UNIVERSIDADE FEDERAL DO PARANÁ

WAGNER AUGUSTO LEODORO

MEMORIAL DE PROJETOS: INTEGRAÇÃO MULTIDISCIPLINAR E PRÁTICAS ITERATIVAS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

CURITIBA

2025

WAGNER AUGUSTO LEODORO

MEMORIAL DE PROJETOS: INTEGRAÇÃO MULTIDISCIPLINAR E PRÁTICAS ITERATIVAS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

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 **WAGNER AUGUSTO LEODORO**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO MULTIDISCIPLINAR E PRÁTICAS ITERATIVAS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE**, 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, 17 de Outubro 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

O memorial apresenta a trajetória de aprendizagem e aplicação dos métodos ágeis de desenvolvimento de software, articulando teoria e prática por meio dos projetos realizados ao longo do curso. O objetivo é demonstrar como as disciplinas se integraram para consolidar uma visão sistêmica do desenvolvimento ágil, baseada em colaboração, entrega incremental e adaptação contínua. Foram elaborados diversos artefatos que ilustram essa evolução, como mapas mentais, diagramas UML, histórias de usuário, protótipos de interface, planos de release, códigos refatorados, testes automatizados, scripts SQL, aplicativos mobile e containers Docker. O memorial destaca que o desenvolvimento ágil ultrapassa o uso de ferramentas, representando uma mudança cultural que valoriza comunicação clara, integração entre equipes e foco na entrega contínua de valor. A experiência adquirida ao longo do curso permitiu não apenas o domínio de técnicas modernas, mas também o desenvolvimento de competências essenciais para atuar em equipes multidisciplinares, adaptando-se rapidamente às mudanças e entregando soluções de alto valor agregado.

Palavras-chave: Desenvolvimento de Software; Métodos Ágeis; Engenharia de Software; Modelagem de Sistemas.

ABSTRACT

The memorial presents the learning journey and practical application of agile software development methods, integrating theory and practice through projects completed throughout the course. Its objective is to demonstrate how the disciplines were interconnected to consolidate a systemic view of agile development, grounded in collaboration, incremental delivery, and continuous adaptation. Several artifacts were created to illustrate this evolution, including mind maps, UML diagrams, user stories, interface prototypes, release plans, refactored code, automated tests, SQL scripts, mobile applications, and Docker containers. The memorial emphasizes that agile development goes beyond the use of tools, representing a cultural transformation that values clear communication, team integration, and a constant focus on value delivery. The experience gained during the course enabled not only the mastery of modern techniques but also the development of essential skills to work effectively in multidisciplinary teams, adapt quickly to change, and deliver high-value software solutions.

Keywords: Software Development; Agile Methods; Software Engineering; System Modeling

LISTA DE FIGURAS

SUMÁRIO

1 PARECER TÉCNICO

O presente parecer técnico tem como finalidade expor, de maneira lógica e estruturada, os projetos realizados no decorrer das disciplinas do curso de Pós-Graduação em Desenvolvimento Ágil de Software, destacando a contribuição de cada etapa para o fortalecimento de uma compreensão integrada e aplicada dos princípios do desenvolvimento ágil.

No âmbito da disciplina de Métodos Ágeis para Desenvolvimento de Software (MADS), foi elaborado um mapa mental que sintetiza conceitos fundamentais como modelos tradicionais, Manifesto Ágil, *Scrum*, *XP*, *Kanban* e entrega contínua, servindo de base conceitual para as demais disciplinas. Em Modelagem Ágil de Software (MAG1 e MAG2), destaca-se o desenvolvimento do Sistema de Gestão de Condomínios, com artefatos como diagramas de caso de uso, histórias de usuário, protótipos de tela, diagramas de classes e sequência, evidenciando a aplicação prática do *Domain Driven Design* (Evans, 2003) e da modelagem incremental. Segundo Evans (2003), o *Domain Driven Design* propõe que a modelagem de software deve ser orientada pelo entendimento profundo do domínio de negócio, permitindo que as soluções desenvolvidas estejam alinhadas às necessidades reais dos usuários e favorecendo a comunicação entre equipe técnica e *stakeholders*. Essa abordagem é especialmente relevante em ambientes ágeis, pois incentiva a evolução incremental do modelo conforme o conhecimento sobre o domínio se aprofunda.

Nas disciplinas de Gerenciamento Ágil de Projetos (GAP1 e GAP2), foram produzidos planos de release, simulações de Kanban Board Game e gráficos de fluxo cumulativo, integrando práticas de planejamento, acompanhamento visual e métricas ágeis. Segundo Prikladnicki *et al*. (2014), o processo de planejamento em métodos ágeis deve ser iterativo e colaborativo, incorporando estimativas realizadas pela equipe para promover maior realismo e comprometimento com os prazos. As estimativas, quando feitas de forma participativa, permitem ajustar o escopo e a capacidade de entrega, tornando o planejamento mais flexível e adaptado à realidade do projeto. Dessa forma, o uso de técnicas como *planning poker* e análise por pontos de função contribui para a transparência e para a tomada de decisão baseada em dados. Em Introdução à Programação, o *backend* de um sistema bancário foi desenvolvido com validação por testes automatizados, consolidando o uso do desenvolvimento guiado por testes e integração com banco de dados relacional.

Segundo Beck (2010), a prática do desenvolvimento guiado por testes (TDD), contribui significativamente para a qualidade do software, pois permite detectar defeitos precocemente e garante que o código atenda aos requisitos esperados. Os testes automatizados promovem confiança nas alterações realizadas e facilitam a manutenção evolutiva do sistema, tornando o processo de desenvolvimento mais seguro e eficiente.

A disciplina de Banco de Dados (BD) trouxe modelos conceituais e lógicos, *scripts* SQL e consultas avançadas, reforçando a importância da persistência e integridade dos dados. Aspectos Ágeis de Programação (AAP) enfatizou práticas de código limpo (Martin, 2009), refatoração incremental e especificação de cenários em *Gherkin* (DADO QUE – QUANDO – ENTÃO), promovendo integração entre requisitos e testes automatizados (Smart, 2015). Segundo Martin (2009), a clareza, simplicidade e legibilidade do código são essenciais para garantir sua manutenibilidade e qualidade, sendo que pequenas melhorias contínuas (refatoração) contribuem para evitar a degradação do software e facilitam a identificação de defeitos. Já Smart (2015) destaca que a utilização de linguagens de especificação como *Gherkin* permite alinhar requisitos de negócio e testes automatizados, tornando o processo de validação mais transparente e colaborativo entre desenvolvedores, testadores e *stakeholders*.

Em Desenvolvimento Web (WEB1 e WEB2), foram implementados CRUDs completos com Angular, integração com *backend* em Spring Boot e banco de dados PostgreSQL, além de protótipos de interface e validações. A disciplina de UX destacou o desenvolvimento do site Guia Náutico Litoral do Paraná, com *wireframes*, paleta de cores e testes de usabilidade, evidenciando a centralidade da experiência do usuário no ciclo ágil. Segundo a ISO 9241-210:2010 (International, 2010), a experiência do usuário é resultado das percepções e respostas de uma pessoa decorrentes do uso de um produto, sistema ou serviço. A norma estabelece requisitos e recomendações para o projeto centrado no ser humano em sistemas interativos, visando melhorar a interação entre usuários e sistemas. Ela é aplicável a todo o ciclo de vida de sistemas interativos computacionais, abrangendo tanto hardware quanto software. Enfatiza a importância de compreender os usuários, suas tarefas e o contexto de uso, além de envolvê-los ativamente no processo de design.

No contexto mobile (MOB1 e MOB2), foram desenvolvidos aplicativos Android nativos, como o FinApp e o app integrado à Harry Potter API, consolidando práticas de persistência local, integração com *web services* e arquitetura. Infraestrutura para

Desenvolvimento e Implantação de Software (INFRA) abordou DevOps, versionamento com Git, automação de pipelines CI/CD e conteinerização com Docker e Kubernetes, integrando práticas de entrega contínua e observabilidade. Segundo Humble e Farley (2014), a entrega contínua preconiza a automação completa dos processos de implantação de software, sustentada por ciclos de feedbacks rápidos e constantes que garantem qualidade, previsibilidade e confiança a cada entrega. Os autores destacam que a automação não só reduz erros humanos, mas também acelera o ciclo de desenvolvimento, permitindo que novas funcionalidades e correções sejam disponibilizadas de forma segura e frequente aos usuários.

Por fim, Testes Automatizados (TEST) consolidou o papel dos testes na garantia de qualidade, com aplicação da Pirâmide de Testes (Fowler, 2012), estratégia *Shift Left* e automação em *Playwright* e *JUnit*. Segundo Fowler (2012), uma estratégia eficaz de testes deve priorizar uma base sólida de testes unitários, pois eles são rápidos, confiáveis e fornecem *feedback* imediato sobre falhas no código. O autor destaca que a automação dos testes, especialmente dos testes unitários, permite detectar problemas precocemente e alcançar uma cobertura de testes adequada, promovendo maior confiança na qualidade do software e facilitando a manutenção evolutiva do sistema

A integração conceitual e prática dos projetos evidencia que o desenvolvimento ágil de software, mais do que um conjunto de ferramentas, representa uma mudança cultural pautada na colaboração, comunicação transparente, entrega incremental e adaptação contínua. Os artefatos apresentados demonstram a interdisciplinaridade e a evolução do conhecimento, consolidando uma base técnica e conceitual para um ciclo ágil sustentável e de alto padrão.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

Com o propósito de oferecer uma visão ampla sobre as atividades técnicas, métodos e ferramentas que sustentam o processo de desenvolvimento de software orientado por práticas ágeis, a disciplina foi organizada em duas unidades complementares. A primeira tratou dos fundamentos da engenharia de software, dos processos de desenvolvimento e dos modelos tradicionais, além de apresentar modelos de maturidade que buscam elevar a qualidade do produto e aprimorar o processo de construção de software. Os modelos tradicionais, apesar de contribuírem para o amadurecimento das práticas de engenharia de software, mostraram-se limitados frente à demanda por entregas mais frequentes, foco no valor ao cliente e na redução de riscos por meio de ciclos curtos (Prikaladnick, 2014). Dessa forma, a migração dos modelos prescritivos para abordagens adaptativas representa a resposta do setor à necessidade de evolução, integrando princípios como colaboração intensa, comunicação transparente e entrega incremental, pilares do desenvolvimento ágil (Prikaladnick, 2014). A segunda unidade concentrou-se nos métodos ágeis, destacando seus valores, princípios, frameworks e práticas, com ênfase nas cerimônias do *Scrum*, nos princípios e práticas do *Extreme Programming* (XP), no uso do *Kanban* e nos fundamentos da entrega contínua de software, que, segundo Hamble e Farley (2014), preconizam a automação completa dos processos de implantação de software, sustentada por ciclos de feedbacks rápidos e constantes que garantem qualidade, previsibilidade e confiança a cada entrega.

Esse embasamento teórico tornou-se indispensável para as demais disciplinas do curso. O domínio dos *frameworks* ágeis serviu como ponto de partida para atividades práticas em Gerenciamento Ágil de Projetos I (GAP1) e Gerenciamento Ágil de Projetos II (GAP2), em Metodologias Ágeis I (MAG1) e Metodologias Ágeis II (MAG2) e em Testes Automatizados (TEST). Além disso, a compreensão dos princípios de entrega contínua e da mentalidade *Lean* se conectou diretamente às práticas de *DevOps* aplicadas em Infraestrutura para Desenvolvimento e Implantação de Software (INFRA), fortalecendo a integração entre desenvolvimento e implantação de software.

Assim, a disciplina de Metodologias Ágeis para Desenvolvimento de Software (MADS) funcionou como pilar conceitual do curso, promovendo uma base sólida para

a aplicação prática dos métodos ágeis, destacando sempre o foco na entrega contínua de valor ao cliente, a colaboração entre equipes e a adaptação a cenários em constante mudança.

O trabalho principal da disciplina consistiu na elaboração de um mapa mental com nove tópicos centrais: Processo de Software, Modelos Tradicionais, Manifesto Ágil, Princípios Ágeis, *Lean Software Development*, *Scrum*, *Extreme Programming*, *Kanban* e Entrega Contínua de Software.

Esse mapa mental (Figura 1), teve papel essencial para a consolidação dos principais conceitos da disciplina, promovendo uma visão comparativa entre os modelos tradicionais e os métodos ágeis. A atividade possibilitou organizar de forma visual e interligada os fundamentos do desenvolvimento ágil, destacando a transição de abordagens prescritivas para métodos mais adaptativos, colaborativos e iterativos.

