UNIVERSIDADE FEDERAL DO PARANÁ

JEAN CARLOS DE CARVALHO COSTA

MEMORIAL DE PROJETOS: INTEGRAÇÃO MULTIDISCIPLINAR NO DESENVOLVIMENTO ÁGIL DE SOFTWARE – DA MODELAGEM À ENTREGA CONTÍNUA

CURITIBA
2025

JEAN CARLOS DE CARVALHO COSTA

MEMORIAL DE PROJETOS: INTEGRAÇÃO MULTIDISCIPLINAR NO DESENVOLVIMENTO ÁGIL DE SOFTWARE – DA MODELAGEM À ENTREGA CONTÍNUA

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..

Orientadora: Profª Dra. Rafaela Mantovani Fontana

CURITIBA
2025

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 **JEAN CARLOS DE CARVALHO COSTA**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO MULTIDISCIPLINAR NO DESENVOLVIMENTO ÁGIL DE SOFTWARE DA MODELAGEM À ENTREGA CONTÍNUA**, 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, 20 de Outubro de 2025.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

O presente memorial foi elaborado com o propósito de integrar e refletir sobre os projetos desenvolvidos ao longo da especialização em Desenvolvimento Ágil de Software, demonstrando como a formação multidisciplinar possibilitou a aplicação prática de princípios e metodologias ágeis no ciclo de desenvolvimento de software. A principal motivação este trabalho consiste na distância ainda existente entre o ensino tradicional de engenharia de software e as práticas contemporâneas de desenvolvimento ágil, baseadas em colaboração contínua, integração de equipes e automação de processos. O objetivo deste memorial é apresentar de forma sintética os conhecimentos adquiridos e as experiências práticas obtidas nas disciplinas que compõem a especialização, evidenciando como cada uma delas contribuiu para a consolidação de uma visão integrada do desenvolvimento ágil de software. Foram estudadas e aplicadas metodologias como Scrum, Kanban e Extreme Programming (XP), além de práticas de modelagem ágil, gerenciamento de projetos, desenvolvimento web e mobile, banco de dados, testes automatizados, experiência do usuário (UX) e DevOps. A partir dessas experiências, buscou-se compreender como a integração entre planejamento, codificação, testes e implantação contínua forma um fluxo de desenvolvimento coeso, colaborativo e eficiente. A vivência prática dos projetos permitiu internalizar os valores centrais do desenvolvimento ágil, tais como comunicação efetiva, entrega incremental e adaptação contínua. A integração entre disciplinas mostrou-se essencial para compreender o papel de cada etapa no ciclo de vida do software. A modelagem ágil proporcionou clareza na definição de requisitos; o desenvolvimento web e mobile evidenciou a importância da arquitetura modular e escalável; as disciplinas de testes e DevOps demonstraram o valor da automação e da entrega contínua; e UX reforçou a centralidade do usuário como foco das decisões de projeto. Essa abordagem integrada possibilitou compreender o desenvolvimento de software como um processo iterativo e colaborativo, sustentado por práticas técnicas e de gestão que garantem qualidade e agilidade. De forma geral, o memorial evidencia que o aprendizado adquirido ao longo da especialização permitiu consolidar uma compreensão abrangente sobre como a combinação entre metodologias, ferramentas e cultura ágil potencializa a eficiência das equipes e a entrega de valor ao cliente. As lições aprendidas demonstram que a integração entre modelagem, programação, testes e automação é fundamental para o desenvolvimento contínuo e adaptativo, características essenciais da engenharia de software. Assim, este trabalho reforça a importância da formação prática e interdisciplinar na consolidação das competências profissionais necessárias para atuar em ambientes ágeis e dinâmicos de desenvolvimento de software.

Palavras-chaves: desenvolvimento ágil de software; integração multidisciplinar; devops; testes automatizados.

ABSTRACT

This memorial was developed to integrate and reflect upon the projects carried out throughout the specialization in Agile Software Development, demonstrating how the multidisciplinary training enabled the practical application of agile principles and methodologies within the software development lifecycle. The main motivation for this work lies in the existing gap between traditional software engineering education and contemporary agile development practices, which are grounded in continuous collaboration, team integration, and process automation. The purpose of this memorial is to concisely present the knowledge acquired and the practical experiences gained across the specialization's courses, highlighting how each contributed to building an integrated understanding of agile software development. Throughout the program, methodologies such as Scrum, Kanban, and Extreme Programming (XP) were studied and applied, along with practices in agile modeling, project management, web and mobile development, databases, automated testing, user experience (UX), and DevOps. These experiences fostered an understanding of how the integration of planning, coding, testing, and continuous deployment forms a cohesive, collaborative, and efficient development flow. The hands-on project experiences reinforced core agile values such as effective communication, incremental delivery, and continuous adaptation. The integration across disciplines proved essential to understanding the role of each stage in the software lifecycle. Agile modeling provided clarity in defining requirements; web and mobile development emphasized the importance of modular and scalable architectures; testing and DevOps practices demonstrated the value of automation and continuous delivery; and UX underscored the centrality of the user in design decisions. This integrated approach enabled a comprehensive understanding of software development as an iterative and collaborative process supported by both technical and managerial practices that ensure quality and agility. Overall, this memorial demonstrates that the learning outcomes from the specialization fostered a broad comprehension of how the combination of methodologies, tools, and agile culture enhances team efficiency and value delivery to clients. The lessons learned emphasize that the integration of modeling, programming, testing, and automation is fundamental to continuous and adaptive development—key characteristics of modern software engineering. Thus, this work reinforces the importance of practical and interdisciplinary training in developing the professional competencies required to operate effectively in agile and dynamic software development environments.

Keywords: agile software development; multidisciplinary integration; devops; automated testing.

SUMÁRIO

1 PARECER TÉCNICO

Este parecer técnico apresenta uma análise integrada dos projetos desenvolvidos ao longo da especialização em Desenvolvimento Ágil de Software, evidenciando como as disciplinas se articularam para promover a prática de métodos, técnicas e ferramentas que caracterizam o desenvolvimento ágil. O objetivo é descrever, de forma lógica e coerente, os trabalhos realizados, indicando seus vínculos com as diferentes disciplinas do curso, e refletir sobre como a integração entre eles possibilitou vivenciar um processo de desenvolvimento próximo ao praticado em ambientes profissionais.

Durante o percurso formativo, foram realizados projetos que contemplaram desde a modelagem e análise de requisitos até a implementação de aplicações *web* e *mobile*, passando pelo gerenciamento ágil de projetos, integração de bancos de dados, práticas de testes automatizados e implantação em ambientes de infraestrutura com suporte a DevOps (Canjungo *et al.*, 2021). Essa diversidade de projetos permitiu experimentar, em um ambiente controlado, todas as etapas de um ciclo de desenvolvimento iterativo e incremental, reforçando a importância de práticas como planejamento adaptativo, colaboração entre áreas e entrega contínua (Novaes *et al.*, 2025).

A integração conceitual ficou evidente já nos primeiros projetos. A modelagem ágil de software, fundamentada em requisitos enxutos e artefatos como casos de uso, histórias de usuário e diagramas UML, serviu como base para orientar a implementação de funcionalidades nas disciplinas de introdução à programação e desenvolvimento *web* 1 e 2 (Nascimento, 2023). O refinamento da modelagem, aliado ao uso de princípios do *Domain Driven Design*, conectou-se diretamente às práticas de banco de dados, que garantiram a consistência das informações e o suporte às aplicações desenvolvidas (Rapôso *et al.*, 2025).

Essa integração se consolidou na prática quando os projetos passaram a incorporar testes automatizados e *pipelines* de integração contínua. A disciplina de Aspectos Ágeis de Programação enfatizou práticas como *Clean Code*, Refatoração e *Test-Driven Development* (TDD), que foram aplicadas de forma articulada com a disciplina de Testes Automatizados, em que ferramentas como *JUnit* e *Selenium* permitiram validar funcionalidades e assegurar a qualidade das entregas. A disciplina de Infraestrutura e *DevOps* complementou esse ciclo, ao possibilitar a criação de *pipelines* de CI/CD com automação de *builds*, testes e *deploys*, aproximando os projetos acadêmicos de um fluxo real de entrega contínua.

No contexto do gerenciamento, as disciplinas de Métodos Ágeis (MADS) e Gerenciamento Ágil de Projetos (GAP1 e GAP2) forneceram a estrutura necessária para organizar atividades, priorizar requisitos e acompanhar o progresso (Costa *et al.*, 2025). A utilização de *frameworks* de metodologia ágil como *Scrum*, *Kanban* e práticas como *Lean Inception* e *Design Sprint* possibilitaram experimentar, na prática, a dinâmica

de equipes ágeis, promovendo visibilidade, adaptação e foco em valor (Silva, 2024). Essas práticas se mostraram essenciais para a coordenação dos demais projetos, funcionando como eixo central de integração entre disciplinas (Brólio, 2024; Moraes, 2021).

Apesar dos avanços obtidos, alguns desafios foram notórios. A adaptação de ferramentas ao contexto acadêmico exigiu simplificações, como o uso de versões reduzidas de quadros *Kanban* ou *pipelines*, o que limita parcialmente a complexidade observada em ambientes corporativos. Outro ponto desafiador foi a necessidade de alinhar prazos curtos às entregas de qualidade, exigindo disciplina no planejamento e comprometimento constante. Esses aspectos refletem desafios reais enfrentados em equipes de desenvolvimento ágil, reforçando o valor da experiência acadêmica como preparação para a prática profissional.

Os projetos desenvolvidos demonstraram a relevância de integrar modelagem, programação, qualidade e automação no desenvolvimento ágil de software. A prática multidisciplinar vivenciada no curso fortaleceu competências essenciais como colaboração, pensamento sistêmico e adaptação contínua.

2 DISCIPLINA: MADS - MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

A disciplina de Métodos Ágeis para Desenvolvimento de Software (MADS) teve como objetivo introduzir os fundamentos das metodologias ágeis, destacando práticas como Scrum, Kanban e Extreme Programming. Esse conhecimento serviu como base para os demais projetos realizados ao longo do curso, uma vez que estabeleceu os princípios de iteração, colaboração e entrega contínua de valor.

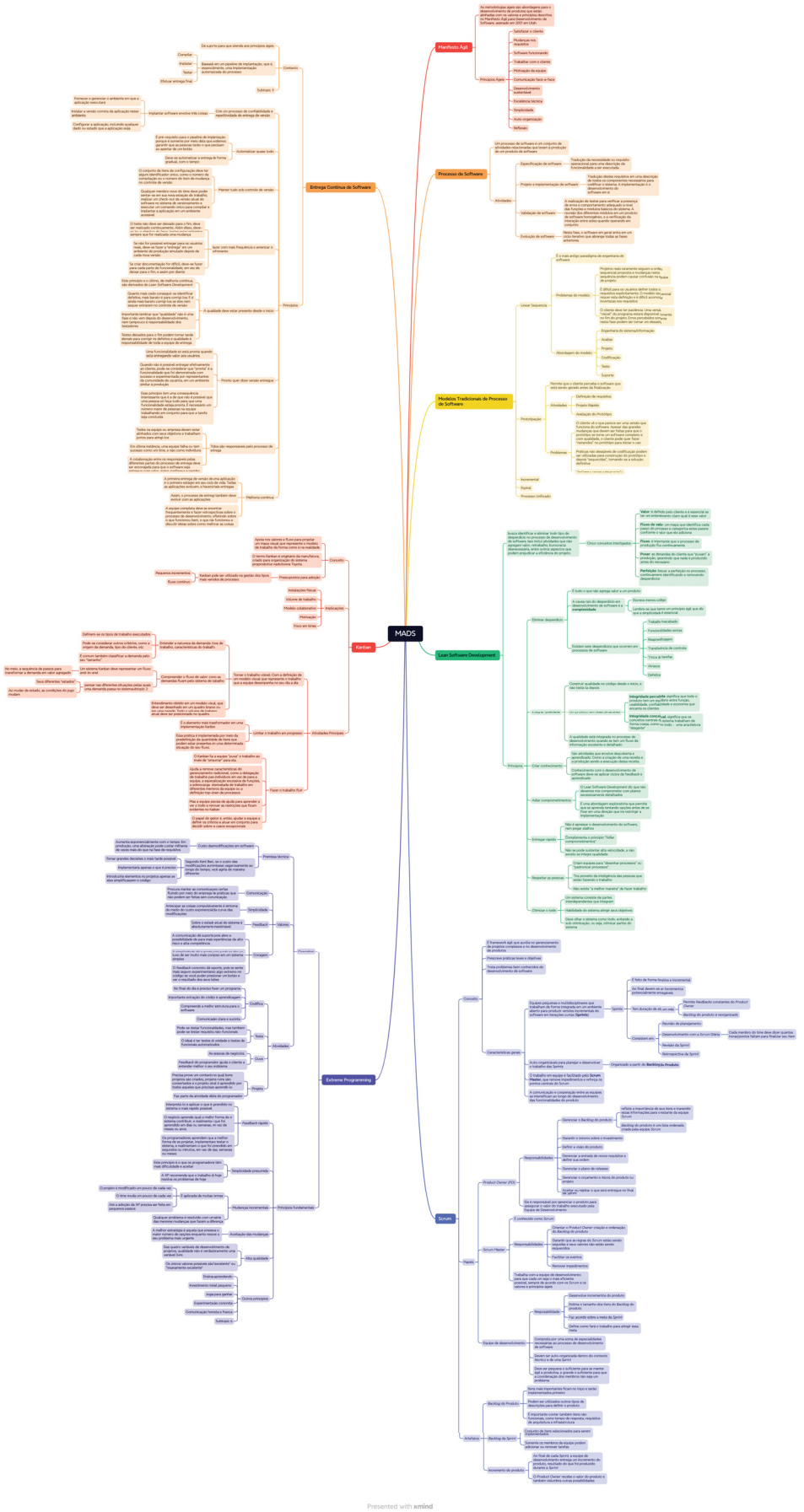
No âmbito desta disciplina, foi desenvolvido um projeto que consistiu na aplicação de práticas ágeis em um cenário de desenvolvimento de software. Este trabalho foi desenvolvido para compreensão dos principais conceitos do processo de software, cuja a compreensão é importante para o desenvolvimento ágil de software. Conforme Oliveira (2014), melhora a produtividade dos processos de desenvolvimento de software, para conseguir atingir uma qualidade ao produto.

