. 2025.

CMMI INSTITUTE. CMMI for Development (CMMI-DEV). Version 2.0. Pittsburgh: CMMI Institute, 2018.

COHN, Mike. User Stories Applied: For Agile Software Development. Boston: Addison-Wesley, 2004.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Algoritmos: Teoria e Prática. 3. ed. Rio de Janeiro: Elsevier, 2012.

DATE, C. J. Introdução a Sistemas de Banco de Dados. 8. ed. Rio de Janeiro: LTC, 2004.

DOCKER, Inc. Docker Documentation. Disponível em: https://docs.docker.com/. Acesso em: 18 jul. 2025.

DUCKETT, Jon. HTML & CSS: Design and Build Websites. Indianapolis: John Wiley & Sons, 2011.

ECKEL, Bruce. Thinking in Java. 4. ed. Upper Saddle River: Prentice Hall, 2006.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 7. ed. São Paulo: Pearson, 2017.

FIGMA. Figma Help Center. Disponível em: https://help.figma.com. Acesso em: 18 jul. 2025.

FORSGREN, Nicole; HUMBLE, Jez; KIM, Gene. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. 1. ed. Portland: IT Revolution Press, 2018.

FREEMAN, Adam; SANDERSON, Steven. Pro Angular. 4. ed. New York: Apress, 2022.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2007.

GAMMELGAARD, Boni. Practical Unit Testing with JUnit and Mockito. New York: Manning Publications, 2017.

GIT. Git Documentation. Disponível em: https://git-scm.com/doc. Acesso em: 18 jul. 2025.

GITLAB. GitLab Documentation. Disponível em: https://docs.gitlab.com/. Acesso em: 18 jul. 2025.

HEJLSBERG, Anders; ROSEN, Steve; KUDER, Luke; CELINE, Mohamed. Programming TypeScript. 2. ed. Sebastopol: O'Reilly Media, 2022.

HIGHTOWER, Kelsey; BURNS, Brendan; BEDA, Joe. Kubernetes: Up and Running. 2. ed. Sebastopol: O'Reilly Media, 2019.

HIPP, D. et al. SQLite Documentation. Disponível em: https://www.sqlite.org/docs.html. Acesso em: 18 jul. 2025.

HUMBLE, Jez; FARLEY, David. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston: Addison-Wesley, 2011.

JOHNSON, Craig Walls. Spring Boot in Action. Shelter Island: Manning Publications, 2016.

KACZMARZYK, Tomek. Mockito Cookbook. Birmingham: Packt Publishing, 2014.

KNAPP, Jake. Sprint: O método usado no Google para testar e aplicar novas ideias em apenas cinco dias. Rio de Janeiro: Intrínseca, 2016.

MARTIN, Robert C. Agile Software Development: Principles, Patterns, and Practices. Upper Saddle River: Prentice Hall, 2002.

MARTIN, Robert C. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River: Prentice Hall, 2009.

MESZAROS, Gerard. xUnit Test Patterns: Refactoring Test Code. Upper Saddle River: Addison-Wesley, 2007.

MERKEL, Dirk. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal, v. 2014, n. 239, p. 2, 2014.

MICROSOFT CORPORATION. Microsoft Edge Developer Documentation. Disponível em: https://learn.microsoft.com/microsoft-edge/. Acesso em: 18 jul. 2025.

MICROSOFT CORPORATION. Playwright Documentation. Disponível em: https://playwright.dev/docs. Acesso em: 18 jul. 2025.

MORRIS, Kief; SMITH, Mike. Infrastructure as Code: Managing Servers in the Cloud. 2. ed. Sebastopol: O'Reilly Media, 2020.

MUNSHI, Zeeshan. DevSecOps: Integrating Security into DevOps. Berkeley: Apress, 2021.

ORACLE CORPORATION. JDBC™ API Specification. Disponível em: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/. Acesso em: 18 jul. 2025.

ORACLE CORPORATION. MySQL 8.0 Reference Manual. Disponível em: https://dev.mysql.com/doc/. Acesso em: 18 jul. 2025.

ORACLE. Java Platform Documentation. Disponível em: https://docs.oracle.com/java/. Acesso em: 18 jul. 2025.

PIVOTAL SOFTWARE. Spring Boot Reference Guide. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 18 jul. 2025.

POPPENDIECK, Mary; POPPENDIECK, Tom. Lean Software Development: An Agile Toolkit. Boston: Addison-Wesley, 2003.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL Documentation. Disponível em: https://www.postgresql.org/docs/. Acesso em: 18 jul. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. The Scrum Guide: The Definitive Guide to Scrum – The Rules of the Game. Scrum.org, 2020. Disponível em: https://scrumguides.org/. Acesso em: 18 jul. 2025.

SHORE, James; WARDEN, Shane. The Art of Agile Development. 2nd ed. O'Reilly Media, 2021.

SIERRA, Kathy; BATES, Bert. Use a Cabeça! Java. 2. ed. Porto Alegre: Bookman, 2009.

SOFTEX. MPS.BR: Guia Geral. Brasília: SOFTEX, 2021. Disponível em: https://softex.br/mpsbr/. Acesso em: 18 jul. 2025.

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