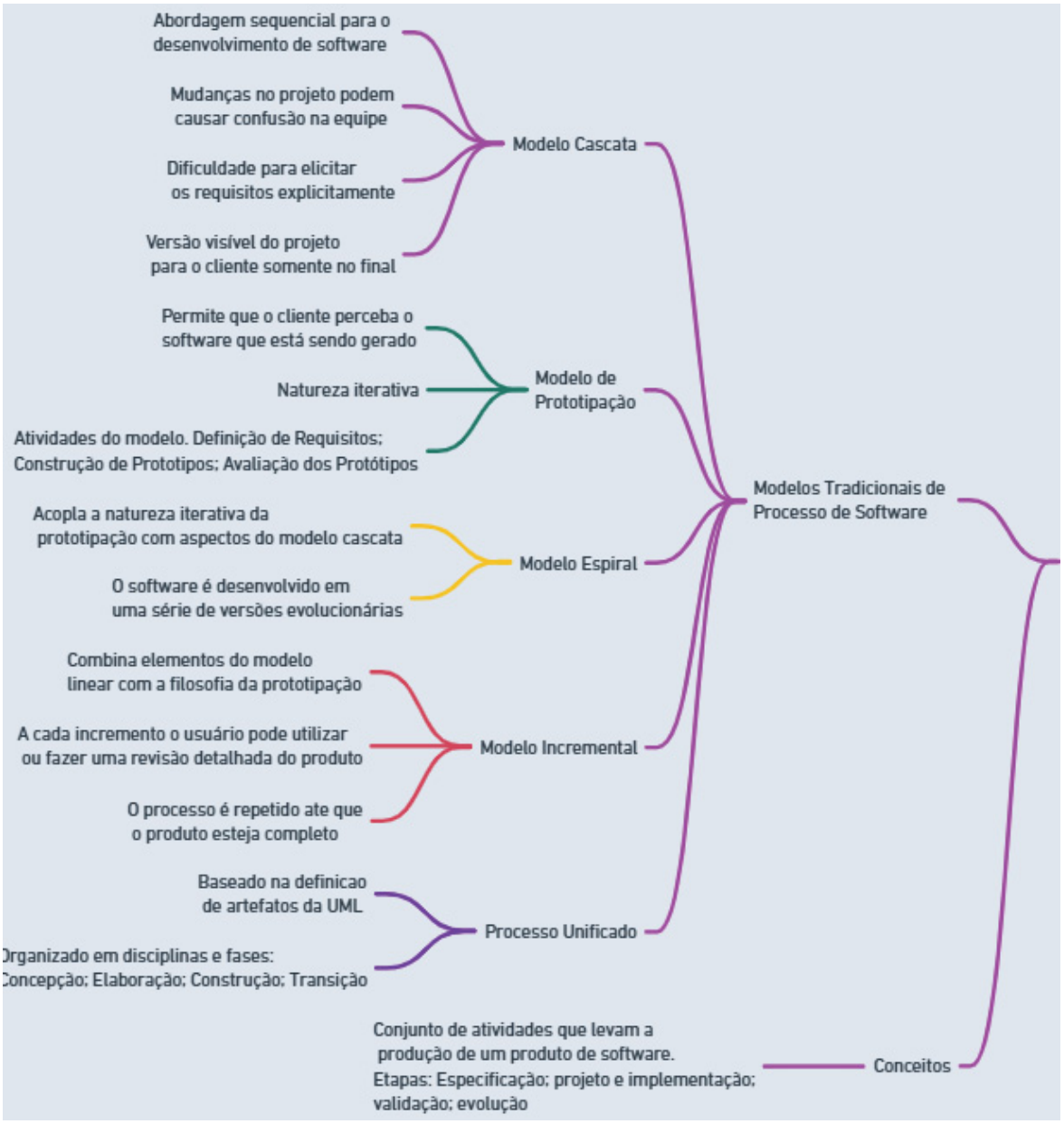
A elaboração do mapa mental permitiu uma visão sistêmica da cultura ágil, reforçando a importância da comunicação clara, da colaboração multidisciplinar e da entrega contínua de valor — aspectos fundamentais que permeiam todo o curso. Assim, MADS não apenas introduziu os conceitos centrais dos métodos ágeis, mas serviu como alicerce conceitual e integrador de todo o percurso formativo.

2.1 ARTEFATOS DO PROJETO

FIGURA 1 - MAPA CONCEITUAL MADS

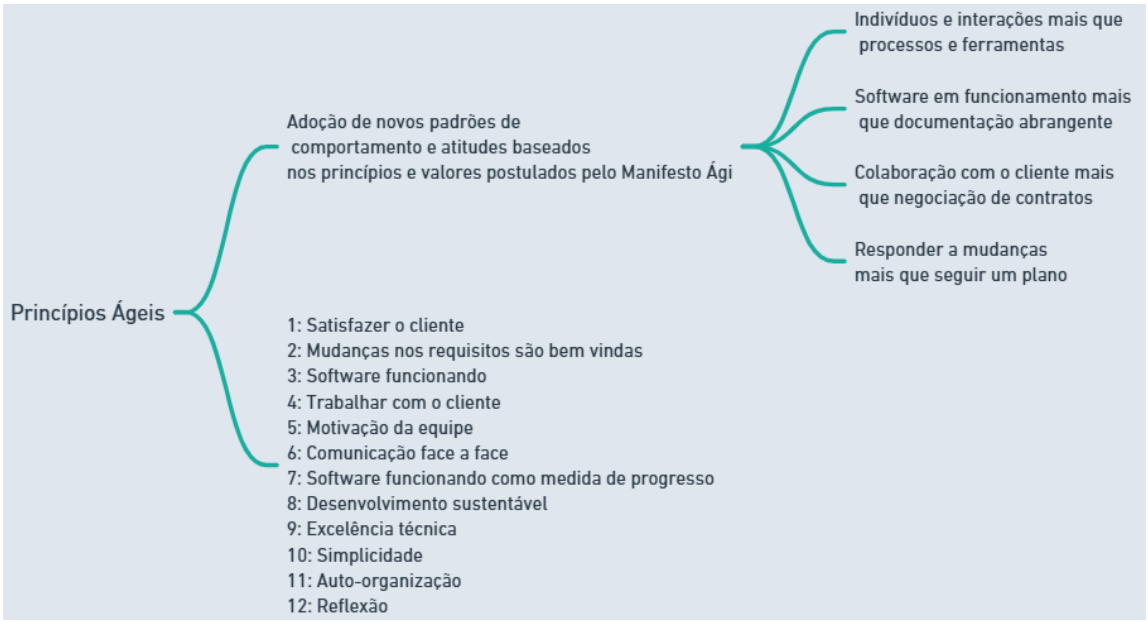
FONTE: O Autor (2025)

FIGURA 2 – MAPA CONCEITUAL: PROCESSOS DE SOFTWARE



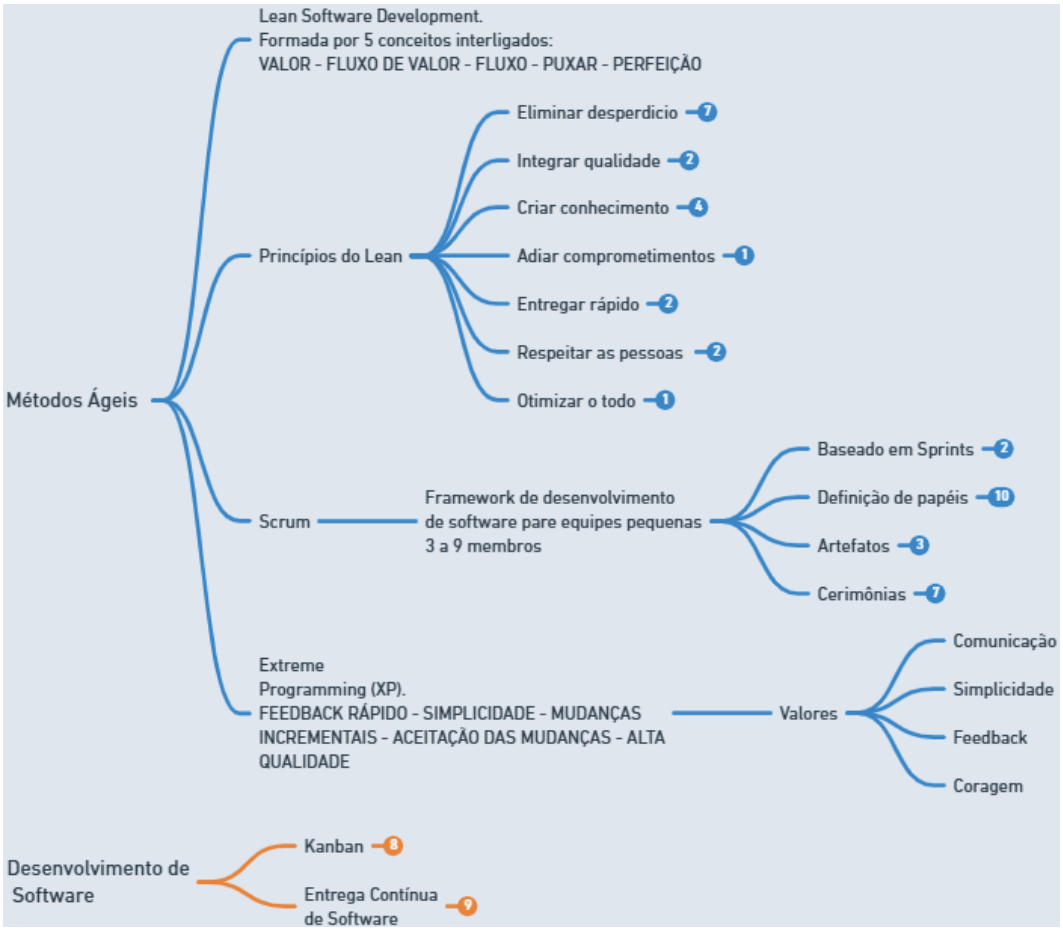
FONTE: O Autor (2025)

FIGURA 3 - MAPA CONCEITUAL: PRINCÍPIOS DA METODOLOGIA ÁGIL



FONTE: O Autor (2025)

FIGURA 4 - MAPA CONCEITUAL: MÉTODOS ÁGEIS



FONTE: O Autor (2025)

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

As disciplinas de Modelagem Ágil de Software (MAG1 e MAG2) tiveram como propósito mostrar como a modelagem, quando aplicada de forma enxuta e incremental, é parte fundamental do processo de desenvolvimento ágil. A filosofia *Domain Driven Design* (DDD) propõe que o software seja modelado a partir do entendimento profundo do domínio de negócio, permite que a modelagem esteja focada na resolução de problemas reais e na criação de soluções alinhadas às necessidades do cliente (Evans, 2003).

O projeto desenvolvido consistiu na criação de um **Sistema de Gestão de Condomínios**, apoiado em uma série de **artefatos** que antecederam a implementação do software. Entre eles destacam-se:

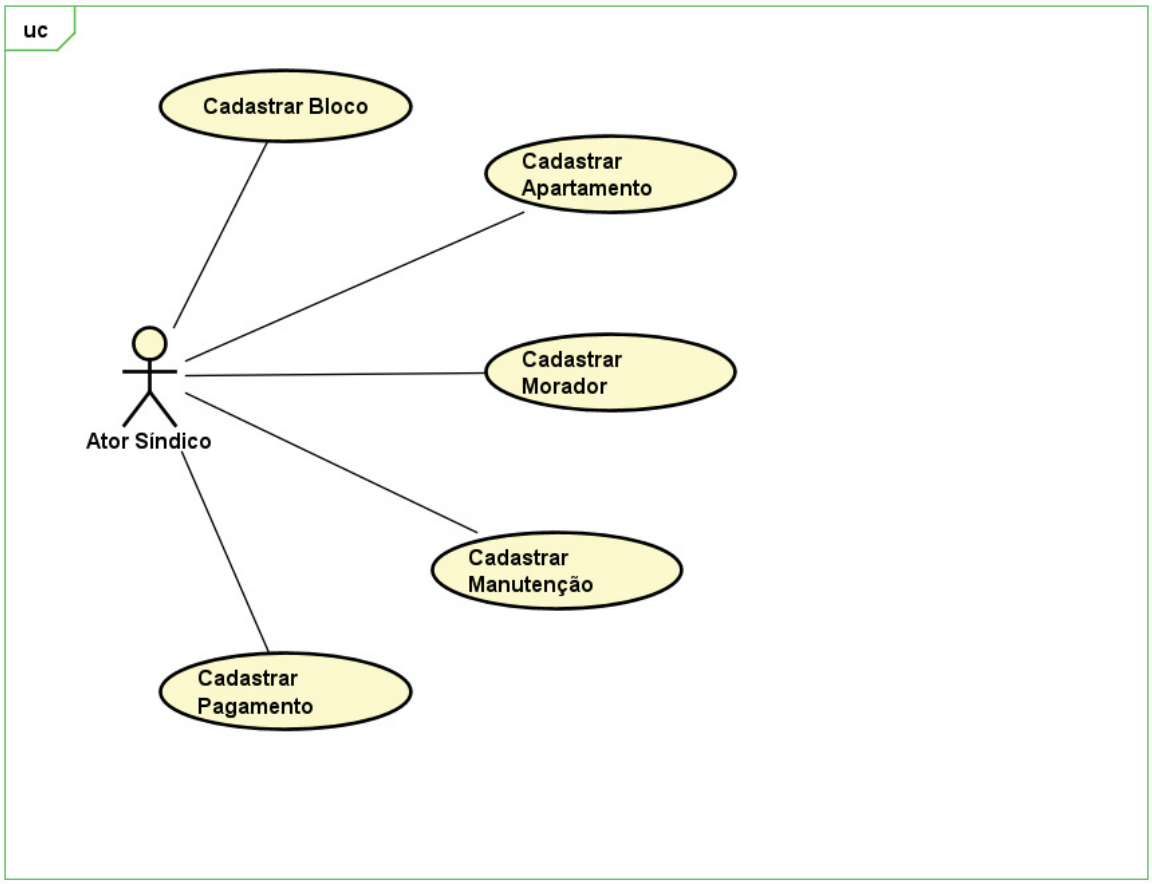
- **Elicitação de requisitos** com base em diferentes exercícios;
- **Diagramas de caso de uso nível 1**, representando as principais interações do sistema (como cadastro de apartamento, bloco, morador, pagamento e manutenção) (Figura 2);
- **Diagramas de caso de uso nível 2**, detalhando em nível de profundidade o relacionamento entres os demais casos de uso (Figura 3)
- **Histórias de usuário** escritas a partir da estrutura *Sendo [persona], quero [funcionalidade] para [benefício]* (Figura 4);
- **Prototipação de telas** e definição de funcionalidades esperadas, garantindo uma visão clara do produto antes da implementação (Figura 5);
- **Diagramas de Classes** (Figura 6), cuja representação permite visualizar a estrutura que dará forma ao sistema, a propriedades e métodos a serem implementados.
- **Diagramas de sequência**, elaborados com o padrão *Dado que [contexto], quando [ação], então [resultado]*, facilitando a especificação do comportamento do sistema (Figura 7);

Esses artefatos foram elaborados com o apoio do software *Astah UML* (Change, 2025), permitindo uma documentação visual que não engessou o processo, mas serviu como guia de comunicação entre os envolvidos. A prática reforçou que, nos métodos ágeis, a modelagem não deve ser extensa e burocrática, mas sim direcionada à resolução de problemas e à entrega de valor contínuo.

A contribuição de MAG1 e MAG2 ultrapassou a simples construção de diagramas: proporcionou a experiência de como alinhar requisitos, protótipos e modelagem a um processo iterativo, fortalecendo a integração com outras disciplinas do curso.

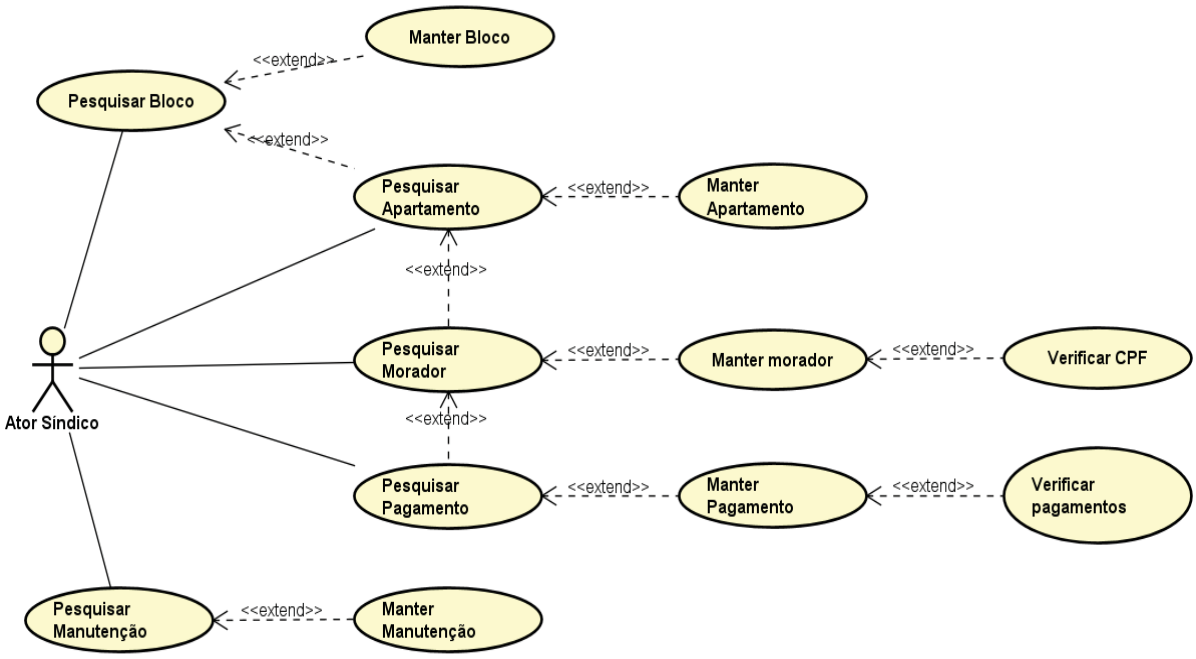
3.1 ARTEFATOS DO PROJETO

FIGURA 5 - DIAGRAMA DE CASO DE USO NÍVEL 1



FONTE: O Autor (2025)

FIGURA 6 - DIAGRAMA DE CASO DE USO NÍVEL 2



FONTE: O Autor (2025)

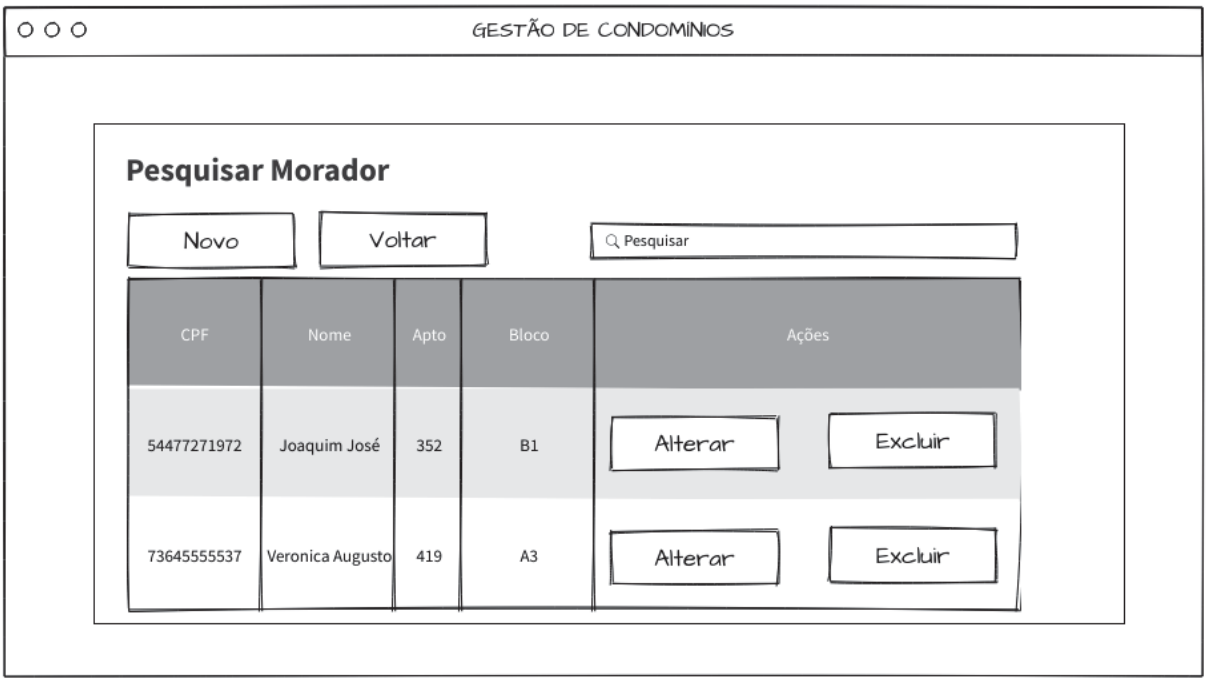
FIGURA 7 - HISTÓRIAS DE USUÁRIO

HU001 – PESQUISAR MORADOR

SENDO	**o síndico**
QUERO	**pesquisar os moradores**
PARA	**fazer manutenções nos seus dados**

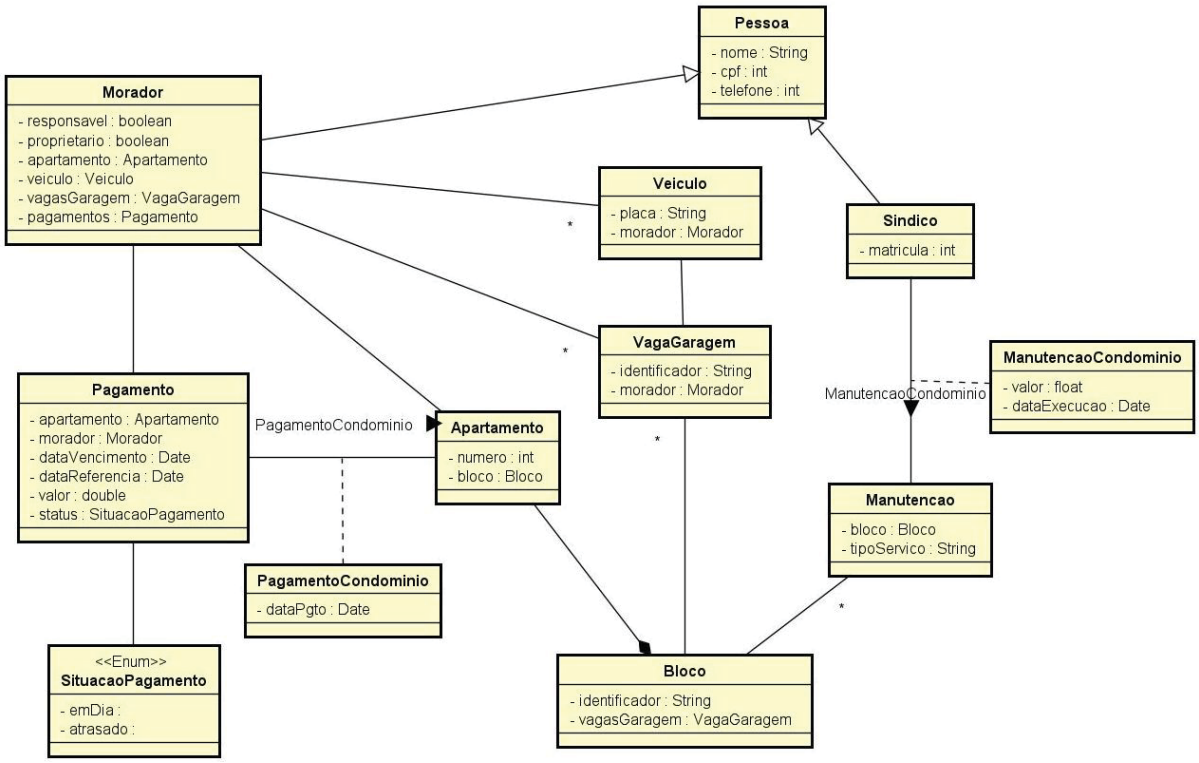
FONTE: O Autor (2025)