A experiência proporcionou uma compreensão prática de como os métodos ágeis podem ser aplicados para organizar equipes e otimizar a entrega de funcionalidades. Além disso, permitiu vivenciar a adaptação do planejamento frente às mudanças de requisitos, característica essencial de processos ágeis.

Este trabalho teve relação com as disciplinas de Gerenciamento Ágil de Projetos, que aprofundaram o uso de ferramentas de gestão, e com Modelagem Ágil de Software, que forneceu subsídios para a priorização e detalhamento de requisitos.

2.1 ARTEFATOS DO PROJETO

FIGURA 1 – Mapa mental para representação dos conceitos principais de processos de software.



Fonte: Autor (2025).

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

A disciplina de Modelagem Ágil de Software teve como objetivo principal demonstrar a importância de alinhar a modelagem à filosofia dos métodos ágeis, evitando a criação de grandes documentos e priorizando modelos enxutos, colaborativos e úteis ao desenvolvimento Prikladnicki, Willi e Milani (2014). Foram revisados conceitos fundamentais de especificação de requisitos, casos de uso e UML, adaptando-os ao contexto ágil, além da introdução às histórias de usuário como artefato essencial para comunicação entre cliente e equipe. Também foram explorados princípios do *Domain Driven Design* (DDD), que reforçam a necessidade de compreender o domínio de negócio para guiar a construção do software. Dessa forma, a disciplina mostrou que a modelagem não é descartada em projetos ágeis, mas sim conduzida de forma objetiva, prática e integrada ao ciclo de desenvolvimento.

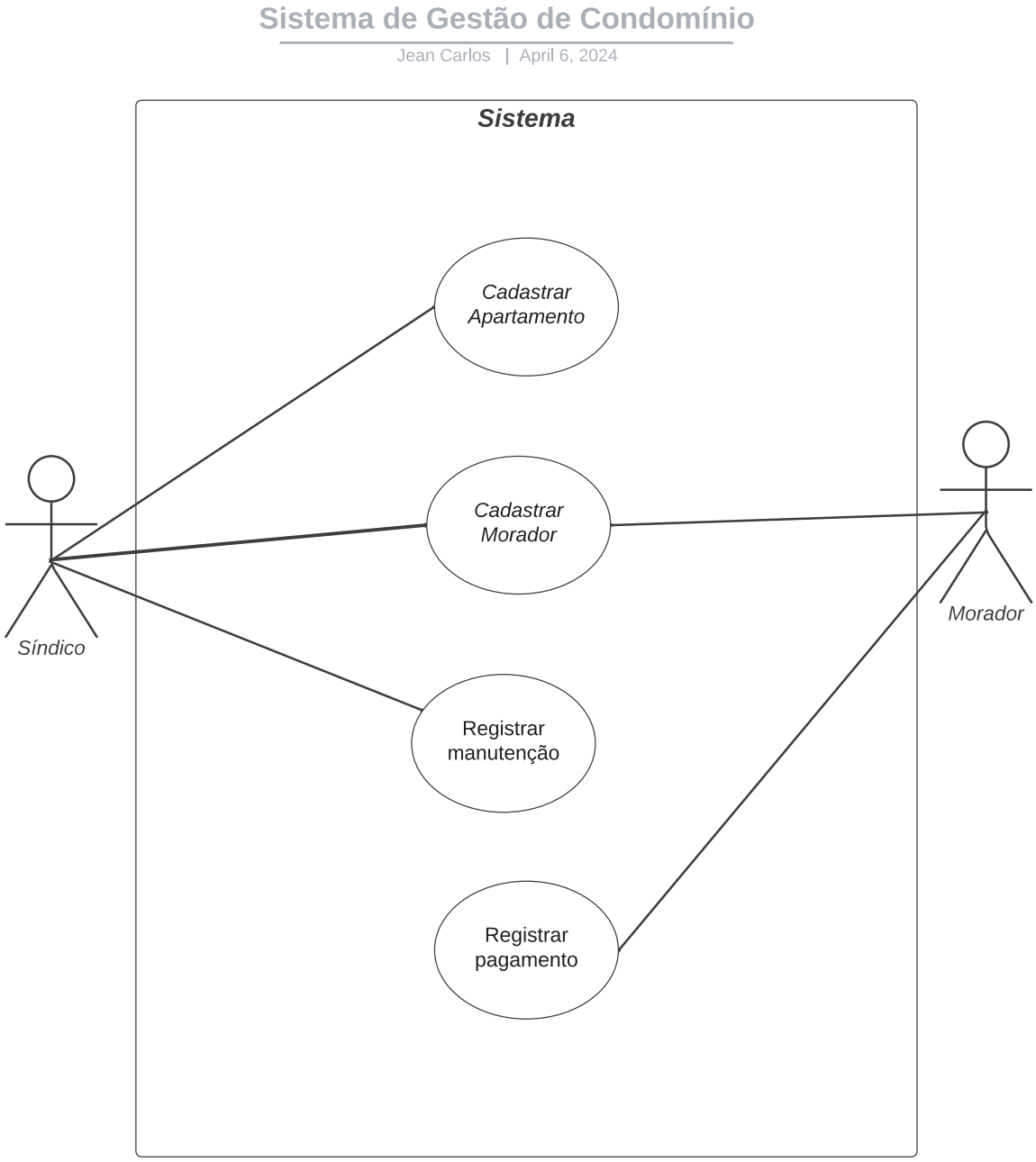
Os trabalhos desenvolvidos são apresentados de forma unificada, cujo é o desenvolvimento de um sistema de gestão de condomínio, abrangendo tanto a disciplina de MAG1 quanto a de MAG2. No MAG1, foram criados os diagramas de caso de uso dos níveis 1 e 2, que apresentam as interações do sistema com os usuários e identificam os responsáveis pelas funções dentro do sistema. Os personas considerados são o Síndico e o Morador. Além disso, foram elaboradas as histórias de usuário para documentar o que será repassado ao desenvolvedor. Em seguida, foram produzidos os *templates* de telas, com base nas histórias de usuário, para ilustrar como será a interação do sistema com o usuário. Por fim, utilizando o programa *Astah UML*, foram implementados os diagramas de classes e de sequência: o diagrama de classes mostra e descreve a estrutura estática de um sistema por meio de uma representação visual de suas classes, atributos (dados) e operações (métodos), bem como dos relacionamentos entre elas; já o diagrama de sequência mostra a interação entre diferentes objetos ou participantes de um sistema ao longo do tempo. Dessa forma, o conjunto de artefatos produzidos garante uma visão consistente do sistema, desde os requisitos iniciais até a modelagem estrutural e comportamental.

O trabalho contribuiu para o desenvolvimento de outras disciplinas, como Introdução à Programação, apresentada no Capítulo 5, e Gerenciamento de Projetos de Software, no Capítulo 4. O trabalho, assim como a disciplina, foi importante para o desenvolvimento dos projetos, reforçando a compreensão de que, na formação do profissional em Engenharia de Software e em gerenciamento de projetos, é fundamental conhecer as metodologias ágeis e seus principais princípios.

3.1 ARTEFATOS DO PROJETO

3.2 DIAGRAMA DE CASO DE USO – NÍVEL 1

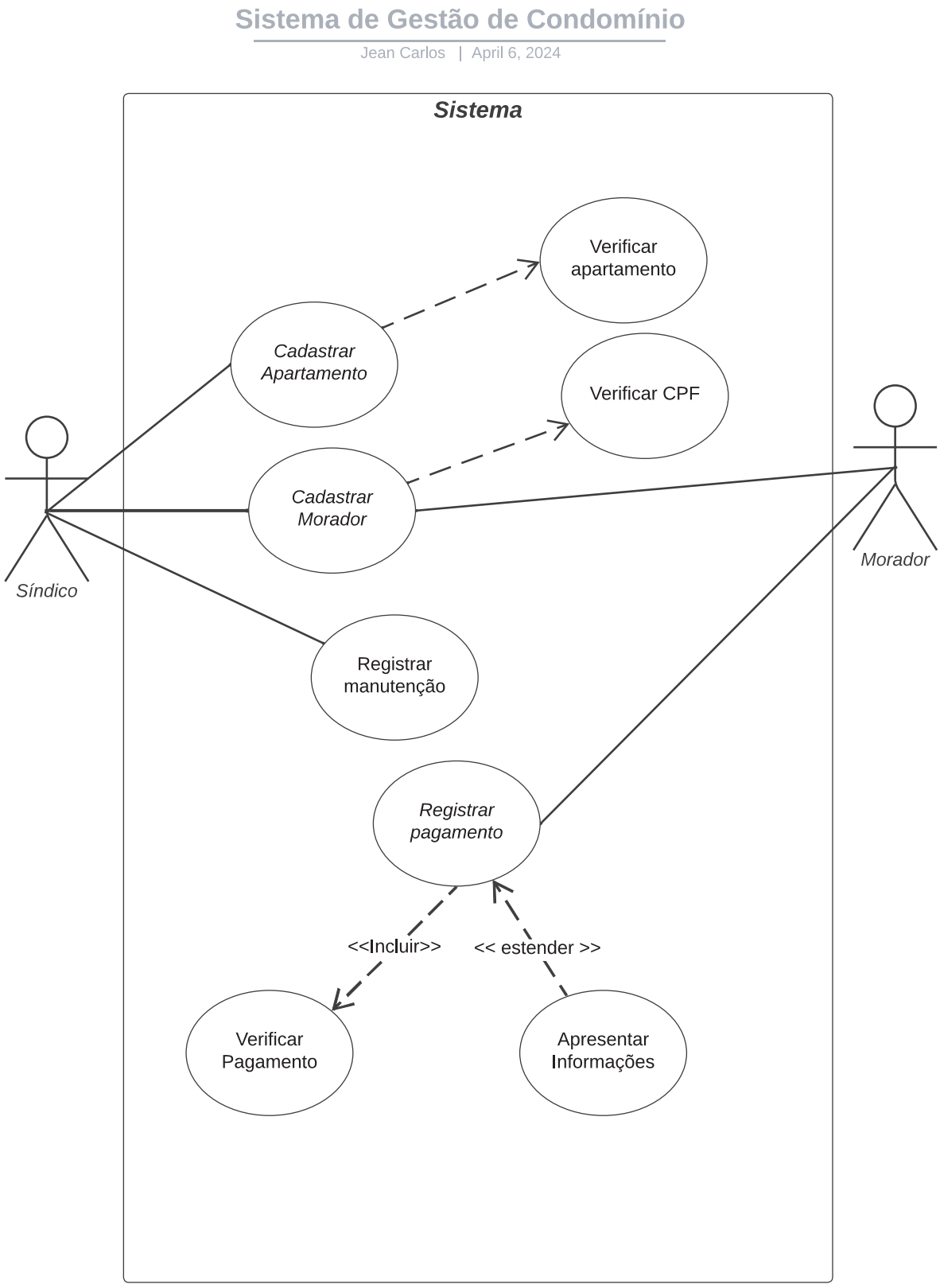
FIGURA 2 – Diagrama de caso de nível 1 para especificar os requisitos do sistema.



Fonte: Autor (2025).

3.3 DIAGRAMA DE CASO DE USO – NÍVEL 2

FIGURA 3 – Diagrama de caso de uso nível 2 para corresponder a interação de telas conforma elaborado no DCU nível 1.



Fonte: Autor (2025).

3.4 HISTÓRIA DE USUÁRIO

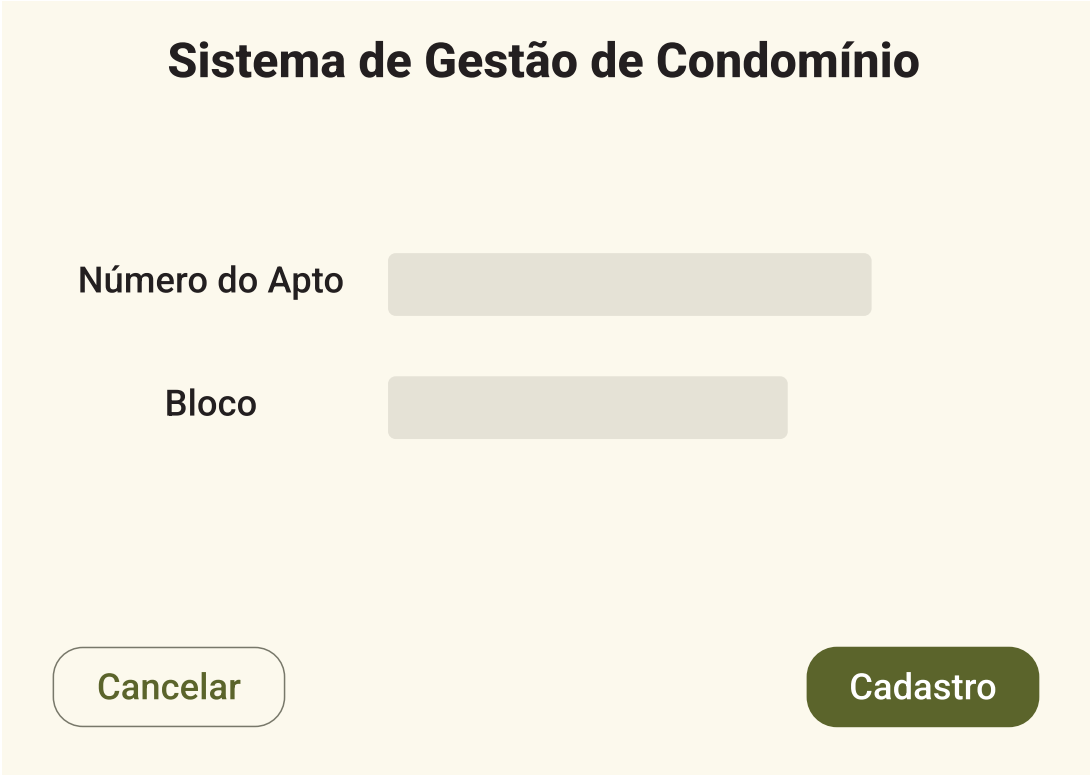
3.4.1 HU001 – Cadastrar apartamento

SENDO	o Sindico
QUERO	Cadastrar os apartamentos
PARA	regularização da hospedagem

Fonte: Autor (2025).

3.4.2 Desenha da Tela

FIGURA 4 – Tela inicial para o sistema de Gestão.



Fonte: Autor (2025).

3.4.3 Lista de Critérios de Aceitação

1. Deve preencher o número do apartamento;
2. Deve preencher o bloco do apartamento;
3. Deve permitir o cadastro do apartamento;
4. Deve permitir cancelar o cadastro do apartamento;
5. O sistema deve validar se o número do apartamento já existe no cadastro.

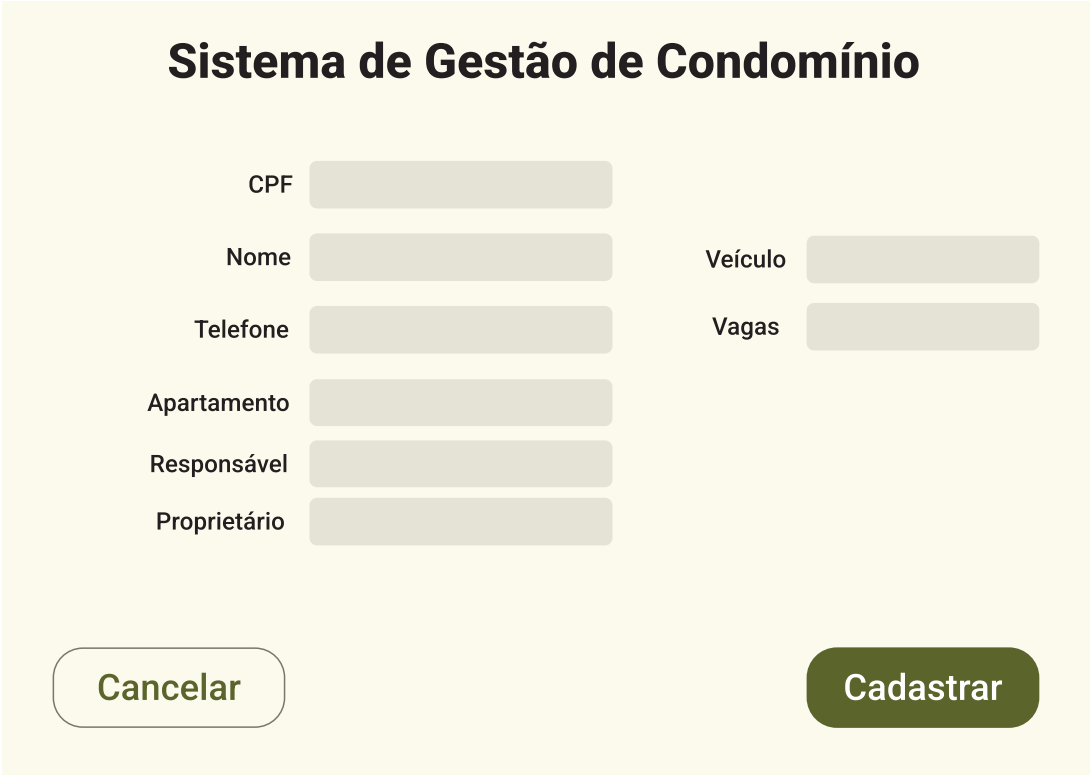
3.4.4 HU002 – Cadastrar Morador

SENDO	o Sindico
QUERO	Cadastrar um morador
PARA	cadastro de novos inquilinos

Fonte: Autor (2025).

3.4.5 Desenho de tela

FIGURA 5 – Tela para cadastrar novos usuários no sistema.



Fonte: Autor (2025).

3.4.6 Lista de Critérios de Aceitação

1. Deve ser possível cadastrar o CPF, nome, telefone, apartamento, se é responsável (S/N) e se é proprietário (S/N).
2. O sistema deve verificar se o CPF já está cadastrado.
3. Os dados dos veículos dos moradores devem ser armazenados, junto com suas vagas de garagem.

3.4.7 HU003 – Registrar pagamento

SENDO	o Sindico
QUERO	registrar o pagamento do condomínio
PARA	manter controle sobre as finanças

3.4.8 Desenho de tela

FIGURA 6 – Demonstração de pagamento de inquilinos no sistema.

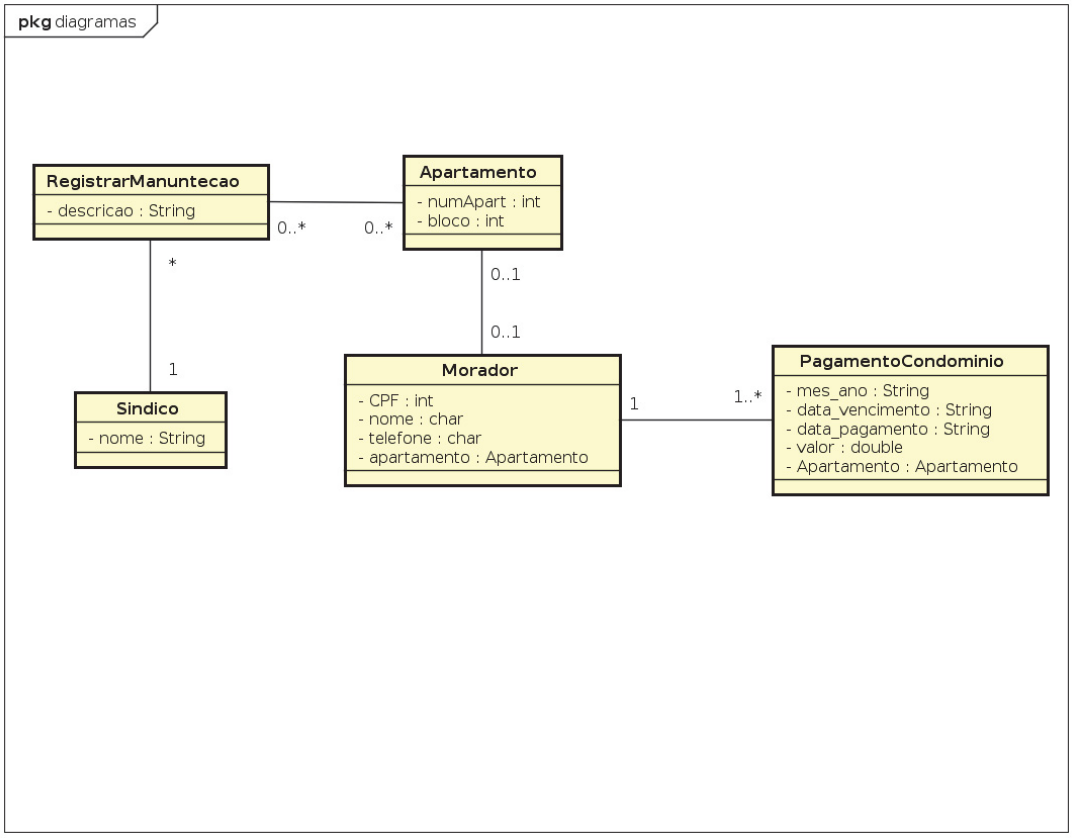
Fonte: Autor (2025).

3.4.9 Lista de Critérios de Aceitação

1. Deve ser possível registrar o pagamento informando o apartamento, mês/ano de referência, data de vencimento, data de pagamento e valor do condomínio.
2. O sistema deve apresentar o nome do morador responsável e o valor do condomínio ao digitar o número do apartamento.
3. O registro só pode ser feito se não houver valores anteriores vencidos.

3.4.10 Diagrama de Classe

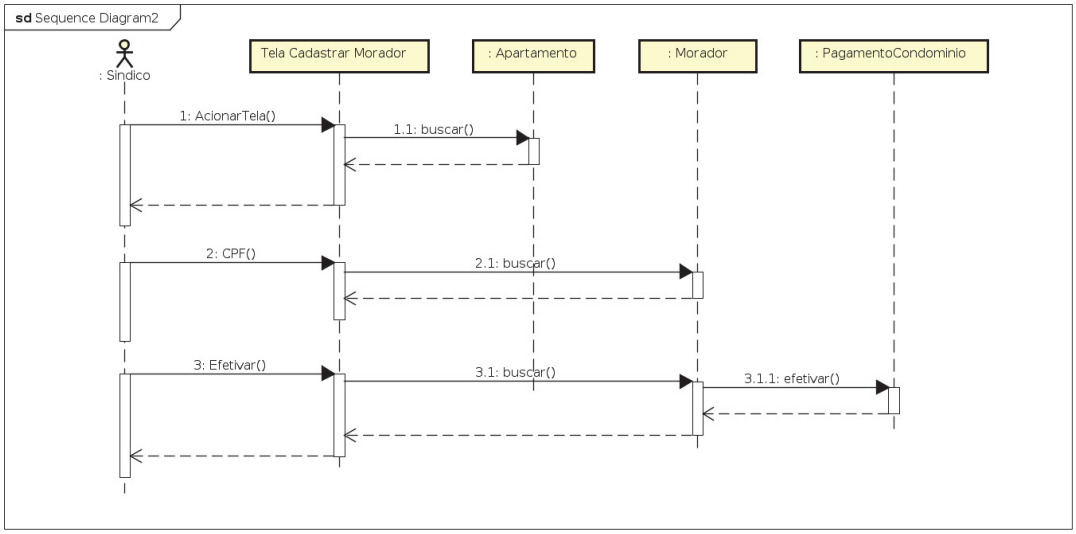
FIGURA 7 – Diagrama de classe do funcionamento do sistema.



Fonte: Autor (2025).

3.4.11 Diagrama de Sequência

FIGURA 8 – Diagrama de sequência do processo de ações dentro do sistema.



Fonte: Autor (2025).

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

As disciplinas de Gerenciamento Ágil de Projetos I e II (GAP1 e GAP2) tiveram como objetivo capacitar os alunos na aplicação de práticas de gestão de projetos voltadas ao desenvolvimento de software em ambientes ágeis. Enquanto a primeira disciplina focou nos fundamentos clássicos de gerenciamento, apoiados no PMBOK 6ª edição, abordando aspectos como iniciação, planejamento, riscos e *stakeholders*, a segunda disciplina avançou para a gestão visual, métricas ágeis e princípios do PMBOK 7ª edição, destacando a importância da adaptabilidade e da entrega de valor contínuo.

No decorrer das disciplinas, foi desenvolvido um projeto que aplicou metodologias híbridas, iniciando com a definição da visão do produto por meio de técnicas como *Design Thinking*, *Lean Inception* e *Design Sprint*, e evoluindo para a gestão das entregas utilizando *Scrum* e *Kanban*. O projeto contemplou a elaboração de canvas de planejamento, definição de *backlog*, priorização de requisitos, organização de sprints e acompanhamento de fluxo de trabalho em quadros visuais.

A realização desse projeto demonstrou a relevância de alinhar práticas tradicionais de gestão com métodos ágeis, favorecendo a organização das atividades, a priorização das demandas e o monitoramento contínuo das entregas. Além disso, evidenciou como o uso de métricas ágeis, como *lead time* e *throughput*, pode apoiar a tomada de decisão e a melhoria do processo.

As disciplinas GAP1 e GAP2 mostraram forte integração com MADS, ao apoiar a definição de metas de desenvolvimento; com Modelagem Ágil de *Software*, ao priorizar requisitos derivados das histórias de usuário; e com Testes Automatizados, ao possibilitar que os critérios de aceitação fossem planejados e monitorados ao longo das iterações. Essa integração evidenciou a gestão ágil como eixo estruturante dos demais projetos do curso.