FIGURA 8 - DESENHO DE TELA



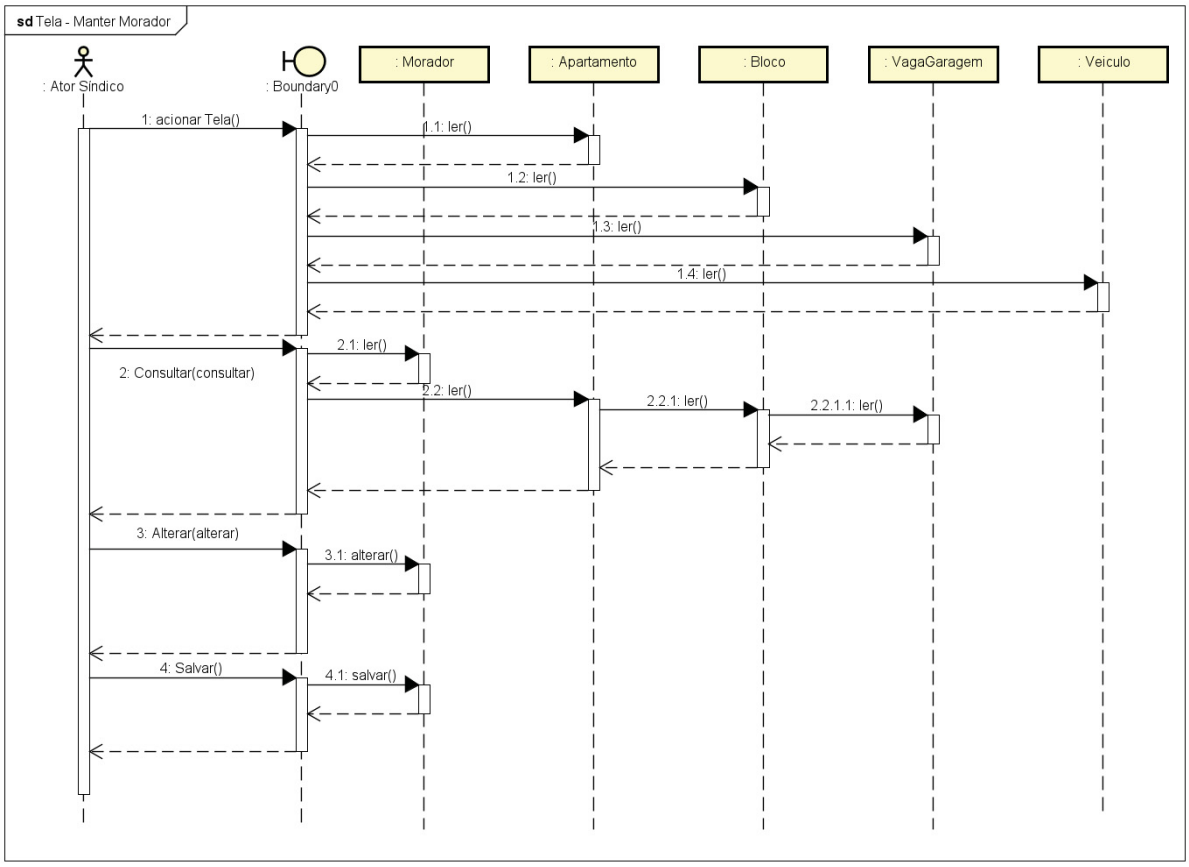
FONTE: O Autor (2025)

FIGURA 9 - DIAGRAMA DE CLASSES



FONTE: O Autor (2025)

FIGURA 10 - DIAGRAMA DE SEQUÊNCIA



FONTE: O Autor (2025)

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

As disciplinas de Gerenciamento Ágil de Projetos (GAP1 e GAP2) tiveram como foco apresentar e praticar técnicas para planejar, acompanhar e controlar projetos de software em ambientes complexos e dinâmicos. Inicialmente, foi abordado o *Project Management Body of Knowledge* (PMBOK) como referência tradicional de gestão, com suas áreas de conhecimento e os cinco grupos de processos (iniciação, planejamento, execução, monitoramento e encerramento), permitindo uma visão estruturada de como os projetos são conduzidos em modelos preditivos. Essa base serviu de contraponto para a introdução ao *framework Cynefin*, que auxilia na análise de contextos de complexidade. A partir dessa premissa, foram adotadas metodologias ágeis de concepção, como o *Design Thinking*, cuja abordagem centrada no usuário visa compreender detalhadamente as necessidades e desafios dos clientes por meio da empatia, definição de problemas, ideação, prototipagem e testes (Camargo e Ribas, 2019). Destaca-se também o *Design Sprints*, que oferece um processo estruturado para validação rápida de hipóteses e aceleração de decisões (*Design*, 2025), além do *Lean Inception*, que emprega práticas como é/não é, faz/não faz, identificação de personas e jornadas do usuário, objetivando alinhar o entendimento entre todos os envolvidos acerca do produto mínimo viável (MVP) (Camargo e Ribas, 2019).

O projeto prático de GAP1 resultou em um Plano de Release para um sistema de agenda de aulas e turmas de uma instituição de ensino, construído por meio de artefatos como mapa de funcionalidades, histórias de usuário, estimativas ágeis com *planning poker* e análise por ponto de função. A simulação do planejamento com o *Microsoft Planner* (Microsoft, 2025) reforçou a importância da transparência e da visibilidade das tarefas, além de promover o entendimento dos papéis, artefatos e cerimônias do *Scrum*. Também foram discutidos aspectos de contratação em projetos ágeis, destacando a necessidade de flexibilidade contratual para lidar com escopo variável.

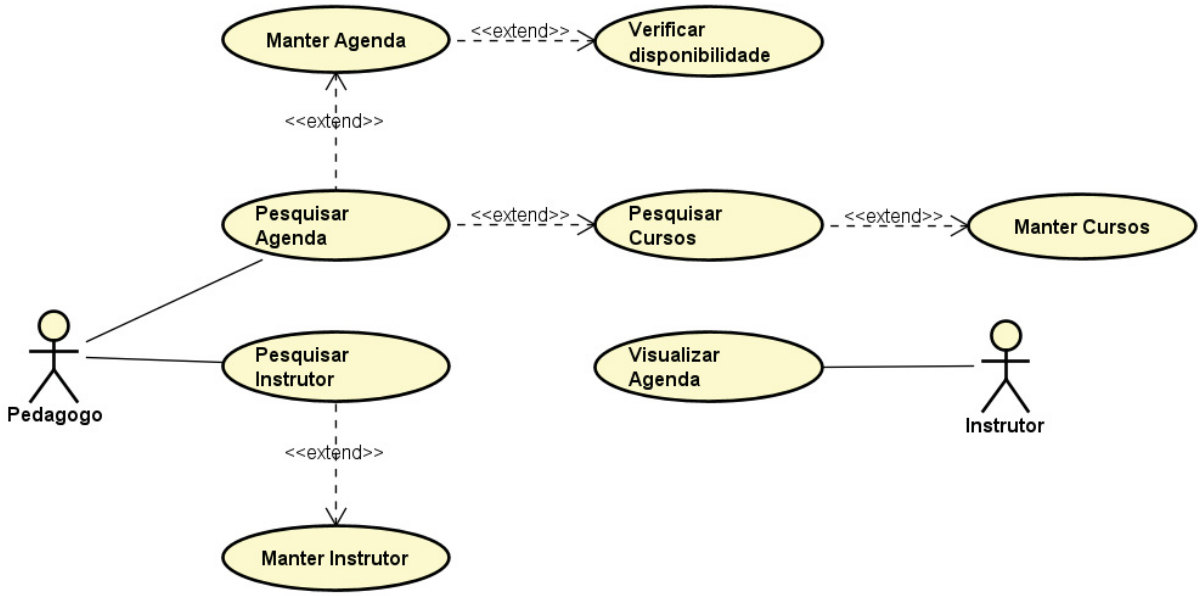
Na sequência, GAP2 aprofundou-se no gerenciamento visual de projetos a partir do Kanban, enfatizando práticas como a limitação do trabalho em progresso, a definição de métricas de fluxo e o uso de gráficos de acompanhamento, como o diagrama de fluxo cumulativo (CFD). O projeto envolveu a simulação do jogo (Figura

11) *Kanban Board Game* (Software, 2025), cujo resultado foi documentado com a pontuação final e um gráfico CFD (Figura 12), permitindo compreender na prática como a limitação de trabalho em progresso favorece previsibilidade e estabilidade no fluxo de desenvolvimento.

Essas disciplinas foram cruciais para estabelecer uma visão estratégica e operacional sobre como conduzir projetos de software em um contexto ágil. O aprendizado de GAP1 e GAP2 conecta-se diretamente às demais disciplinas do curso: apoia a modelagem ágil (MAG1 e MAG2) ao estruturar o backlog e planejar *releases*, fortalece as práticas de programação e testes (AAP e TEST) por meio de métricas e indicadores de qualidade e, por fim, integra-se com INFRA, ao preparar o terreno para estratégias de entrega contínua e monitoramento. Dessa forma, o gerenciamento ágil de projetos se mostrou não apenas uma disciplina de apoio, mas um eixo organizador de todo o ciclo de vida do software no curso.

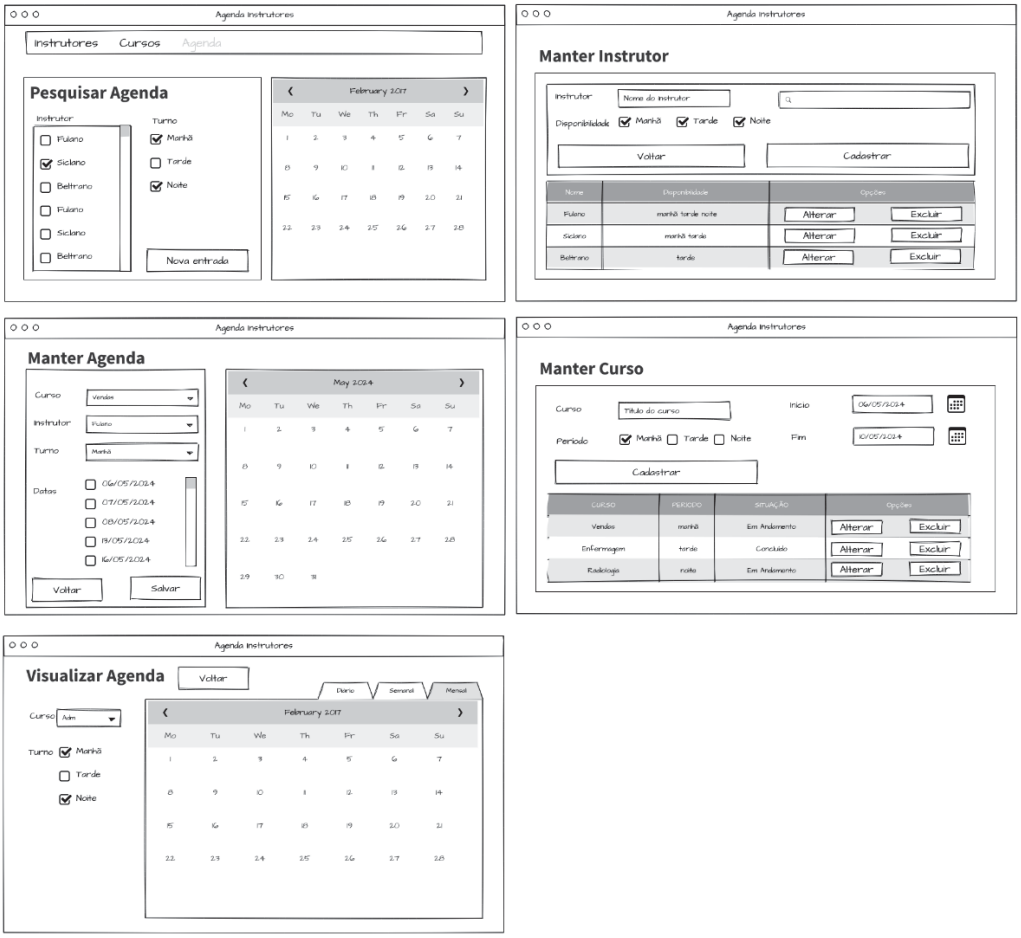
4.1 ARTEFATOS DO PROJETO

FIGURA 11 - CASO DE USO: AGENDA DE INSTRUTORES



FONTE: O Autor (2025)

FIGURA 12 - PROTÓTIPOS DE TELA - SISTEMA DE AGENDA



FONTE: O Autor (2025)

FIGURA 13 - PLANO DE RELEASE

Cálculo da Velocidade:

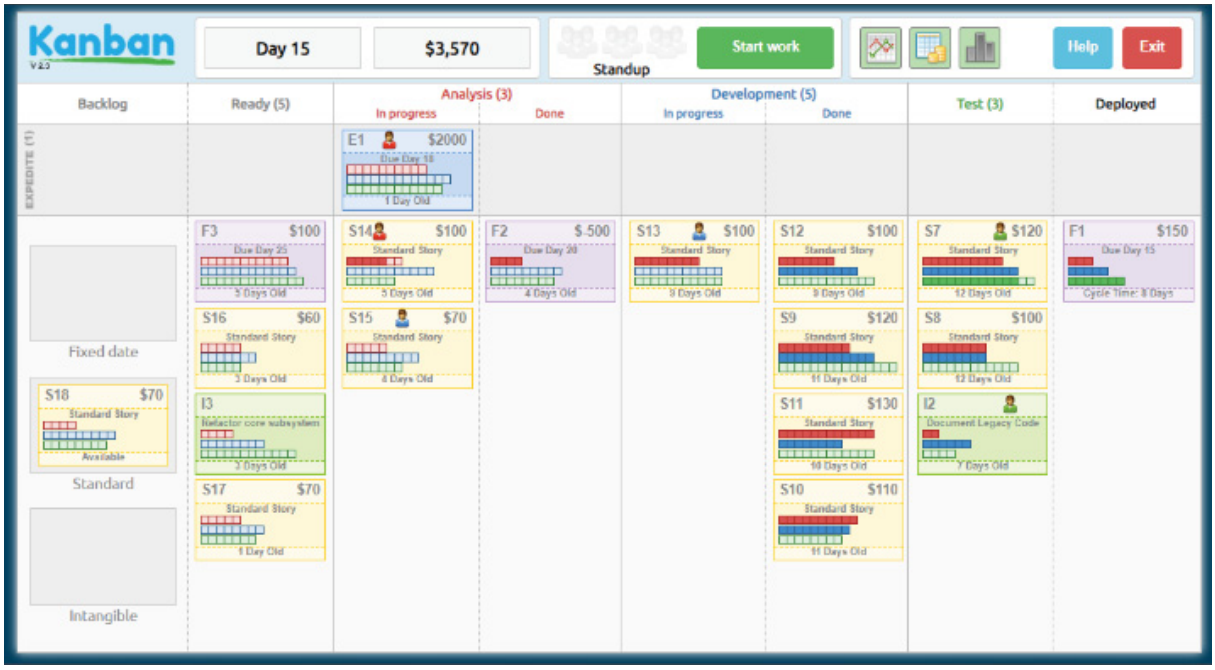
Horas disponíveis por dia:	4 horas	Tamanho da Sprint:	2 semanas
Horas disponíveis por Sprint:	40 horas	Velocidade:	5 pontos

Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3
Data Início: 06/05/2024	Data Início: 20/05/2024	Data Início: 03/06/2024
Data Fim: 17/05/2024	Data Fim: 31/05/2024	Data Fim: 14/06/2024
<HU004 - Manter Cursos> **SENDO**: Pedagogo **QUERO**: manter os dados dos cursos **PARA**: que seus dados fiquem atualizados **ESTIMATIVA** (3)	**<HU001 – Manter Agenda>** **SENDO**: Pedagogo **QUERO**: manter uma nova agenda **PARA**: que seus dados fiquem atualizados **ESTIMATIVA** (3)	**<HU003 – Pesquisar Agenda>** **SENDO**: Pedagogo **QUERO**: pesquisar a agenda dos instrutores **PARA**: consultar seus dados **ESTIMATIVA** (5)
<HU002 – Manter Instrutor> **SENDO**: Pedagogo **QUERO**: manter os dados do instrutor **PARA**: que seus dados fiquem atualizados **ESTIMATIVA** (2)	**<HU005 – Visualizar Agenda>** **SENDO**: Pedagogo / Instrutor **QUERO**: visualizar os dados da agenda **PARA**: manter-se informado sobre os eventos **ESTIMATIVA** (2)	

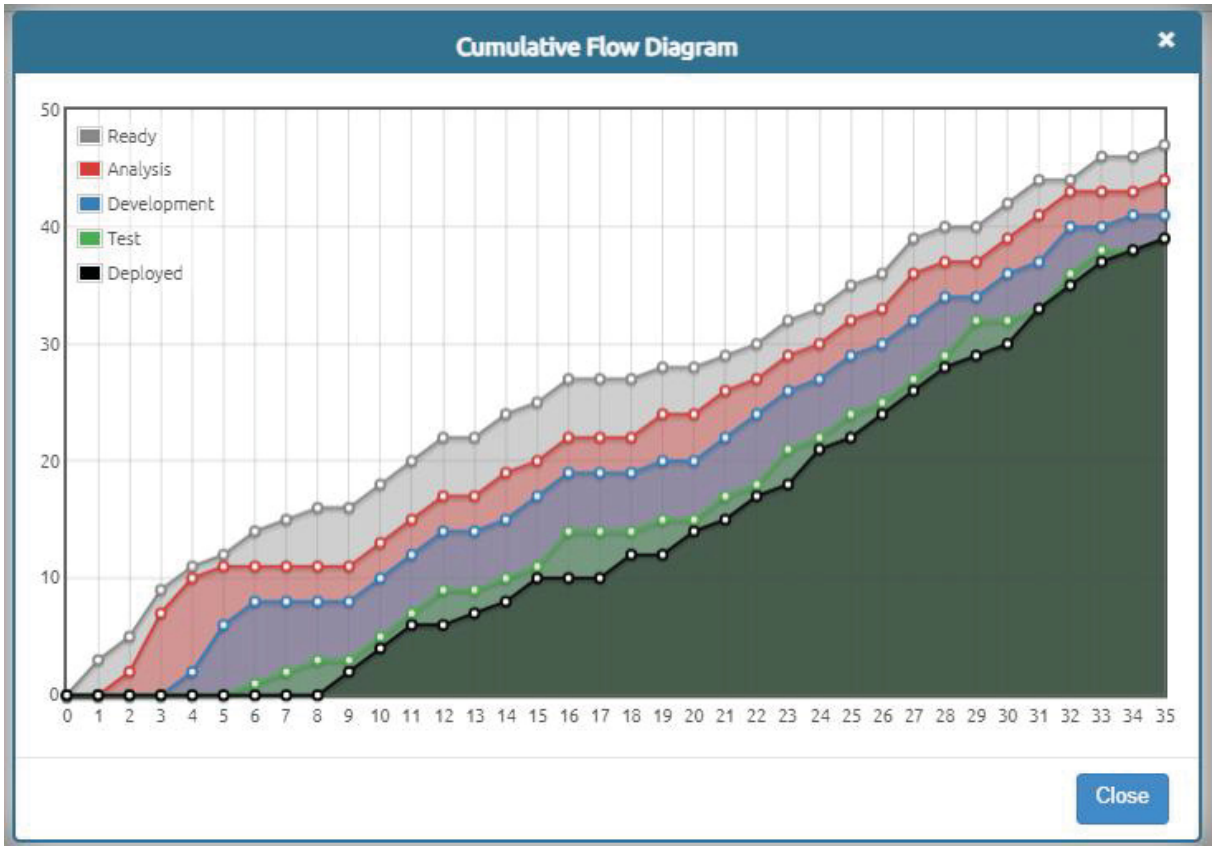
FONTE: O Autor (2025)

FIGURA 14 - SIMULAÇÃO DO JOGO KANBAN BOARD GAME



FONTE: O Autor (2025)

FIGURA 15 - DIAGRAMA DE FLUXO CUMULATIVO (CFD)



FONTE: O Autor (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação representou o ponto de partida técnico do curso, apresentando os fundamentos da programação orientada a objetos com Java e preparando a base para projetos mais complexos. Foram trabalhados conceitos como sintaxe da linguagem, estruturas de controle, tratamento de erros, arranjos, classes, objetos, herança, interfaces e exceções, sempre acompanhados por práticas orientadas a testes - *Test Driven Development* (TDD). A disciplina também incluiu a integração entre a aplicação Java e o banco de dados MySQL, utilizando JDBC e o padrão DAO, favorecendo a compreensão do mapeamento entre classes e tabelas.

Como atividade avaliativa, foi desenvolvido o *backend* de um sistema bancário simplificado, com funcionalidades de cadastro de clientes, criação de contas correntes e de investimento, além de operações de crédito e débito. O projeto exigiu a implementação de código (Figura 13), que atendesse a um conjunto de 42 casos de teste automatizados previamente definidos. A meta estabelecida foi a aprovação mínima de 40 testes simultaneamente, promovendo a aplicação direta dos princípios do desenvolvimento orientado por testes (TDD). Essa experiência reforçou a importância de validar continuamente o software e de alinhar o desenvolvimento ao conceito de qualidade embutida defendido nos métodos ágeis.

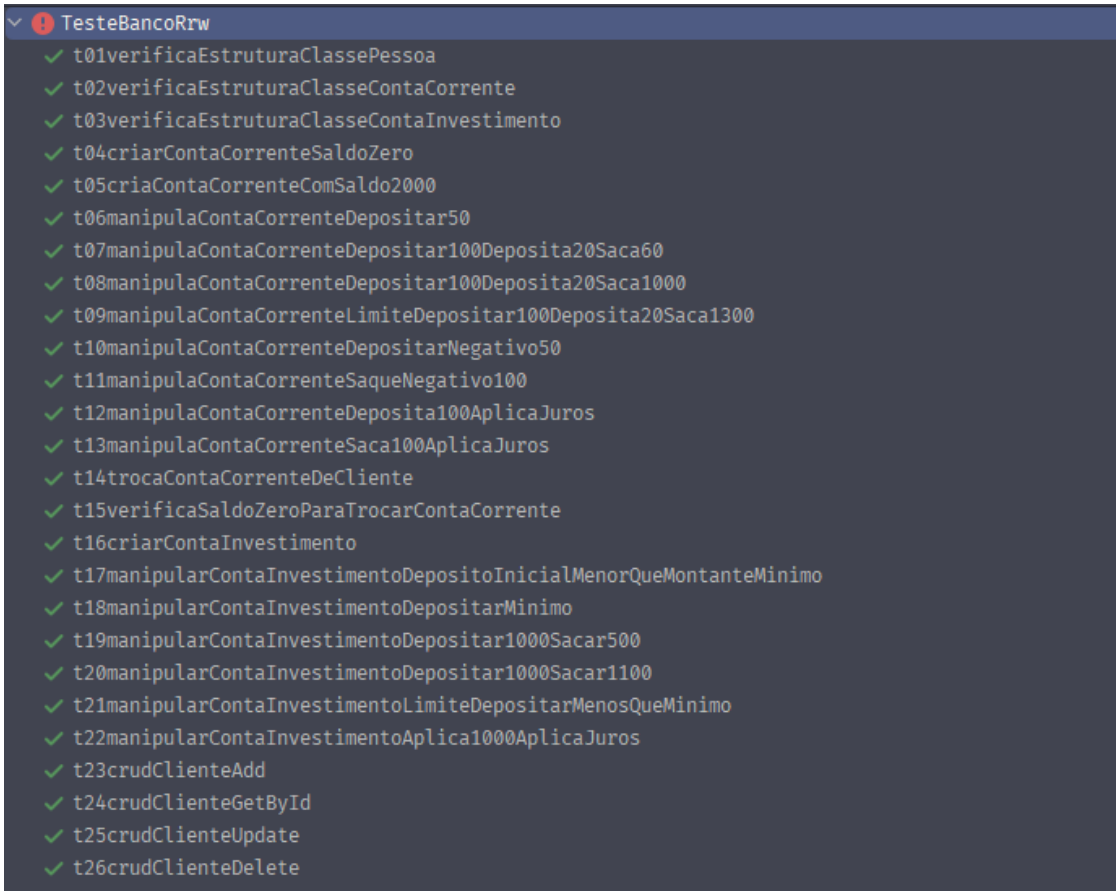
Além de fornecer a base técnica para a programação orientada a objetos, a disciplina também introduziu a cultura de testes, utilizando JUnit, como parte integrante do processo de desenvolvimento. Tal prática se mostrou decisiva para a integração com disciplinas posteriores, como Aspectos Ágeis de Programação (AAP), que expandiu a aplicação de boas práticas de codificação, e TEST, que consolidou a automação de testes como rotina de entrega contínua. Da mesma forma, a experiência com persistência em banco de dados antecipou conhecimentos fundamentais para a disciplina BD, garantindo coerência no processo de aprendizado.

Assim, Introdução à Programação destacou-se por unir conceitos teóricos fundamentais à aplicação prática em projetos reais, permitindo que o aprendizado da orientação a objetos fosse continuamente reforçado por exercícios de codificação e validação. A ênfase no TDD demonstrou como os testes podem guiar o desenvolvimento de maneira incremental, garantindo que cada nova funcionalidade fosse integrada de forma segura ao sistema. Essa abordagem permitiu a aproximação

de um processo iterativo e adaptativo, no qual o código evolui em pequenos ciclos de construção e verificação, em consonância com os princípios ágeis de qualidade e entrega de valor progressiva.

5.1 ARTEFATOS DO PROJETO

FIGURA 16 - RESULTADO DOS TESTES UNITÁRIOS COM JUNIT



FONTE: O Autor (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

A disciplina de Banco de Dados teve como propósito apresentar os fundamentos teóricos e práticos da área, abordando modelagem conceitual e lógica, linguagem SQL, além dos princípios de transações e concorrência. A aprendizagem foi conduzida de forma progressiva, combinando exercícios de modelagem no software brModelo 3.32 (Cândido, 2024), com a implementação de scripts em SQL nos sistemas gerenciadores de banco de dados relacionais.