4.1 ARTEFATOS DO PROJETO

FIGURA 9 – Plano de Release - Gerenciamento Ágil de Projetos I

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

Cálculo da Velocidade:

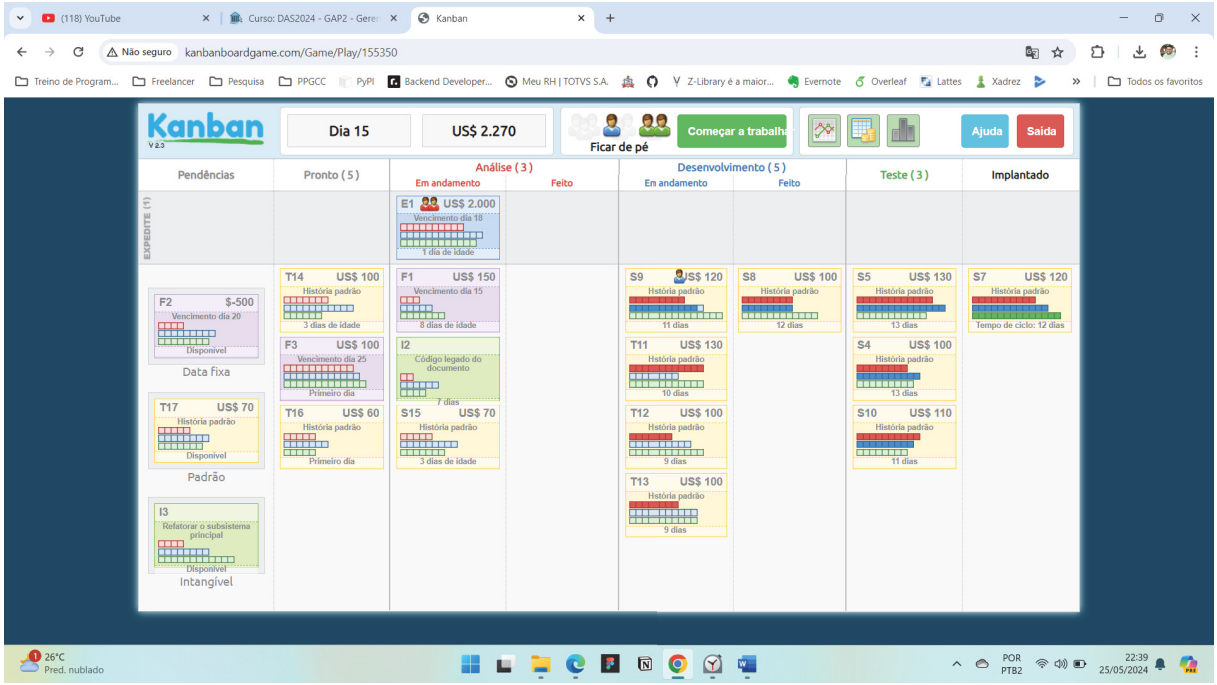
Horas disponíveis por dia: 4 horas		Tamanho da Sprint: 2 semanas	
Horas disponíveis por Sprint: 40 horas		Velocidade: 5	

Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4
Data Início: 03 de maio de 2024	Data Início: 20 de maio de 2024	Data Início: 04 de junho de 2024	Data Início: 18 de junho de 2024
Data Fim: 17 de maio de 2024	Data Fim: 03 de junho de 2024	Data Fim: 17 de junho de 2024	Data Fim: 01 de julho de 2024
<HU03> SENDO cliente QUERO agendar um horário de corte PARA não ter que esperar na fila ESTIMATIVA (3)	<HU07> SENDO Funcionário QUERO Poder marcar um cliente como atendido PARA Manter a agenda atualizada ESTIMATIVA (2)	<HU05> SENDO Cliente QUERO Ver o histórico dos meus cortes anteriores PARA Lembrar do estilo que eu gostei ESTIMATIVA (2)	<HU04> SENDO Funcionário QUERO Registrar os serviços prestados PARA Manter o histórico do cliente ESTIMATIVA (2)
<HU01> SENDO Funcionário QUERO Ver a agenda do dia PARA Organizar os atendimentos ESTIMATIVA (1)	<HU02> SENDO Cliente QUERO Visualizar os serviços oferecidos e seus preços PARA Escolher o serviço desejado ESTIMATIVA (3)	<HU08> SENDO Funcionário QUERO Gerar relatórios de atendimento PARA Avaliar o desempenho e identificar áreas de melhoria ESTIMATIVA (3)	<HU6> SENDO Cliente QUERO Poder cancelar um horário agendado PARA alterar os planos conforme necessário ESTIMATIVA (2)

Fonte: Autor (2025).

FIGURA 10 – Diagrama de fluxo cumulativo do jogo Kanban, apresentando a evolução das tarefas nas etapas de preparação, análise, desenvolvimento, teste e implantação ao longo do tempo.

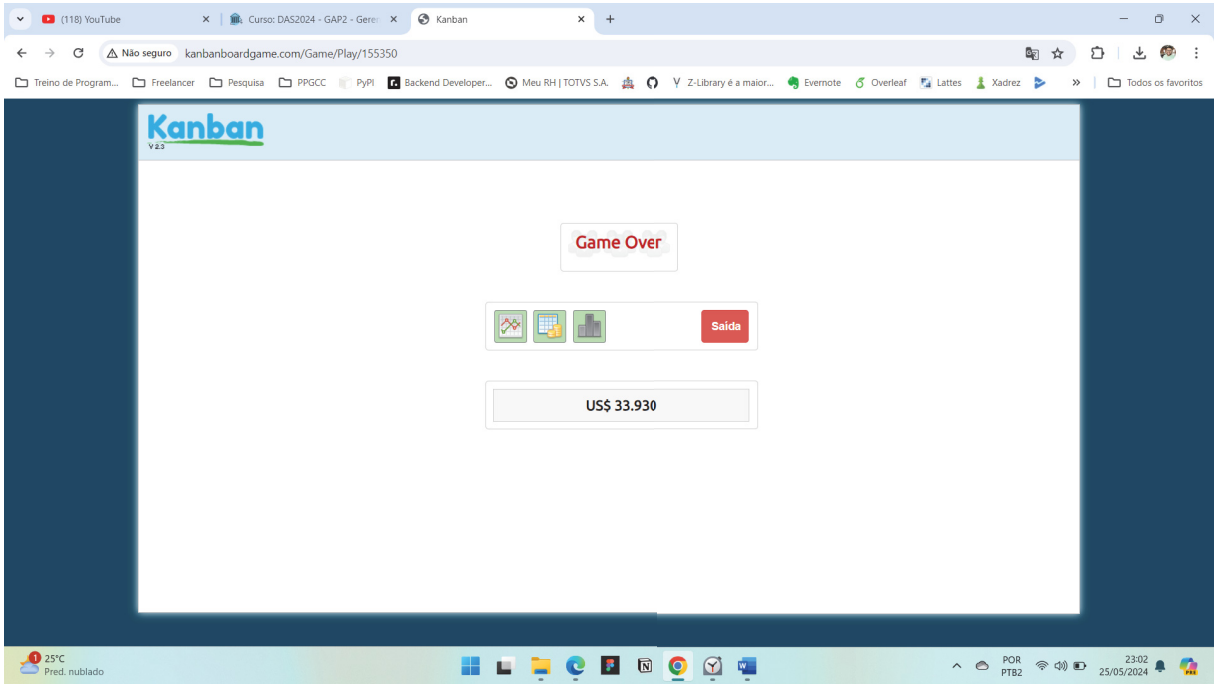


Fonte: Autor (2025).

FIGURA 11 – Tela de encerramento da simulação no jogo Kanban, indicando o valor total acumulado de US$ 33.930.

Fonte: Autor (2025).

FIGURA 12 – Visualização do quadro Kanban no dia 15 da simulação, com distribuição de tarefas entre as colunas: pendências, pronto, análise, desenvolvimento, teste e implantado.



Fonte: Autor (2025).

5 DISCIPLINA: INTRO — INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação teve como objetivo proporcionar os fundamentos essenciais para o desenvolvimento de algoritmos e programas de computador, servindo de base para todas as demais disciplinas do curso. Foram abordados conceitos como lógica de programação, variáveis, operadores, estruturas condicionais e de repetição, vetores, funções, programação orientada a objetos e integração com banco de dados. Essa formação inicial permitiu compreender a estrutura e o raciocínio lógico necessários para resolver problemas computacionais de forma sistemática.

No contexto da disciplina, foram propostos exercícios práticos e um projeto de *back-end*, ambos voltados à fixação da lógica de programação. Os exercícios contemplaram a criação de algoritmos para cálculo matemático, manipulação de vetores, processamento de dados e implementação de funções. Já o projeto final consistiu no desenvolvimento de uma aplicação em *Java*, que consolidou os principais conceitos trabalhados, abrangendo desde operações básicas de entrada e saída de dados até o uso de estruturas mais complexas, como repetição e modularização do código. O objetivo foi implementar o *back-end* de um sistema bancário simplificado, responsável pelo controle de clientes, contas correntes e contas de investimento. Para apoiar o desenvolvimento, foram fornecidos: (i) um diagrama de classes do sistema, (ii) um projeto no *NetBeans* contendo testes unitários, e (iii) um *script* DDL para criação das tabelas no *MySQL*. Além disso, casos de teste em *JUnit* foram utilizados para validar os algoritmos implementados. O projeto está disponível no repositório do GitHub[1].

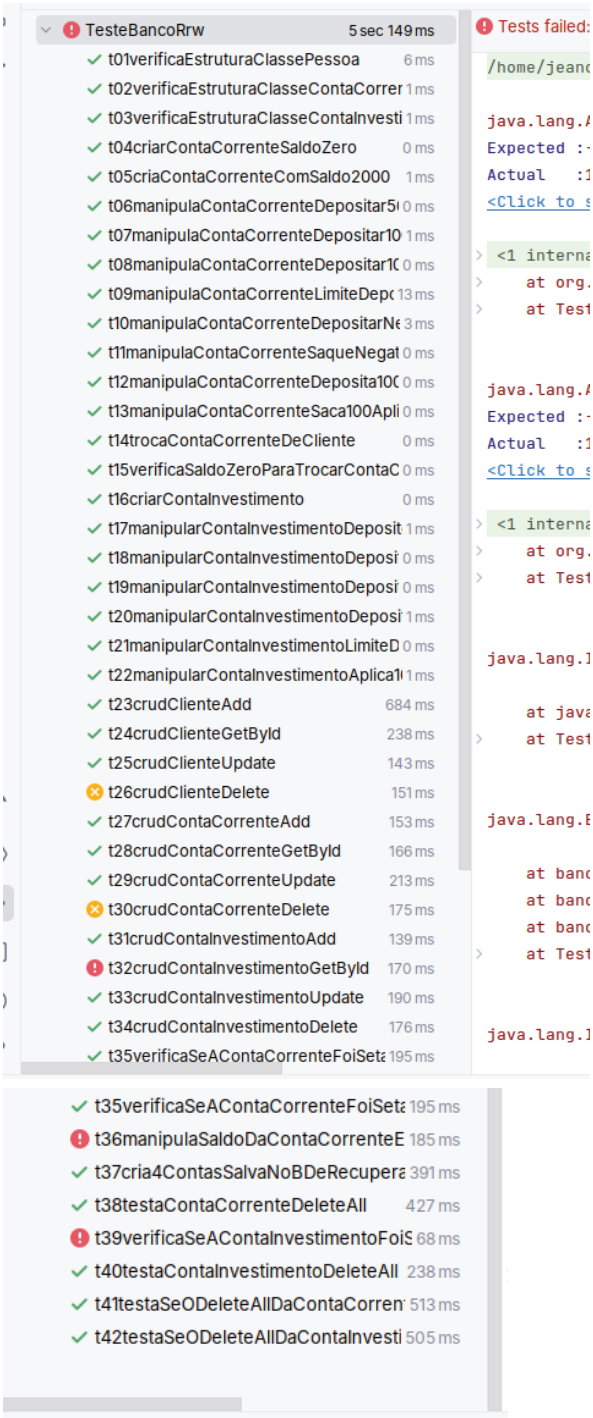
O aprendizado adquirido nesta disciplina foi fundamental para a aplicação das práticas ágeis, uma vez que forneceu a base técnica necessária para implementar funcionalidades de forma incremental. Ao dominar a lógica de programação, tornou-se possível compreender e desenvolver histórias de usuário em código executável, garantindo entregas contínuas e alinhadas com os princípios ágeis.

A disciplina de Introdução à Programação se integrou diretamente com Modelagem Ágil de Software, ao transformar diagramas e histórias de usuário em código funcional, e com Testes Automatizados, permitindo validar as soluções desenvolvidas. Também forneceu suporte para as práticas de Gerenciamento Ágil de Projetos, ao possibilitar que as atividades de programação fossem planejadas e executadas em *sprints*, conectando teoria e prática no desenvolvimento colaborativo.

5.1 ARTEFATOS DO PROJETO

[1] https://github.com/jeancarloscc/sistema-conta-corrente-trab-final

FIGURA 13 – Resultados dos testes unitários com *JUnit* como comprovação de trabalho realizado.



Fonte: Autor (2025).

6 DISCIPLINA: BD – BANCO DE DADOS

A disciplina de Banco de Dados teve como objetivo introduzir os fundamentos teóricos e práticos relacionados ao armazenamento, organização e manipulação de dados. Foram trabalhados conceitos essenciais de modelagem entidade-relacionamento, modelo relacional, normalização e álgebra relacional, servindo como base sólida para o entendimento de sistemas de gerenciamento de banco de dados (SGBDs) e para a aplicação de consultas em SQL.

O projeto desenvolvido na disciplina consistiu na criação de um banco de dados relacional a partir da modelagem de um domínio específico. Essa atividade envolveu a construção do diagrama entidade-relacionamento, a definição do esquema lógico no modelo relacional, a aplicação de regras de normalização e, por fim, a implementação em SQL utilizando comandos de definição (DDL) e manipulação de dados (DML).

A experiência mostrou que o domínio de banco de dados é essencial para projetos ágeis, uma vez que a correta modelagem e estruturação dos dados impactam diretamente na qualidade e desempenho das aplicações. Além disso, a capacidade de elaborar consultas eficientes e manter a integridade dos dados é um requisito fundamental para dar suporte a entregas incrementais e ao desenvolvimento iterativo.

A disciplina de Banco de Dados apresentou forte integração com disciplinas como Modelagem Ágil de Software, que forneceu requisitos e diagramas para posterior estruturação no modelo relacional, e Programação, que utilizou o banco implementado para persistência de dados nas aplicações desenvolvidas. Essa conexão evidenciou a importância da base de dados como pilar central no ciclo de desenvolvimento de software.

6.1 ARTEFATOS DO PROJETO

6.2 INTRODUÇÃO

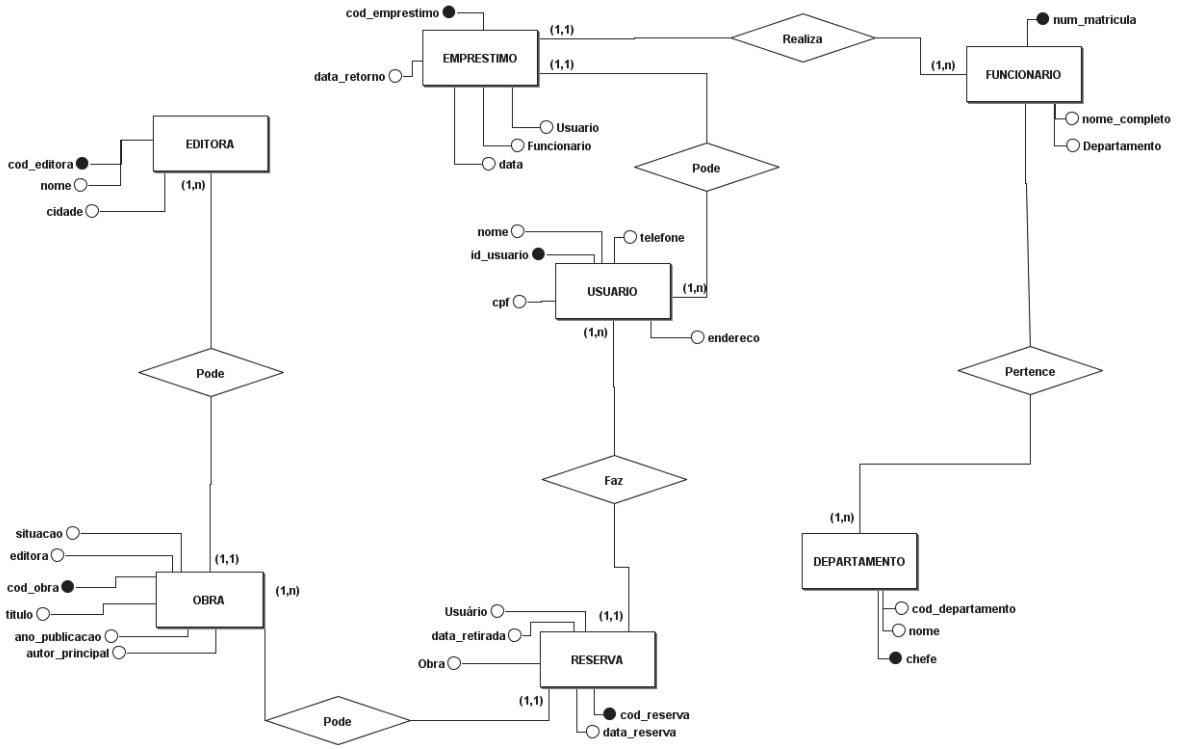
O objetivo deste trabalho é desenvolver os modelos conceitual e lógico de considerando o sistema de controle de biblioteca. Para o desenvolvimento dos modelos, é utilizado o software brModelo.

6.3 QUESTÃO 1

Na Figura 1 e 2 é apresentado o modelo conceitual, desenvolvido utilizando o modelo Entidade-Relacionamento e o modelo lógico com os diagramas de tabelas, os quais constam as entidades editora, obra, empréstimo, funcionário, usuário, departamento e reserva. Este modelo foi implementado utilizando o brModelo. Da mesma, no modelo lógico, contém as mesmas entidades, porém são apresentadas com tabelas,

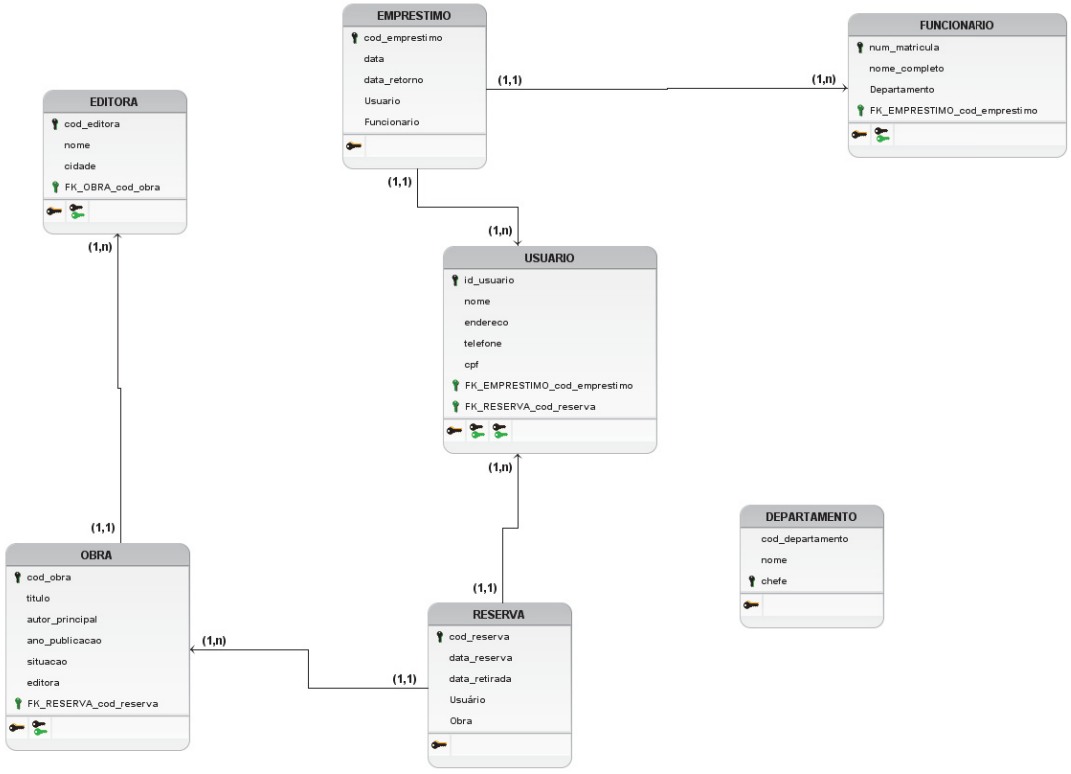
campos, chaves-primárias e estrangeiras e os relacionamentos.

FIGURA 14 – Modelo Conceitual.



Fonte: Autor (2025).

FIGURA 15 – Modelo lógico.

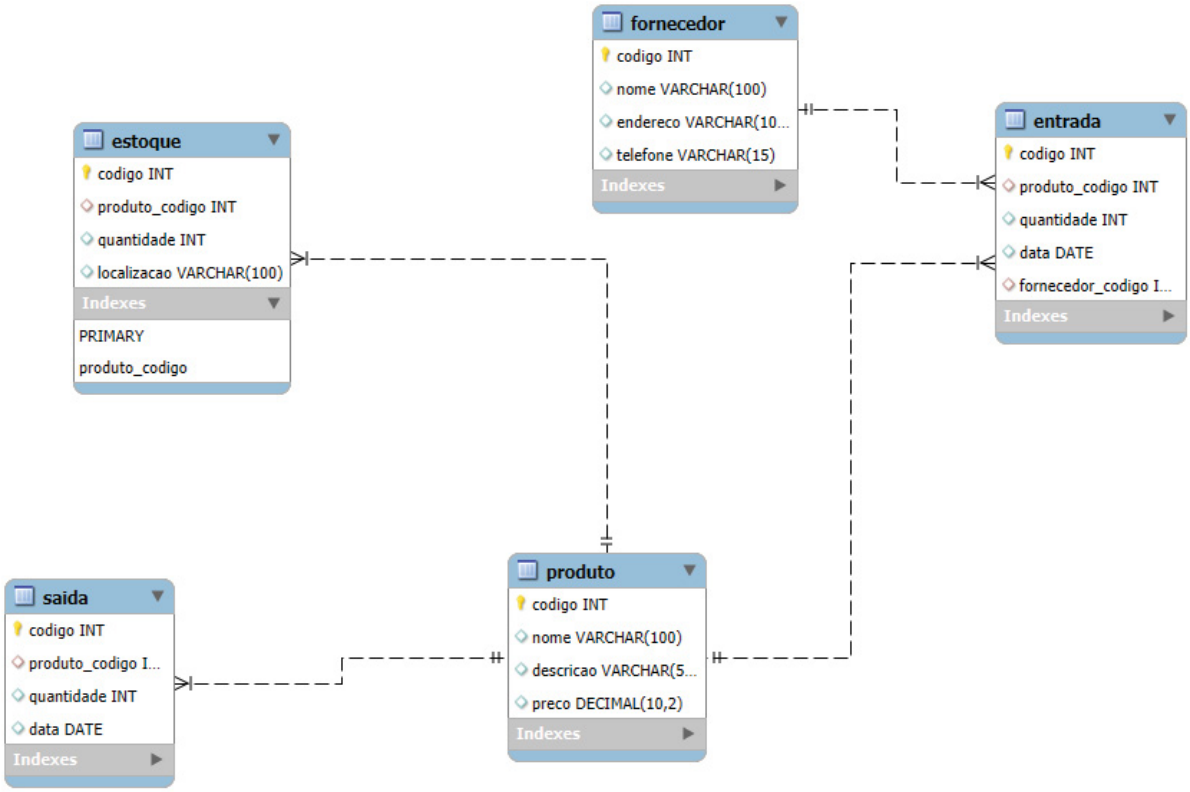


Fonte: Autor (2025).

6.4 QUESTÃO 2

O objetivo deste projeto é desenvolver um sistema de controle de estoque utilizando um banco de dados relacional. O sistema deve ser capaz de gerenciar informações sobre produtos, fornecedores, entradas e saídas de estoque, além de permitir a consulta de dados para análise e controle. Na Figura 16 é mostrado o modelo lógico, define a estrutura das tabelas e seus relacionamentos no banco de dados. A seguir, são apresentados os comandos SQL para criação das tabelas e a inserção de dados de exemplo.

FIGURA 16 – Modelo Lógico criado no MySQL Workbench.



Fonte: Autor (2025).

FIGURA 17 – Exemplo de criação do banco de dados estoque e os registros.

21	22:31:31	create database estoque	1 row(s) affected	0.015 sec
22	22:31:31	use estoque	0 row(s) affected	0.000 sec
23	22:31:31	CREATE TABLE Produto (codigo INT PRIMARY KEY, nome VARCHAR(100), descricao varchar(50), ...	0 row(s) affected	0.047 sec
24	22:31:31	CREATE TABLE Fornecedor (codigo INT PRIMARY KEY, nome VARCHAR(100), endereco varchar(1...	0 row(s) affected	0.031 sec
25	22:31:31	CREATE TABLE Estoque (codigo INT PRIMARY KEY, produto_codigo INT, quantidade INT, locali...	0 row(s) affected	0.078 sec
26	22:31:31	CREATE TABLE Entrada (codigo INT PRIMARY KEY, produto_codigo INT, quantidade INT, data ...	0 row(s) affected	0.094 sec
27	22:31:31	CREATE TABLE Saida (codigo INT PRIMARY KEY, produto_codigo INT, quantidade INT, data DA...	0 row(s) affected	0.094 sec
28	22:31:31	INSERT INTO Produto (codigo, nome, descricao, preco) VALUES (1, 'Produto A', 'Descrição do Produto A', 10...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
29	22:31:31	INSERT INTO Fornecedor (codigo, nome, endereco, telefone) VALUES (1, 'Fornecedor X', 'Endereço X', '111...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
30	22:31:31	INSERT INTO Estoque (codigo, produto_codigo, quantidade, localizacao) VALUES (1, 1, 100, 'A1'), (2, 2, 200...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
31	22:31:31	INSERT INTO Entrada (codigo, produto_codigo, quantidade, data, fornecedor_codigo) VALUES (1, 1, 50, '20...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
32	22:31:31	INSERT INTO Saida (codigo, produto_codigo, quantidade, data) VALUES (1, 1, 20, '2024-07-06'), (2, 2, 30, '2...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
33	22:31:41	SELECT * FROM estoque.entrada LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
34	22:31:45	SELECT * FROM estoque.estoque LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Result Grid | Filter Rows: | Edit: | Export/Impor

codigo	produto_codigo	quantidade	data	fornecedor_codigo
1	1	50	2024-07-01	1
2	2	150	2024-07-02	2
3	3	200	2024-07-03	3
4	4	250	2024-07-04	4
5	5	300	2024-07-05	5
NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

codigo	produto_codigo	quantidade	localizacao
1	1	100	A1
2	2	200	B1
3	3	300	C1
4	4	400	D1
5	5	500	E1
NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

codigo	nome	endereco	telefone
1	Fornecedor X	Endereço X	1111-1111
2	Fornecedor Y	Endereço Y	2222-2222
3	Fornecedor Z	Endereço Z	3333-3333
4	Fornecedor W	Endereço W	4444-4444
5	Fornecedor V	Endereço V	5555-5555
NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

codigo	nome	descricao	preco
1	Produto A	Descrição do Produto A	10.00
2	Produto B	Descrição do Produto B	20.00
3	Produto C	Descrição do Produto C	30.00
4	Produto D	Descrição do Produto D	40.00
5	Produto E	Descrição do Produto E	50.00
NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

codigo	produto_codigo	quantidade	data
1	1	20	2024-07-06
2	2	30	2024-07-07
3	3	40	2024-07-08
4	4	50	2024-07-09
5	5	60	2024-07-10
NULL	NULL	NULL	NULL

Fonte: Autor (2025).