O trabalho avaliativo foi estruturado em duas etapas complementares. A primeira parte consistiu no desenvolvimento de um sistema de controle de biblioteca, no qual foram elaborados os diagramas conceitual (Figura 14) e lógico (Figura 15), evidenciando entidades como obras, editoras, usuários, funcionários e departamentos. Essa etapa permitiu compreender as boas práticas de modelagem, normalização e definição de relacionamentos essenciais para a integridade dos dados, além de exercitar a criação de chaves primárias e estrangeiras e a implementação de restrições de integridade.

Na segunda parte, de tema livre, foi projetado um sistema de controle de estoque, cuja especificação exigia a utilização de diferentes tipos de relacionamentos (1:1, 1:N e N:N). O modelo lógico concebido (Figura 16) deu origem ao script SQL de criação de tabelas, incluindo *constraints*, definição de atributos obrigatórios e permissões de nulidade. Além disso, foram inseridos registros de exemplo em cada tabela, de modo a demonstrar na prática a aplicação dos relacionamentos modelados.

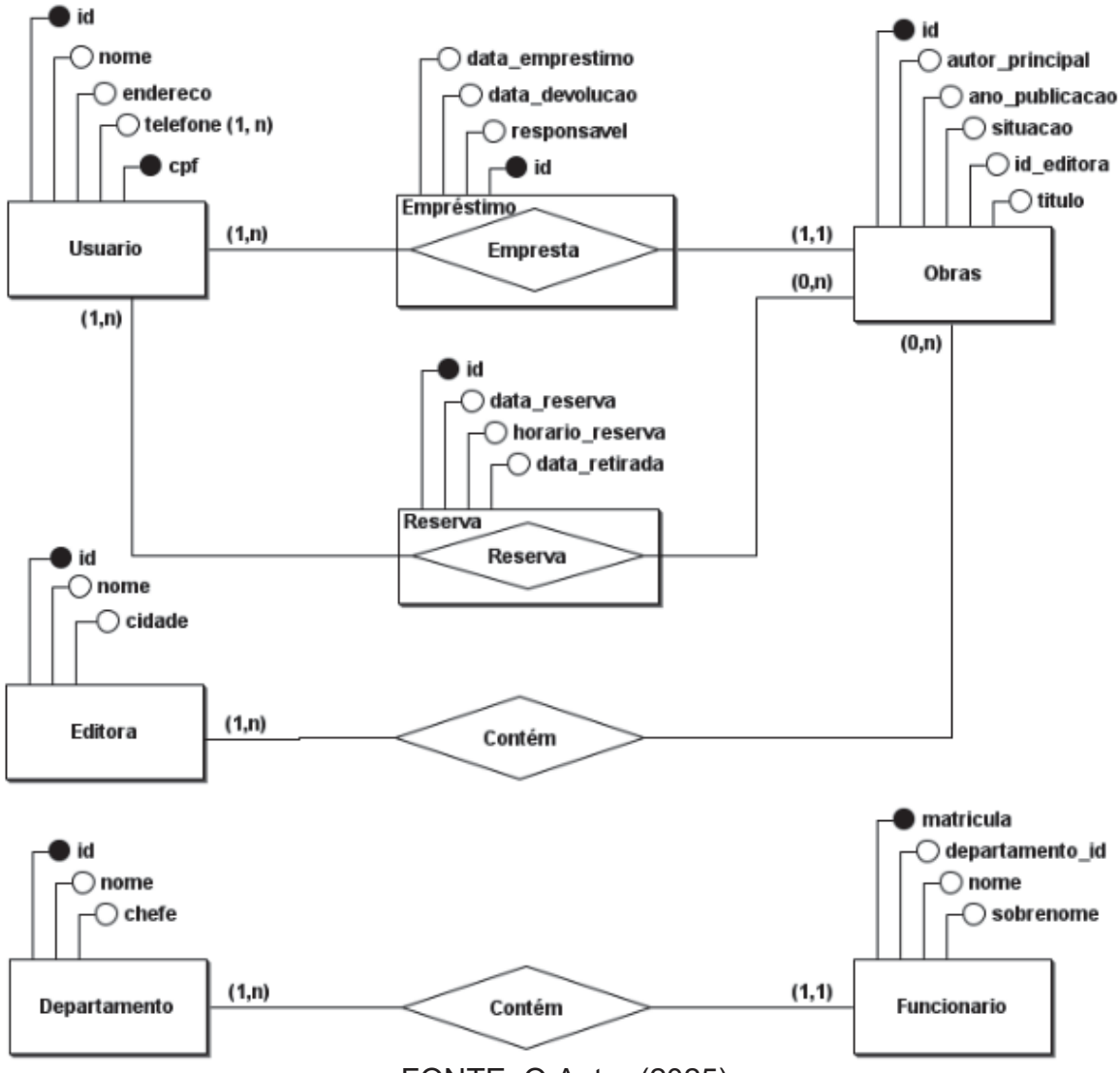
Para complementar a análise, foram realizadas diversas consultas avançadas em SQL, utilizando *joins* e a criação de *views*, reforçando a consistência e a integridade dos relacionamentos entre as entidades do sistema. Essas práticas permitiram validar o modelo lógico implementado, conforme ilustrado na Figura 17.

A construção dos dois sistemas mostrou-se complementar: o exercício de modelagem da biblioteca funcionou como base para o entendimento conceitual, enquanto o sistema de estoque consolidou a prática de modelagem e implementação em cenários mais próximos do desenvolvimento real de sistemas. Essa abordagem metodológica reforçou o papel dos bancos de dados como elemento estruturante para aplicações orientadas a objetos, promovendo o domínio dos processos de modelagem, normalização e persistência de dados.

O desenvolvimento de competências teóricas e práticas nessa área proporcionou uma base sólida, favorecendo a integração e evolução das atividades técnicas em projetos subsequentes, especialmente nos contextos de automação, desenvolvimento web e práticas ágeis de engenharia de software.

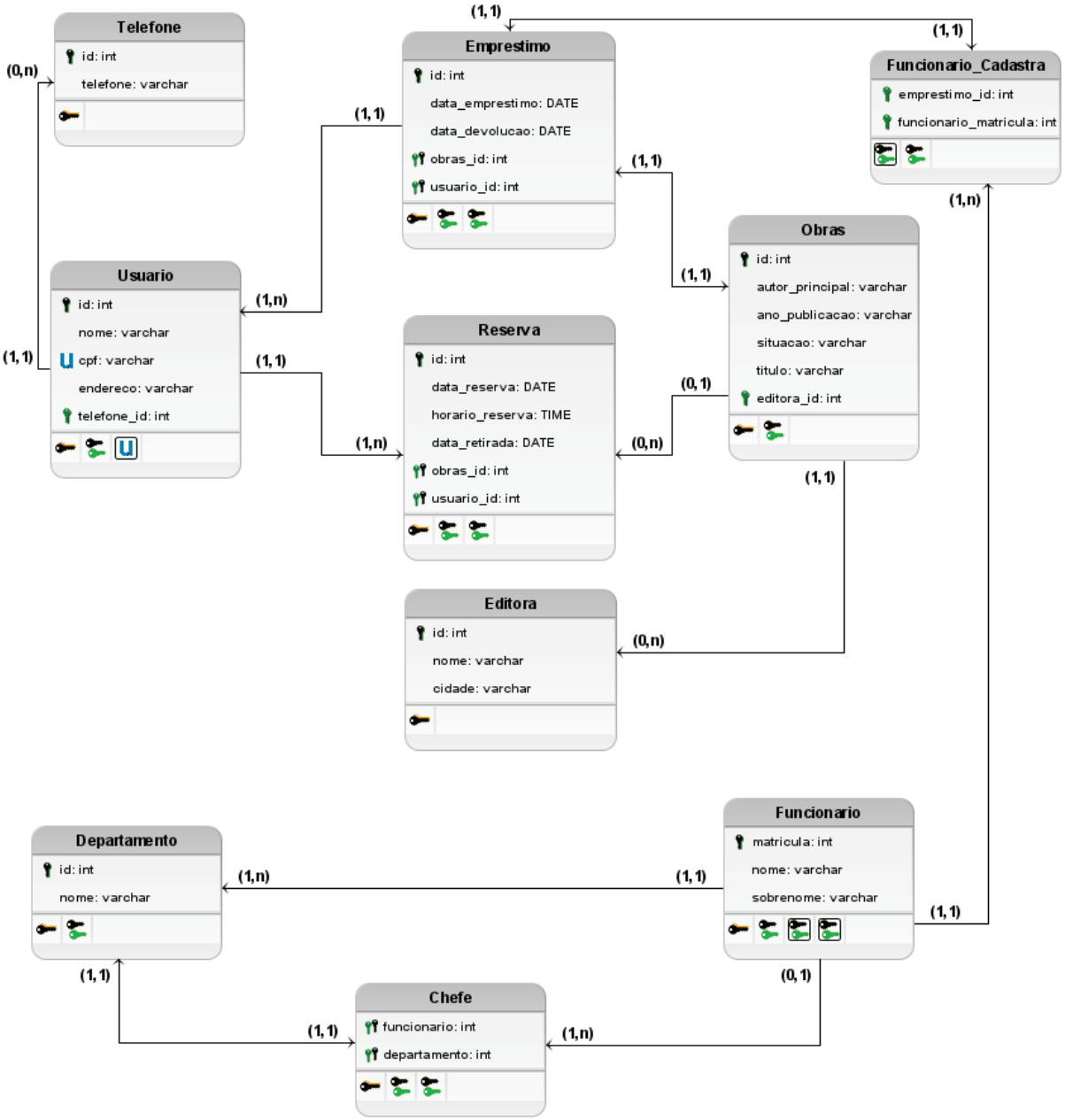
6.1 ARTEFATOS DO PROJETO

FIGURA 17 – MODELO CONCEITUAL SISTEMA BIBLIOTECA



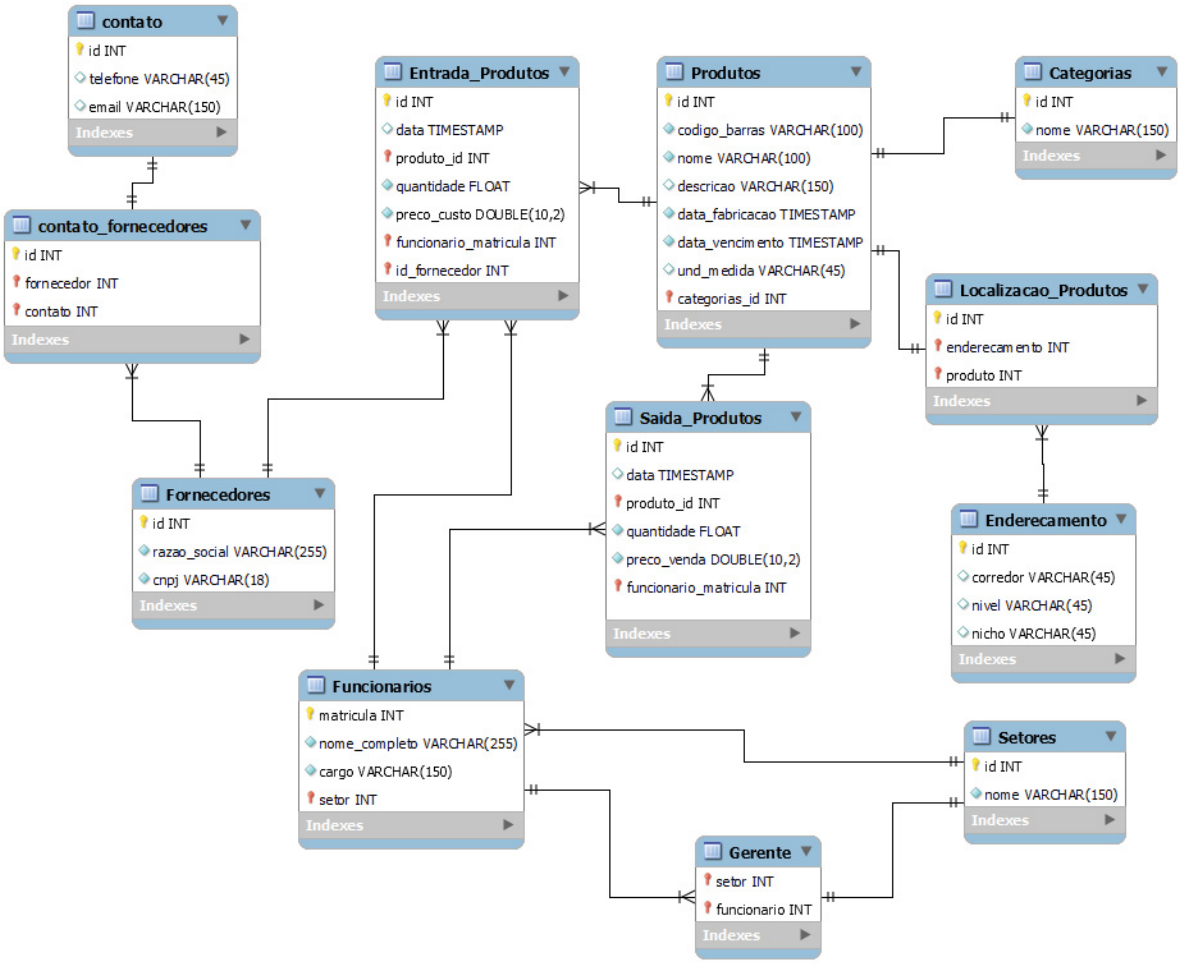
FONTE: O Autor (2025)