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como foco principal apresentar práticas e técnicas que garantem a qualidade contínua do código em projetos de software desenvolvidos sob metodologias ágeis. Foram abordados conceitos de refatoração, código limpo e boas práticas de programação, com ênfase na importância da legibilidade e manutenibilidade do software para facilitar sua evolução ao longo do tempo.

O projeto da disciplina consistiu na aplicação de técnicas de *Test-Driven Development* (TDD) e na implementação de testes automatizados como parte integrante do ciclo de desenvolvimento. Também foram exploradas ferramentas de integração contínua (CI/CD), reforçando a ideia de que a automação dos testes e das entregas é essencial para sustentar a agilidade e reduzir riscos durante as iterações.

A experiência reforçou a compreensão de que a agilidade não se limita à gestão de projetos, mas depende fortemente de práticas de programação que sustentem entregas rápidas e de qualidade. A disciplina demonstrou que a aplicação de testes automatizados, aliada à refatoração contínua, reduz a incidência de erros e aumenta a confiabilidade das entregas, permitindo ciclos de *feedback* mais curtos.

Essa disciplina se conectou de forma direta com Testes Automatizados, que aprofundou o uso de *frameworks* de teste, e com Infraestrutura (DevOps), que integrou as práticas de programação ágil aos *pipelines* de entrega contínua. Além disso, teve relação com Modelagem Ágil de Software e Introdução à Programação, pois reforçou a importância de alinhar boas práticas de implementação com os modelos e requisitos previamente definidos.

7.1 ARTEFATOS DO PROJETO

FIGURA 18 – Código desenvolvido na disciplina.

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

class BubbleSort {

    // An optimized version of Bubble Sort
    static void bubbleSort(int arrayNumero[], int totalArray)
    {
        int i;
        for (i = 0; i < totalArray - 1; i++) {
            // 3ª Modificação: extraí a lógica de troca de elementos para um método separado "realizarTrocas".
            if (realizarTrocas(arrayNumero, totalArray, i)) break;
        }
    }

    private static boolean realizarTrocas(int[] arrayNumero, int totalArray, int i) {
        boolean trocado = false;
        int j;
        for (j = 0; j < totalArray - i - 1; j++) {
            if (arrayNumero[j] > arrayNumero[j + 1]) {
                // 1ª Modificação: criei o método "isTrocado" para realizar a troca de elementos.
                isTrocado(arrayNumero, j);
                trocado = true;
            }
        }
        // 2ª Modificação: criei o método "verificarTroca" para verificar se houve trocas e decidir se o laço externo deve ser interrompido.
        return verificarTroca(trocado);
    }

    private static boolean verificarTroca(boolean trocado) {
        // If no two elements were
        // trocado by inner loop, then break
        if (trocado == false)
            return true;
        return false;
    }

    // 1ª Modificação: método criado para encapsular a lógica de troca de elementos.
    private static void isTrocado(int[] arrayNumero, int i) {
        int temp = arrayNumero[i];
        arrayNumero[i] = arrayNumero[i + 1];
        arrayNumero[i + 1] = temp;
    }

    // Function to print an array
    static void printArray(int arrayNumero[], int size)
    {
        int i;
        for (i = 0; i < size; i++)
            System.out.print(arrayNumero[i] + " ");
        System.out.println();
    }

    // Driver program
    public static void main(String args[])
    {
        int arrayNumero[] = { 64, 34, 25, 12, 22, 11, 90 };
        int totalArray = arrayNumero.length; // 4ª Modificação: renomeei a variável "n" para "totalArray" para melhorar a legibilidade.
        bubbleSort(arrayNumero, totalArray);
        System.out.println("Array ordenado: ");
        printArray(arrayNumero, totalArray);
    }
}

```

Fonte: Autor (2025).

8 DISCIPLINA: *WEB* 1 E *WEB* 2 – DESENVOLVIMENTO *WEB* 1 E 2

A disciplina de Desenvolvimento *Web* 1 apresentou os fundamentos para construção de aplicações *Web* utilizando o *framework Angular*. Foram explorados conceitos de *TypeScript, ECMAScript, Node.js* e *HTML*, bem como a estrutura e organização de projetos em *Angular*. Além da introdução ao ambiente de desenvolvimento (*Node, NPM, Angular CLI* e *VS Code*), a disciplina abordou a modularização de aplicações, a criação de componentes e o uso de bibliotecas de estilo como Bootstrap e Material Design.

O projeto desenvolvido na disciplina consistiu na implementação de um CRUD de Pessoas, com funcionalidades de listagem, inserção, edição e remoção. Esse trabalho possibilitou aplicar os conceitos de módulos, serviços e componentes, consolidando os conhecimentos de *Angular* e reforçando o aprendizado por meio de práticas de codificação.

A importância dessa disciplina para o desenvolvimento ágil está em oferecer a base tecnológica para a criação de *Single Page Applications* (SPAs), que permitem maior dinamismo e experiência de usuário, alinhadas às demandas atuais do mercado. O domínio dessas ferramentas facilita a integração com práticas ágeis, como entregas incrementais e prototipagem rápida. Além disso, Desenvolvimento *Web* 1 se integra fortemente a disciplinas como Banco de Dados, que fornece a estrutura de persistência, e Modelagem Ágil de *Software*, que guia o design dos requisitos e artefatos. O conhecimento adquirido também prepara terreno para Desenvolvimento *Web* 2, no qual os conceitos são expandidos com maior complexidade.

A disciplina de Desenvolvimento *Web* 2 aprofundou os conhecimentos em *Web* 1, avançando para o desenvolvimento de funcionalidades mais complexas. Foram trabalhados formulários com diretivas personalizadas, validação de campos, máscaras, *pipes*, e integração com *APIs REST*. Também foi dada ênfase à programação reativa, autenticação de usuários e à implementação de *back-end* com *Spring Boot* e *Spring Data JPA*, permitindo a criação de CRUDs integrados e persistência em banco de dados.

A relevância dessa disciplina está em consolidar uma visão *full stack*, capacitando o aluno a desenvolver aplicações completas, seguras e escaláveis. Essa abordagem é essencial em projetos ágeis, pois permite ciclos de entrega mais curtos e maior autonomia da equipe de desenvolvimento na implementação de funcionalidades ponta a ponta.

Por fim, Desenvolvimento *Web* 2 integra-se diretamente a disciplinas como Aspectos Ágeis de Programação e Testes Automatizados, uma vez que reforça práticas de qualidade de código, testes e integração contínua. Também se conecta a GAP1 e GAP2, já que a gestão de projetos ágeis exige compreender a viabilidade técnica e os ciclos de desenvolvimento.

8.1 ARTEFATOS DO PROJETO

FIGURA 19 – Lista de alunos cadastrados no sistema.

Fonte: Autor (2025)

FIGURA 20 – Tela de cadastro de novos alunos.

Fonte: Autor (2025)

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE *SOFTWARE*

A disciplina de UX no Desenvolvimento Ágil de *Software* teve como objetivo introduzir conceitos fundamentais de *User Experience* (UX) e *User Interface* (UI), enfatizando sua aplicação em contextos de metodologias ágeis. Foram apresentados princípios de design centrado no usuário, técnicas de pesquisa com usuários e ferramentas de prototipagem, reforçando a importância de considerar as necessidades e expectativas do público-alvo desde as etapas iniciais do desenvolvimento de *Software* .

O projeto da disciplina consistiu na elaboração de protótipos interativos, que permitiram validar hipóteses de design e testar fluxos de navegação antes da implementação definitiva. Esse trabalho contemplou a definição de personas, jornadas do usuário e *wireframes*, favorecendo a experimentação e a melhoria contínua das soluções propostas.

A importância dessa disciplina para o desenvolvimento ágil está na sua contribuição para a entrega de valor real ao usuário, uma vez que garante que as funcionalidades priorizadas nos sprints estejam alinhadas às demandas práticas de usabilidade. O enfoque em prototipagem rápida também reforça o caráter iterativo do ágil, permitindo ajustar soluções com base em feedback constante.

Por fim, a disciplina de UX se integrou diretamente a áreas como Modelagem Ágil de *Software* , ao detalhar requisitos sob a ótica do usuário, e Desenvolvimento *Web* e *Mobile*, fornecendo subsídios para a implementação de interfaces mais intuitivas e eficazes. Além disso, sua articulação com Testes Automatizados assegurou que critérios de usabilidade fossem considerados junto à validação técnica, completando a visão de qualidade no ciclo ágil.

9.1 ARTEFATOS DO PROJETO

9.2 EXPLICAÇÃO DO PRODUTO

Nesta seção, serão discutidas as funcionalidades do aplicativo *SysCosm* para compreensão e clareza.

9.2.1 Função do *SysCosm*

O *SysCosm* é um aplicativo para gerenciamento de revendedores e consultores de produtos cosméticos, a qual um executivo de vendas é responsável por gerenciar o seu pessoal. O aplicativo promove uma produtividade e o gerenciamento eficaz da equipe. O *SysCosm* ajuda o executivo de vendas a gerenciar os revendedores e consultores, porque muitos deles não são acostumados a realizar os pedidos no site. Logo, a revendedora precisa do acesso dos revendedores e consultores para realizar os pedidos de toda a rede.

9.2.2 Justificativa

Executivos de vendas frequentemente enfrentam desafios na organização de grandes listas de revendedores e consultores, especialmente quando essa gestão é realizada manualmente em cadernos ou planilhas. Essa metodologia atrasa a consulta e impede uma rápida visualização das informações de cada membro. O *SysCosm* foi desenvolvido para otimizar esse processo, proporcionando um ambiente digital centralizado para a consulta e edição de informações essenciais, trazendo agilidade ao dia a dia dos executivos de vendas.

9.2.3 Objetivo

O objetivo do *SysCosm* é proporcionar uma ferramenta intuitiva para o gerenciamento, garantindo que os executivos de vendas consigam gerenciar suas equipes de revendedores e consultores de atividades de forma prática e acessível.

9.3 DESENVOLVIMENTO DAS TELAS

9.3.1 Tela 1: Tela de *Login*

Descrição: nesta tela são solicitadas as credenciais do usuário (executiva de vendas, as quais são: o nome de usuário e a senha. Para entrar é clicado no botão “Entrar”. Quando o usuário não tiver cadastro, será automaticamente direcionado para tela de cadastro para realizar o registro no aplicativo. Para o cadastro é solicitado o nome de usuário, a senha e a confirmação de senha, após o preenchimento, deverá clicar no botão “Cadastrar”.

9.3.2 Tela 2: Lista de revendedores e consultores

Descrição: Na FIGURA 22a Permite ao executivo de vendas listar revendedores e consultores com detalhes como nome, registro e senhas que foram adicionados. Os campos são intuitivos, e o design minimalista facilita o preenchimento rápido. Também estão inclusos os botões de “editar”: a qual pode ser utilizados para realizar alterações no perfil, e de “excluir”: caso não existe cadastro, ou seja, foram encerradas atividades como revendedor ou consultor, e logo ele será direcionado para a tela da FIGURA 22b. Para adicionar novo revendedor ou consultor, tem o botão com sinal “+” para incluir na lista.

9.3.3 Tela 3: Tela de cadastro ou edição de revendedores, ou consultores

Descrição: Tela onde o executivo de vendas pode modificar ou alterar um revendedor existente. O perfil possui as seguintes características: nome, registro, senha da usuária, CPF, RG e data de nascimento.

FIGURA 21 – Telas de *Login* e Cadastro

(a) Login

(b) Cadastro

Fonte: Autor (2025).

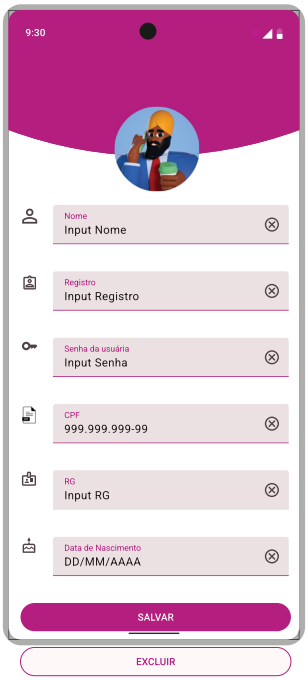
FIGURA 22 – Telas de listagem e confirmação de exclusão.

(a) Tela de listagem.

(b) Tela de confirmação para exclusão.

Fonte: Autor (2025).

FIGURA 23 – Formulário do revendedor ou consultor



Fonte: Autor (2025).

9.4 EXPLICAÇÃO DO PRODUTO

Nesta seção, será discutido as funcionalidades do aplicativo *SysCosm* para compreensão e clareza.

9.4.1 Função do *SysCosm*

O *SysCosm* é um aplicativo para gerenciamento de revendedores e consultares de produtos cosméticos, a qual um executivo de vendas é responsável de gerenciar o seu pessoal. O aplicativo promove uma produtividade e o gerenciamento eficaz da equipe. O *SysCosm* ajuda o executivo de vendas a gerenciar os revendedores e consultores porque muitos deles não são acostumados a realizar os pedidos no site. Logo, a revendedora precisa do acesso dos revendedores e consultores para realizar os pedidos de toda a rede.

9.4.2 Justificativa

Executivos de vendas frequentemente enfrentam desafios na organização de grandes listas de revendedores e consultores, especialmente quando essa gestão é realizada manualmente em cadernos ou planilhas. Essa metodologia atrasa a consulta e impede uma rápida visualização das informações de cada membro. O *SysCosm* foi desenvolvido para otimizar esse processo, proporcionando um ambiente digital centralizado para a consulta e edição de informações essenciais, trazendo agilidade ao dia a dia dos executivos de vendas.

9.4.3 Objetivo

O objetivo do *SysCosm* é proporcionar uma ferramenta intuitiva para o gerenciamento, garantindo que os executivos de vendas consigam gerenciar suas equipes de revendedores e consultores atividades de forma prática e acessível.

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

A disciplina de Infraestrutura para Desenvolvimento e Implantação de *software* (*DevOps*) teve como objetivo apresentar técnicas e práticas voltadas para a automação e a integração contínua no ciclo de vida do *software*. Foram explorados conceitos essenciais como sistemas de controle de versão, pipelines de integração e entrega contínua (CI/CD), automação de builds, implantação de aplicações, conteinerização com *Docker* e orquestração com *Kubernetes*. Além disso, a disciplina abordou práticas modernas de *Infrastructure as Code*, microserviços e observabilidade, destacando como esses elementos contribuem para maior agilidade, confiabilidade e escalabilidade no desenvolvimento de *software*. A proposta central foi evidenciar a importância da cultura *DevOps*, que promove colaboração entre equipes de desenvolvimento e operações, redução de desperdícios e entregas mais rápidas e seguras de valor ao cliente.

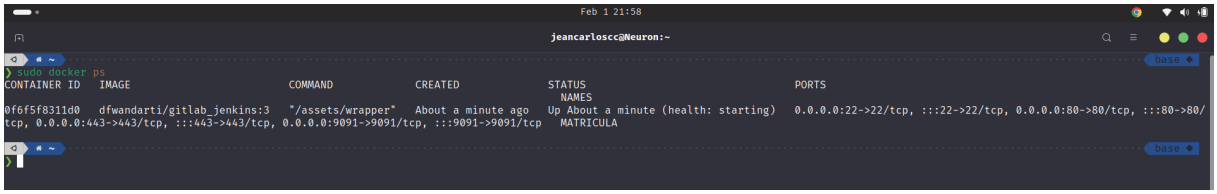
Durante a disciplina, foi desenvolvido um projeto prático que envolveu a criação de um pipeline automatizado de integração e entrega contínua, utilizando Git como sistema de versionamento, além da configuração de ambientes com Docker para garantir a portabilidade das aplicações. O trabalho incluiu a definição de scripts de build e deploy, bem como a execução de testes automatizados integrados ao pipeline. Também foram utilizadas métricas de desempenho e qualidade para acompanhar a eficiência das entregas, reforçando a importância de monitoramento e *feedback* rápido. Esse projeto permitiu aplicar na prática os conceitos de CI/CD, automação e infraestrutura como código, simulando um fluxo real de entrega contínua em ambientes ágeis.