FIGURA 18 - MODELO LÓGICO SISTEMA BIBLIOTECA



FONTE: O Autor (2025)

FIGURA 19 – MODELO LÓGICO SISTEMA DE CONTROLE DE ESTOQUE



FONTE: O Autor (2025)

FIGURA 20 - CONSULTA SQL SISTEMA DE CONTROLE DE ESTOQUE

```
  -- Localização de produtos no estoque
  CREATE VIEW `controle_estoque`.`localizacao` AS
SELECT
        `p`.`codigo_barras` AS `codigo`,
    `p`.`nome` AS `produto`,
    `e`.`corredor` AS `corredor`,
    `e`.`nivel` AS `nivel`,
    `e`.`nicho` AS `nicho`
FROM
    ((`controle_estoque`.`enderecamento` `e`
    JOIN `controle_estoque`.`localizacao_produtos` `l` ON
((`l`.`id` = `e`.`id`)))
    JOIN `controle_estoque`.`produtos` `p` ON ((`p`.`id` =
`l`.`produto`)));
```

	codigo	produto	corredor	nivel	nicho
▶	0000567890123	Notebook	A1	N1	C1
	00065432109876	TV	A2	N1	C2
	00009876543210	Arroz	B1	N2	C3
	000010987654321	Refrigerante	B2	N2	C4
	765432109876543	Camiseta	C1	N3	C5

FONTE: O Autor (2025)

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como objetivo apresentar boas práticas no desenvolvimento de software, alinhadas à filosofia do *software craftsmanship* e aos princípios de código limpo (Martin, 2009), desenvolvimento orientado a testes (Beck, 2010), desenvolvimento orientado a comportamento (Smart, 2015) e à utilização da linguagem *Gherkin* para especificação de cenários de teste. O *Gherkin*, nesse contexto, foi trabalhado como a linguagem que permite formalizar os cenários de BDD de forma legível por usuários de negócio e interpretável por ferramentas de automação, no formato Dado – Quando – Então. Essa prática reforçou a integração entre requisitos, comportamento esperado e implementação técnica, favorecendo a comunicação entre áreas distintas do projeto. Ao longo da disciplina, foram abordados temas como nomenclatura significativa, escrita de funções coesas, legibilidade do código, manuseio de erros, limites de dependências, testes unitários, desenho emergente, concorrência e princípios da arquitetura limpa (Martin, 2019), com ênfase no SOLID.

Como artefato prático, foi realizada a refatoração de um algoritmo de ordenação do tipo *Bubble Sort*, com o objetivo de aplicar gradualmente os princípios do código limpo. Entre as modificações realizadas, destacaram-se:

- remoção de comentários redundantes, substituídos por código expressivo;
- reorganização dos métodos, com a função principal posicionada acima das demais;
- simplificação da assinatura do método *bubbleSort*, que passou a receber apenas um *array* como parâmetro;
- padronização da declaração de arrays (`int[] array`);
- substituição da variável genérica `n` por `size`, com escopo restrito ao necessário;
- renomeação da variável `trocado` para `swapped`, padronizando a linguagem em inglês;
- abstração da instrução `for` por meio do método `isSwapped`, reforçando a clareza semântica;
- declaração de variáveis diretamente em seus escopos de uso;

- implementação de laço `for` em modo *inline* no método de impressão de resultados.

Esse exercício evidenciou como ajustes incrementais e sucessivos de refatoração aumentam a legibilidade e a manutenibilidade do código, em consonância com a ideia de desenho emergente, definida por Kent Beck como um processo em que o design do software surge de pequenas melhorias constantes no código, ao invés de grandes decisões tomadas antecipadamente (Beck, 2004). Ao mesmo tempo, a prática demonstrou a relevância dos princípios de arquitetura limpa apresentados por Robert C. Martin, especialmente a responsabilidade única e a clareza de intenções, pois, como afirma o autor, “um módulo deve ter uma, e apenas uma, razão para mudar” (Martin, 2009 p.138), reforçando a importância da coesão e da simplicidade no desenho de sistemas.

No contexto do curso, a disciplina contribuiu diretamente para a qualidade técnica no processo de entrega de software, ao consolidar práticas que elevam a confiabilidade e reduzem o custo de manutenção. Sua integração com INTRO e TEST foi particularmente significativa: em INTRO, os fundamentos de programação orientada a objetos puderam ser revisitados sob uma ótica mais crítica, reforçando a escrita padronizada de código; em TEST, a prática de TDD e BDD se ampliou para a construção de cenários automatizados, ampliando a cobertura e garantindo que a evolução do sistema não comprometesse funcionalidades existentes. Assim, Aspectos Ágeis de Programação estabeleceu-se como um elo essencial entre teoria, prática e qualidade contínua, sustentando o desenvolvimento ágil com bases técnicas sólidas.

7.1 ARTEFATOS DO PROJETO

FIGURA 21 - CÓDIGO REFATORADO: BUBBLE SORT

```java
class BubbleSort {
    public static void main(String[] args)
    {
        int[] array = { 64, 34, 25, 12, 22, 11, 90 };
        bubbleSort(array);
        printResult(array);
    }

    static void bubbleSort(int[] array)  3 usages
    {
        boolean swapped;
        int size = array.length;
        for (int i = 0; i < size - 1; i++) {
            swapped = isSwapped(array, i);
            if (!swapped)
                break;
        }
    }

    private static boolean isSwapped(int[] array, int i) {  1 usage
        boolean swapped = false;
        int size = array.length;
        for (int j = 0; j < size - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
                swapped = true;
            }
        }
        return swapped;
    }

    static void printResult(int[] array)  1 usage
    {
        System.out.println("Array ordenado: ");
        for (int j : array) System.out.print(j + " ");
    }
}
```

FONTE: O Autor (2025)

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

As disciplinas de Desenvolvimento Web 1 e 2 tiveram como objetivo capacitar na construção de aplicações modernas utilizando o framework Angular (Angular, 2025), correlacionando conceitos fundamentais da programação web com práticas de desenvolvimento ágil. Inicialmente, em WEB1, foram abordados os conceitos introdutórios da linguagem *TypeScript* (Typescript, 2025), estruturas de HTML, criação de formulários e utilização de bibliotecas visuais como Bootstrap (Bootstrap, 2025) e Material Design (Material, 2025). Esse primeiro contato foi consolidado por meio da elaboração de pequenos projetos, culminando no trabalho final, que consistiu na implementação de operações de criação, leitura, atualização e remoção de dados das entidades Aluno e Curso — utilizando Angular, *Local Storage* e rotas devidamente configuradas para permitir inserção, edição, listagem e remoção de registros

Na sequência, WEB2 ampliou os conhecimentos, introduzindo recursos avançados como programação reativa, consumo de dados por meio de interfaces de programação de aplicações *(API's REST)* e integração com *backend* em Spring Boot (Spring, 2025) e banco de dados PostgreSQL (Postgresql, 2025). O trabalho final dessa disciplina expandiu a aplicação construída em WEB1 para contemplar três CRUDs: Aluno, Curso e Matrícula. A matrícula relacionava alunos e cursos, armazenando informações adicionais como data e nota final, reforçando a prática de modelagem relacional e integração com banco de dados. O sistema incluiu menu de navegação, uso obrigatório de Bootstrap ou Material Design, além da implementação de telas de listagem, inserção, edição e remoção com confirmação, bem como visualização em modal

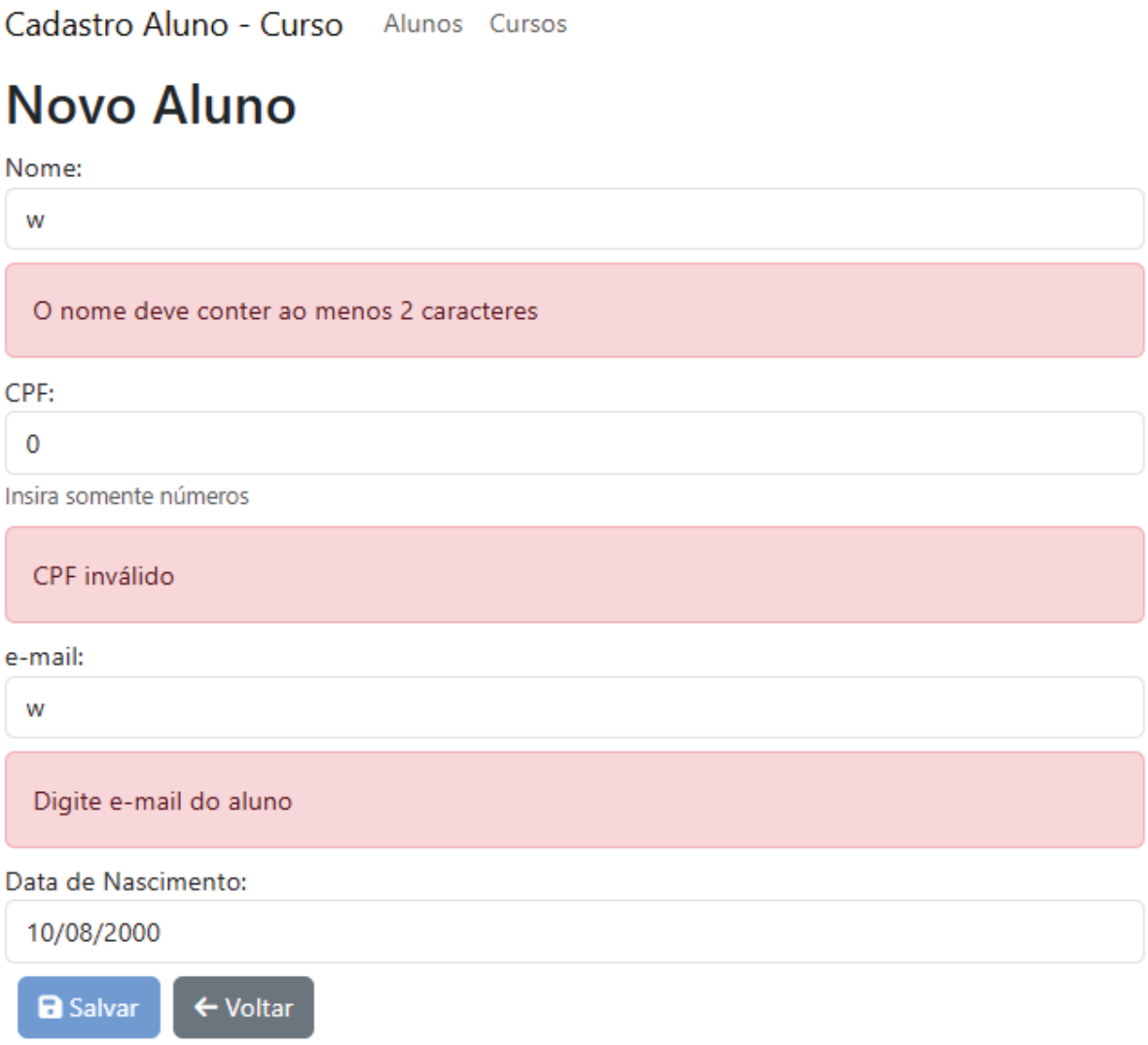
A evolução entre WEB1 e WEB2 evidenciou a transição de uma aplicação estática e baseada em armazenamento local para um sistema completo, integrado a banco de dados e *backend* estruturado, em linha com os princípios do desenvolvimento incremental. Os projetos desenvolvidos demonstraram, na prática, a importância da separação de responsabilidades entre *frontend* e *backend*, além de consolidarem o papel das *APIs* no suporte a arquiteturas escaláveis.