A disciplina teve papel essencial na consolidação dos princípios do desenvolvimento ágil, uma vez que possibilitou compreender como práticas de *DevOps* reduzem o tempo de entrega de valor, aumentam a confiabilidade do *software* e facilitam a adaptação a mudanças. A automação de tarefas repetitivas e a integração contínua garantiram ciclos curtos de *feedback*, alinhando-se diretamente à filosofia ágil de entregas incrementais. Além disso, a abordagem de observabilidade e monitoramento contínuo reforçou a ideia de melhoria constante e de resposta rápida a incidentes, aspectos fundamentais para manter a qualidade em ambientes de alta complexidade.

Os conteúdos estudados tiveram forte integração com diversas disciplinas do curso. Com testes automatizados, o *DevOps* se mostrou essencial para garantir que os testes fossem executados de forma contínua dentro do *pipeline*. Com desenvolvimento web e mobile, possibilitou a automação de *deploys* e a disponibilização de novas versões em ambientes controlados. A disciplina também se relacionou com Gerenciamento Ágil de Projetos, ao apoiar a organização de entregas frequentes, e com Banco de Dados, no uso de *scripts* de migração automatizados integrados ao processo de implantação. Dessa forma, a disciplina consolidou-se como um elo central para viabilizar a entrega de valor no ecossistema.

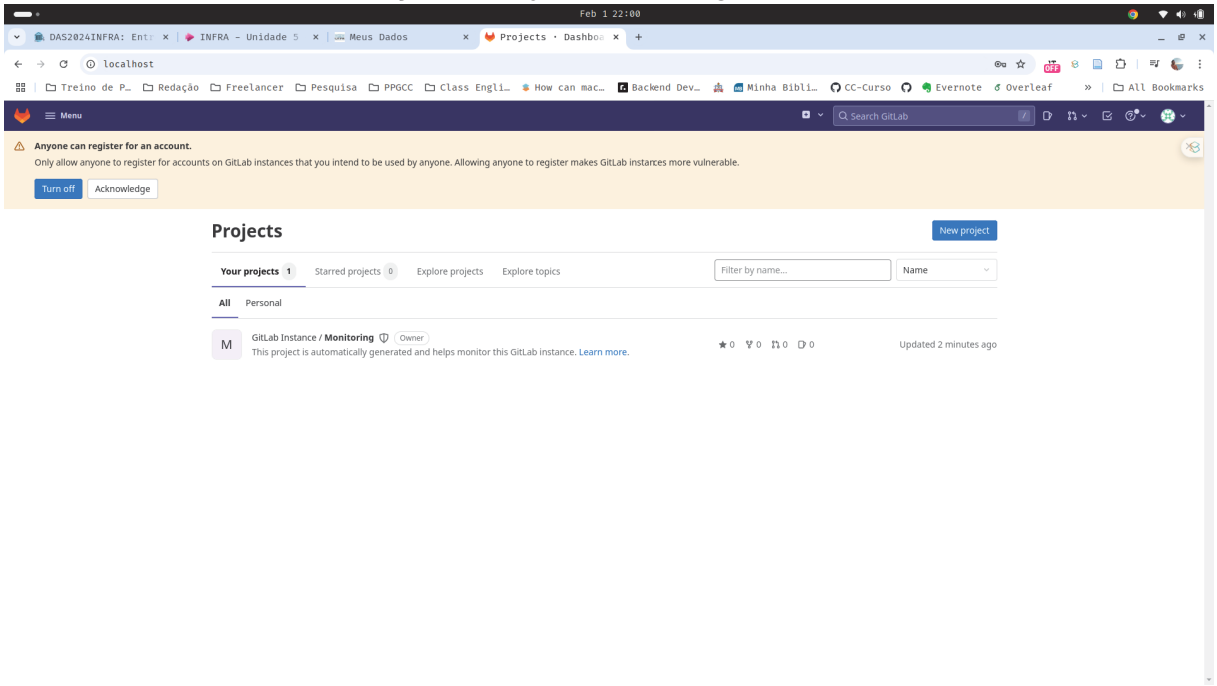
10.1 ARTEFATOS DO PROJETO

FIGURA 24 – Container *Docker* em execução hospedando a instância GitLab para o projeto *Monitoring*.



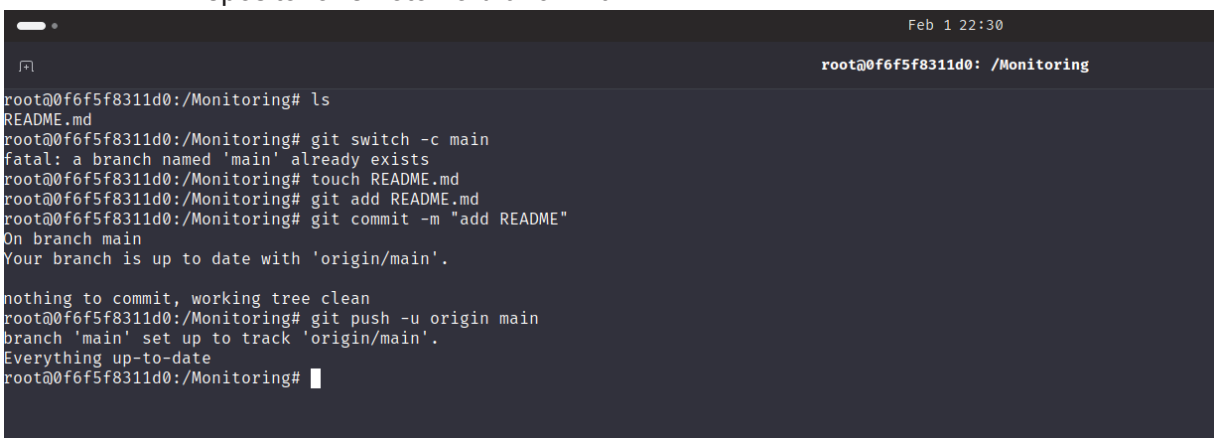
Fonte: Autor (2025)

FIGURA 25 – Visualização do projeto ‘Monitoring’ criado e disponível no GitLab.



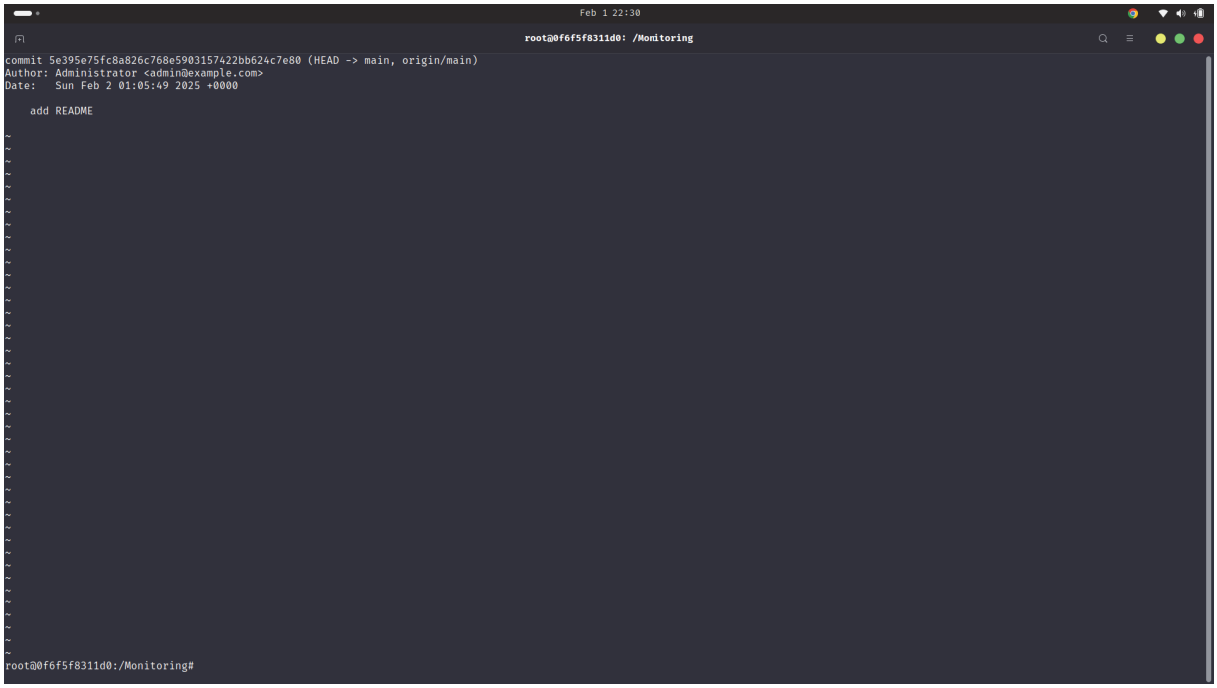
Fonte: Autor (2025)

FIGURA 26 – Execução de comandos *Git* para criar, adicionar e enviar o arquivo *README* ao repositório remoto no *branch main*.



Fonte: Autor (2025)

FIGURA 27 – Histórico de *commit* mostrando a adição do arquivo README no *branch* principal (*main*).



Fonte: Autor (2025)

11 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados teve como foco apresentar conceitos, técnicas e ferramentas voltadas para a automação de testes de software. Foram abordados fundamentos sobre a importância da qualidade no ciclo de desenvolvimento, diferentes níveis de testes (unidade, integração, sistema e aceitação), além de práticas modernas como *Test-Driven Development* (TDD), *Behavior-Driven Development* (BDD) e *Continuous Testing*. Também foram exploradas ferramentas amplamente utilizadas no mercado, como *JUnit*, *Selenium*, *Cypress* e *Postman*, permitindo compreender como os testes automatizados se inserem no fluxo de integração e entrega contínua. A disciplina reforçou que a automação de testes não é apenas um suporte técnico, mas uma prática essencial para assegurar qualidade, reduzir custos de manutenção e aumentar a confiabilidade das entregas ágeis.

Como parte prática da disciplina, foi desenvolvido um projeto que consistiu na implementação de uma suíte de testes automatizados para validar funcionalidades de uma aplicação *web*. O trabalho contemplou desde a criação de testes unitários com *JUnit* até a automação de cenários de interface com *Selenium*, incluindo a execução de testes de integração para verificar a comunicação entre diferentes módulos do sistema. A suíte de testes foi integrada a um *pipeline* de integração contínua, garantindo que cada nova versão da aplicação fosse validada automaticamente antes de ser implantada. Esse projeto proporcionou a aplicação direta dos conceitos de TDD e BDD, bem como a experiência com *frameworks* de automação, consolidando a prática de qualidade contínua no desenvolvimento ágil.

A automação de testes é um pilar fundamental para o desenvolvimento ágil, pois garante ciclos curtos de *feedback* e possibilita entregas frequentes com maior segurança. Ao reduzir a dependência de testes manuais e repetir execuções de forma automática, a disciplina mostrou como é possível manter a qualidade mesmo em cenários de rápidas mudanças de requisitos. Além disso, o uso de TDD e BDD contribuiu para alinhar o código às especificações de negócio, aproximando desenvolvedores e *stakeholders* no processo de validação de funcionalidades. Dessa forma, a disciplina evidenciou que a automação de testes não apenas acelera o desenvolvimento, mas também promove maior confiabilidade nas entregas.

O conteúdo da disciplina se conectou diretamente com Aspectos Ágeis de Programação, ao aplicar práticas de TDD e refatoração em conjunto com testes automatizados. Também se integrou à disciplina de Infraestrutura (*DevOps*), já que a execução de testes contínuos foi incorporada aos pipelines de CI/CD. Nas disciplinas de Desenvolvimento *Web* e *Mobile*, os testes garantiram a qualidade das interfaces implementadas, enquanto em Banco de Dados auxiliaram na validação da integridade e consistência das consultas. Com isso, a disciplina de Testes Automatizados se des-

tacou como um elo de sustentação para todo o processo de desenvolvimento ágil, fortalecendo a integração entre codificação, implantação e validação.

11.1 ARTEFATOS DO PROJETO

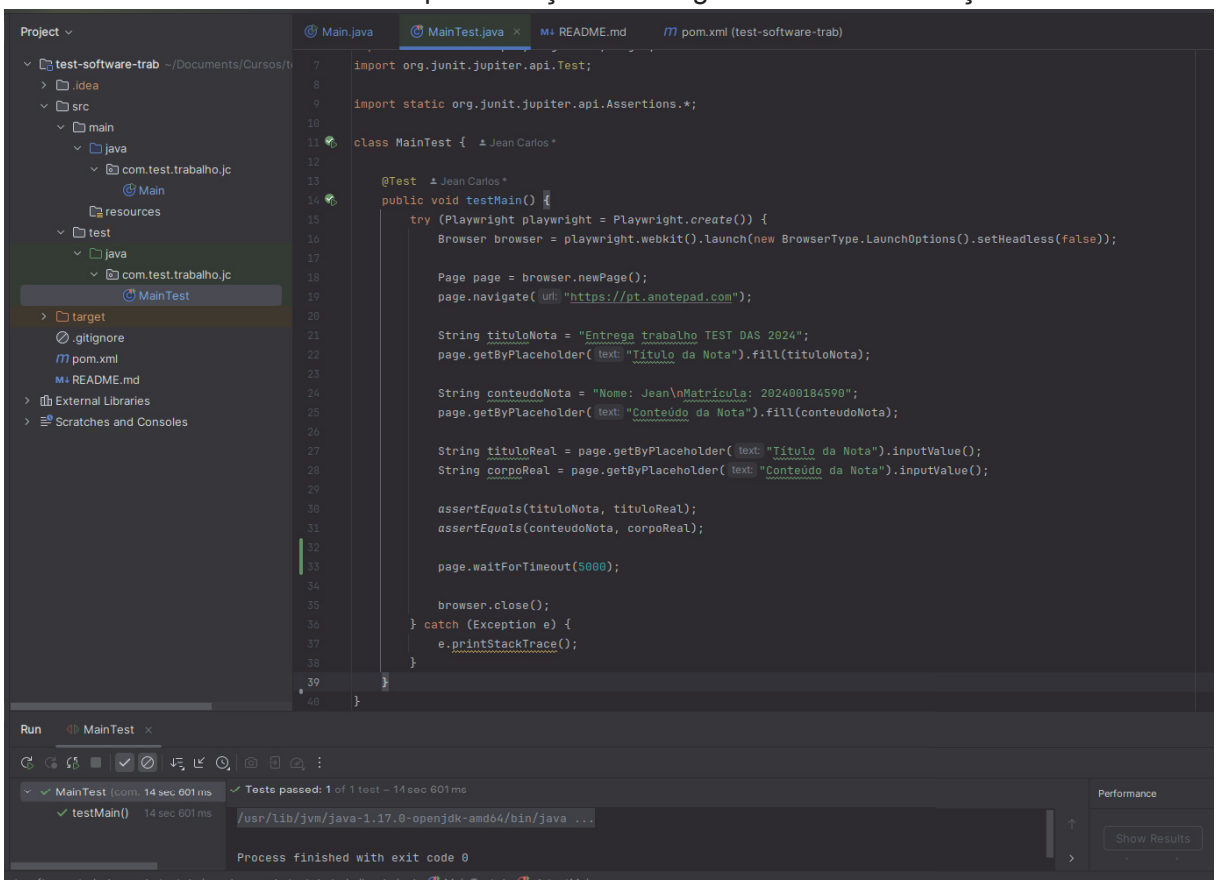
11.1.1 Introdução

Este trabalho tem como objetivo apresentar os resultados do código implementado utilizando testes em um ambiente web, como o *Playwright*. Como principal atividade do trabalho, é utilizar a ferramenta *Playwright* para acessar o site Anotepad, criar uma nota no título e adicionar no corpo contendo o nome e matrícula.

11.1.2 Implementação

O código[1] abaixo utiliza *Playwright* para abrir o navegador, acessar o site e preencher os campos solicitados na descrição do trabalho.

FIGURA 28 – Implementação do código de testes e execução.



Fonte: Autor (2025).

[1] https://github.com/jeancarloscc/test-software-trab

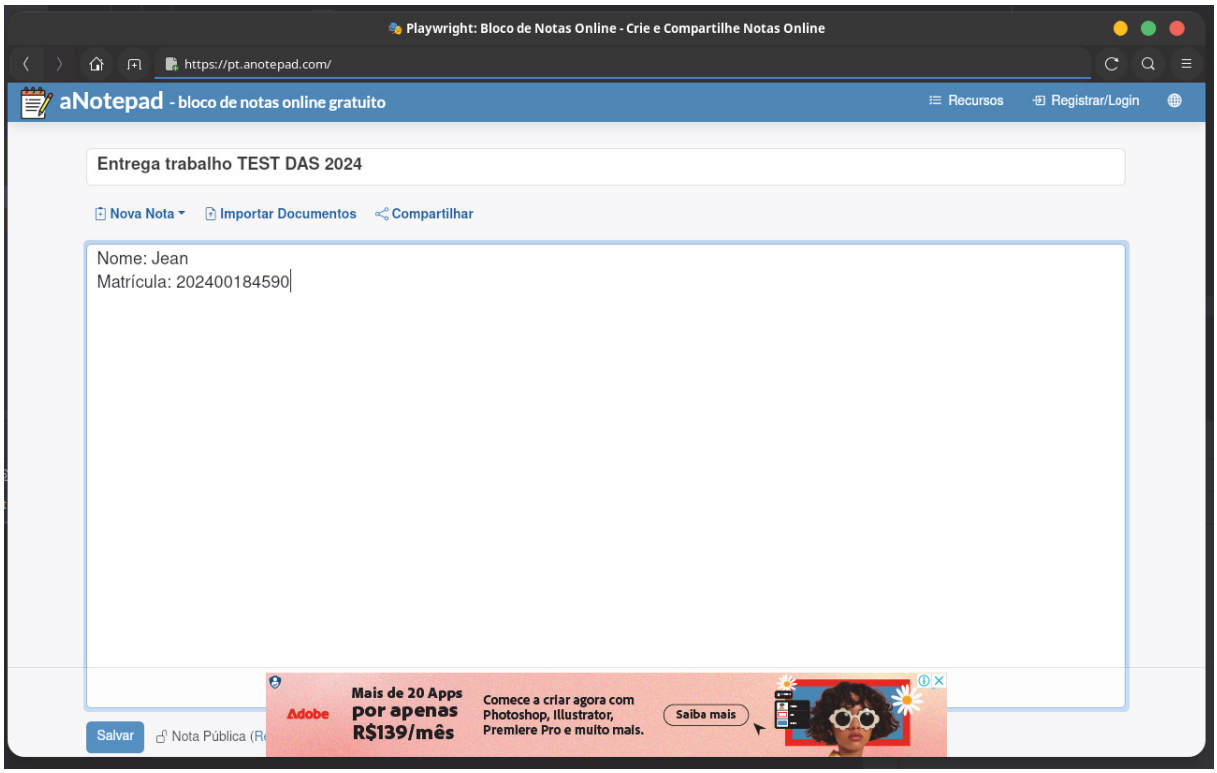
11.1.3 Explicação do Código

O código inicia o *Playwright*, abre o navegador *webkit* e acessa o site https://pt.anotepad.com e aguarda o carregamento da página. Assim, são criadas as notas e preenchidos os campos com as informações:

- Título da Nota: “Entrega trabalho TEST DAS 2024”
- Conteúdo da Nota: “Nome: Jean e Matricula: 202400184590.

Por fim, aguarda um pequeno tempo para garantir que os dados sejam inseridos corretamente e fecha o navegador.

FIGURA 29 – Registro de entrega do trabalho TEST DAS 2024 no aNotepad, contendo nome e matrícula do aluno.



Fonte: Autor (2025).

11.1.4 Conclusão

O *Playwright* proporcionou um código eficiente para a automação do site Anotepad. Com a utilização da ferramenta, foi possível automatizar a criação de notas rápida e eficaz, garantindo a correta execução do trabalho proposto.

12 CONCLUSÃO

O objetivo central deste memorial foi analisar de forma integrada os projetos desenvolvidos ao longo da especialização em Desenvolvimento Ágil de *software*, buscando compreender como as diferentes disciplinas se articularam na prática do desenvolvimento ágil. Esse objetivo foi plenamente alcançado, uma vez que os trabalhos apresentados permitiram vivenciar de maneira aplicada os princípios de colaboração, iteração, automação e entrega contínua de valor.

Os principais resultados evidenciaram que a integração entre disciplinas como modelagem, programação, banco de dados, testes, UX e *DevOps* proporcionou uma visão sistêmica do processo de desenvolvimento. A articulação entre planejamento ágil, implementação, automação de testes e entrega contínua mostrou-se fundamental para aproximar os projetos acadêmicos das demandas reais enfrentadas em ambientes profissionais.

As contribuições deste trabalho se manifestam em três aspectos. No campo acadêmico, o memorial reforça a importância de currículos que integrem teoria e prática, oferecendo experiências que simulam o contexto profissional. Na prática profissional, a experiência adquirida amplia competências em metodologias ágeis, automação e integração de equipes multidisciplinares. Do ponto de vista social, o domínio dessas práticas fortalece a formação de profissionais mais preparados para atuar em equipes colaborativas, entregando soluções de *software* com maior qualidade e foco no usuário.

Em reflexão final, este memorial demonstrou que a integração multidisciplinar no desenvolvimento ágil de *software* é não apenas viável, mas necessária para formar profissionais capazes de enfrentar os desafios da engenharia de *software* contemporânea. Ao unir teoria e prática, os projetos desenvolvidos reforçam o valor da educação aplicada como instrumento de transformação acadêmica e profissional, contribuindo para a consolidação de uma abordagem ágil, colaborativa e orientada a resultados no campo do desenvolvimento de *software*.

REFERÊNCIAS

BRÓLIO, D. R. **Agilidade na gestão de projetos: Estratégias e práticas**. 1 ed. São Paulo: Editora Senac, nov. 2024.

CANJUNGO, A. R. *et al.* **Análise da aderência de práticas ágeis e o Framework Lean IT sua complementariedade no gerenciamento de projectos: caso de estudo no setor bancário**. 2021. Mestrado em Engenharia Informática e Sistemas de Informação – Departamento de Engenharia Informática e Sistema de Informação, Universidade Lusófona de Humanidades e Tecnologias, Lisboa.

COSTA, J. L. d. *et al.* **Metodologia de gerenciamento de projetos aplicada ao desenvolvimento de produtos em uma empresa do ramo metalmecânico**. Jan. 2025. Monografia (Bacharelato em Engenhaia Mecânica) – Instituto Federal de Educação, Ciência e Tecnologia, Ibirubá. 74 f.

MORAES, A. R. O. **Metodologias Ágeis e Indústrias Criativas Um Caso de Aplicação do Lean Inception em Projetos de Investigação**. 2021. Diss. (Mestrado) – Universidade do Porto (Portugal).

NASCIMENTO, I. J. M. d. **Desenvolvimento de uma aplicação móvel para gerenciamento de atividades complementares em instituições de ensino superior**. 2023. Monografia (Engenharia de Software) – Universidade Federal do Ceará, Quixadá.

NOVAES, G. F. *et al.* **Sucesso de projetos de transformação digital**: uma análise comparativa das metodologias de gerenciamento de projetos e proposição de modelo. 2025. f. 138. Doutorado em Administração – Programa de Pós-Graduação em Gestão de Projetos, Universidade Nove de Julho, São Paulo.

OLIVEIRA, B. H. **Qualidade de software no desenvolvimento com métodos ágeis**. 2014. f. 85. Mestrado em Ciências de Computação e Matemática Computacional – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Disponível em: http://dx.doi.org/10.11606/D.55.2014.tde-12082014-165244.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos Ágeis para Desenvolvimento de Software**. [*S. l.*]: Bookman Editora, 2014. ISBN 9788582602089. Disponível em: https://books.google.com.br/books?id=8rQABAAAQBAJ.

RAPÔSO, C. F. L. *et al.* Domain-Driven Design: revisão sistemática da literatura, lacunas e direções futuras. **Revista Tópicos**, v. 3, n. 23, p. 1–16, jul. 2025.

SILVA, L. R. M. **Desenvolvimento de Software e Metodologias Ágeis**. 1. ed. Rio de Janeiro: Freitas Bastos, 2024. p. 230. ISBN 9786556754390.