No contexto geral do curso, as disciplinas de desenvolvimento web contribuíram de forma decisiva para a integração dos aprendizados prévios: a modelagem realizada em MAG1 e MAG2 forneceu a base para a estruturação de entidades e relacionamentos, enquanto os conceitos de BD foram aplicados na

persistência dos dados. Além disso, práticas vistas em AAP e TEST foram fundamentais para garantir a qualidade do código e ampliar a cobertura de testes, reforçando a entrega de software com valor contínuo e confiabilidade técnica.

8.1 ARTEFATOS DO PROJETO

FIGURA 22 - TELA DE CADASTRO DE ALUNO COM VALIDAÇÕES



FONTE: O Autor (2025)

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

A disciplina de UX no Desenvolvimento Ágil de Software teve como foco apresentar os fundamentos da experiência do usuário (UX) e sua relação direta com os princípios dos métodos ágeis. Inserida em um contexto no qual a entrega de valor ao cliente é central, a disciplina evidenciou que projetar interfaces usáveis, acessíveis e alinhadas às necessidades reais dos usuários é parte integrante do ciclo ágil. Foram discutidos conceitos como design de experiência do usuário (UX), design de interfaces (UI), arquitetura da informação (IA) e design de interação (IXD), apoiados em normas internacionais como a ISO 9241-210:2010 e a ISO/IEC 25000:2014, além de métodos de pesquisa de usuários, criação de personas, jornadas, *wireframes* e prototipação. Permitiu aprofundar a compreensão sobre como elementos visuais e interativos influenciam a experiência do usuário final, alinhando-se com o objetivo de criar interfaces intuitivas, acessíveis e visualmente atraentes a partir dos conceitos e fundamentos de design visual, tipografia, cores, ícones e grids de layout.

A prototipação e os testes de usabilidade foram destacados como práticas essenciais dentro da disciplina, pois possibilitam a validação iterativa de hipóteses antes do desenvolvimento definitivo do produto. Essa abordagem está em consonância com a filosofia ágil de entregas incrementais e feedback contínuo, reduzindo riscos e assegurando que as funcionalidades implementadas estejam alinhadas às expectativas do usuário final. Assim, a disciplina reforçou que o design não deve ser entendido como uma etapa isolada, mas como parte de um processo de construção colaborativo e evolutivo.

O artefato prático consistiu no desenvolvimento do site Guia Náutico Litoral do Paraná, cujo objetivo foi centralizar informações sobre transporte náutico entre as ilhas do litoral norte do Paraná, incluindo rotas, horários, contatos de barqueiros e dicas turísticas. O trabalho contemplou:

- registro de *wireframes* de baixa fidelidade para orientar o protótipo (Figura 20).
- escolha da tipografia (Figura 21), priorizando legibilidade e acessibilidade;
- elaboração de telas prontas (Figura 22)
- definição de uma paleta de cores (Figura 23) inspirada em elementos náuticos (azul predominante, amarelo para destaques e verde oceano para contraste);
- definição de layouts baseados em *cards* e colunas, visando fluidez e padronização da navegação;

O protótipo foi submetido a testes de usabilidade com um usuário real, permitindo coletar feedbacks importantes: dificuldade na localização do menu de busca, ausência de destaque no item ativo do menu, falta de feedback visual em botões e semelhança excessiva entre a página inicial e o blog. Essas observações permitiram validar hipóteses de design e demonstraram a relevância da prototipagem iterativa para evolução da solução.

No contexto do curso, a disciplina de UX se mostrou fundamental para integrar o aspecto técnico e humano no desenvolvimento de software. Ao lado de disciplinas como MAG1 e MAG2, que tratam da modelagem de requisitos, e WEB1/WEB2 e MOB1/MOB2, que exploram a implementação de interfaces, a UX garantiu que o processo de entrega de software mantivesse o foco na experiência do usuário final. Essa integração fortaleceu a visão de que projetos ágeis devem unir qualidade técnica e valor percebido pelo cliente, reforçando a importância da usabilidade e da acessibilidade como pilares do desenvolvimento iterativo e incremental.

9.1 ARTEFATOS DO PROJETO

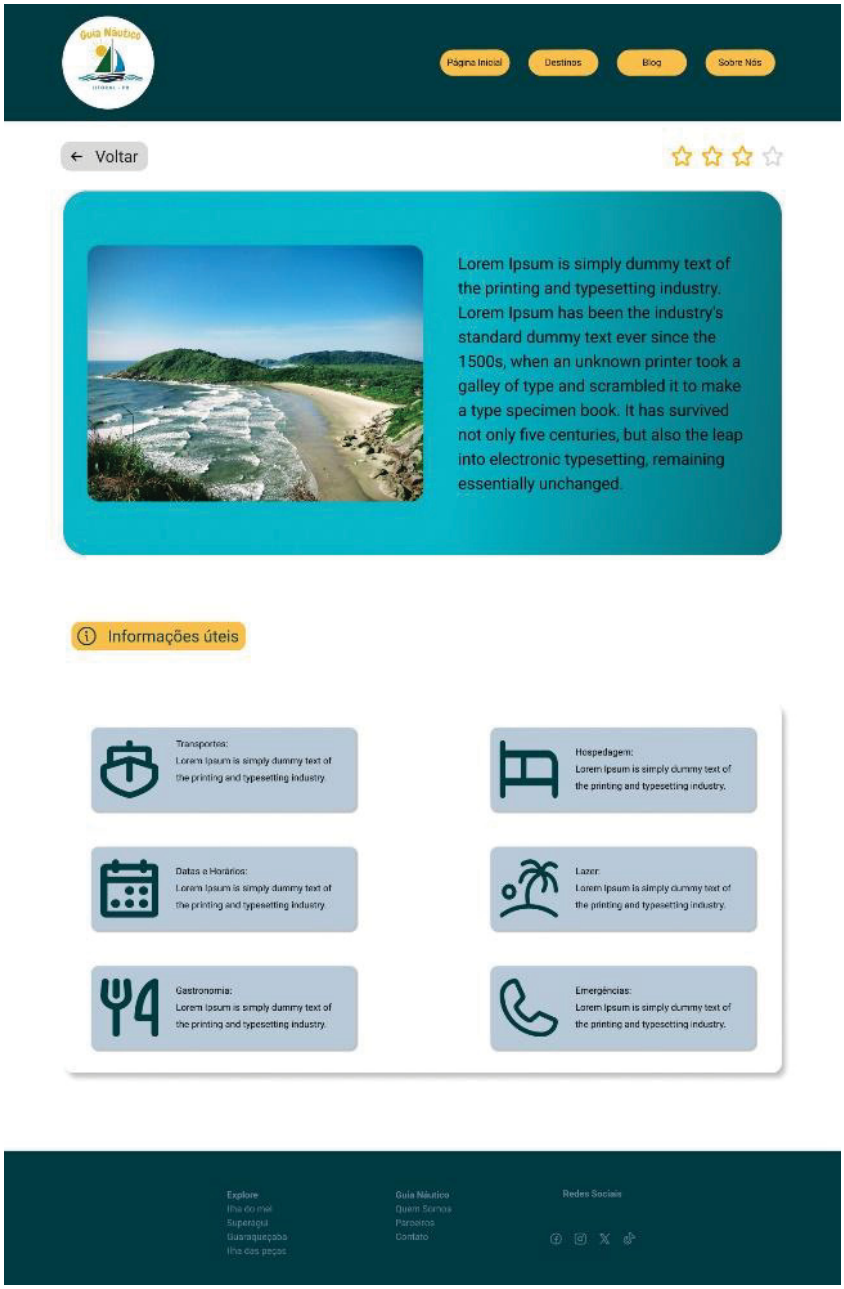
FIGURA 23 - WIREFRAME DE BAIXA FIDELIDADE

FONTE: O Autor (2025)

FIGURA 24 - TIPOGRAFIA

FONTE: O Autor (2025)

FIGURA 25 - PÁGINA DE SERVIÇOS



FONTE: O Autor (2025)

FIGURA 26 - PALETA DE CORES

FONTE: O Autor (2025)

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

As disciplinas Desenvolvimento Mobile 1 e 2 foram estruturadas para proporcionar o domínio da construção de aplicações Android nativas, empregando Kotlin (Kotlin, 2025) e Android Studio (Android, 2025). O conteúdo abordou desde componentes básicos até integração com serviços externos e persistência de dados por meio de exercícios práticos progressivos. Seguindo um modelo de aprendizagem ativa, aprender fazendo, em que o estudante constrói o conhecimento a partir da execução prática de tarefas reais, reforçando a compreensão de conceitos teóricos por meio da experimentação direta (Schank, 1995) foi possível estabelecer uma relação consistente entre teoria e prática no contexto do desenvolvimento móvel.

Em MOB1, a evolução partiu de pequenos aplicativos de experimentação — como Frases do Dia, Cambio App, Number Guess e Churrascômetro — até projetos mais elaborados, como a calculadora de IMC (multi-activity) e o app de receitas de drinks (listas e ListView). O trabalho final de MOB1 resultou no desenvolvimento do FinApp, um aplicativo de finanças pessoais para cadastro e gerenciamento de receitas e despesas, que permitia classificá-las como débitos ou créditos e exibi-las em um extrato. Embora os dados fossem armazenados apenas em memória, a aplicação representou uma etapa significativa de integração de conceitos de interface, modularização e manipulação de dados.

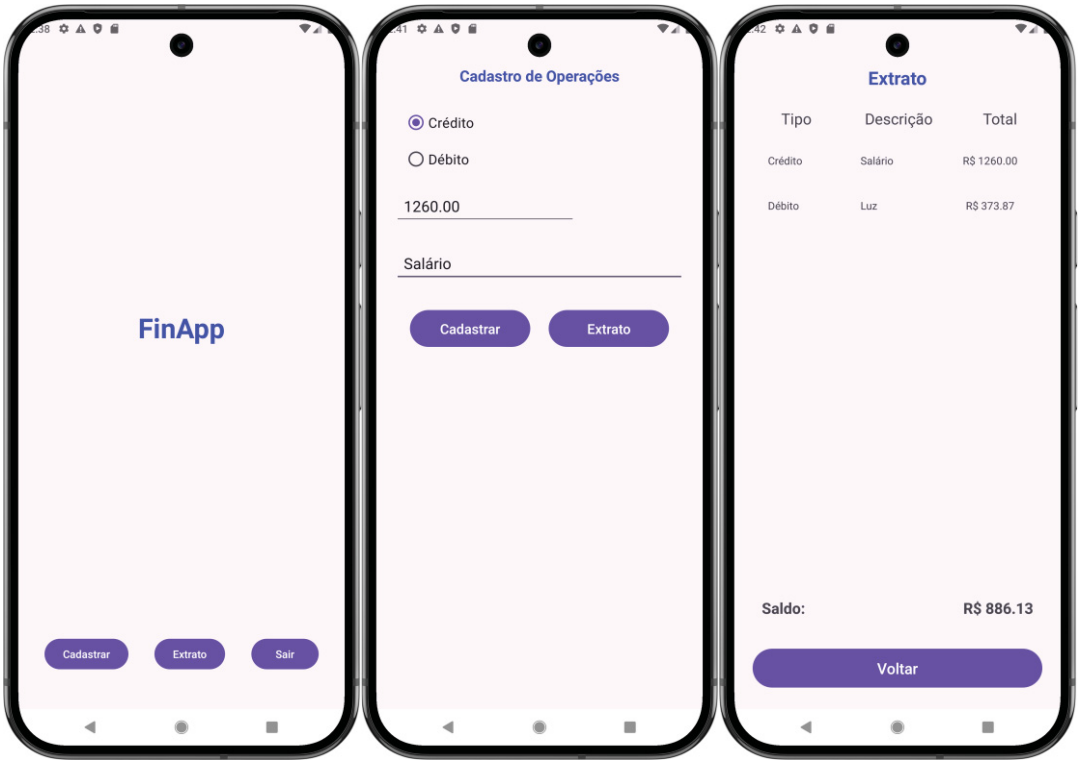
A disciplina MOB2 deu continuidade a esse aprendizado, introduzindo persistência de dados (SQLite, DAO), corrotinas para operações assíncronas e integração com *web services* via Retrofit. Após exercícios intermediários com APIs públicas, o projeto final consolidou esses conceitos no desenvolvimento de um aplicativo completo integrado à Harry Potter API (Hp-Api, 2025) que permitia listar personagens por *id*, professores de Hogwarts e estudantes de cada casa. A solução consolidou a integração entre persistência local, comunicação com *web services*, uso de corrotinas e arquitetura modular, representando uma entrega com características de produto funcional.

De forma integrada, MOB1 e MOB2 evidenciaram como o desenvolvimento mobile pode ser conduzido de forma iterativa e incremental, partindo de pequenos experimentos locais até chegar a aplicações completas e conectadas. Além disso, reforçaram a importância da validação contínua e da entrega de valor em ciclos curtos, características centrais da filosofia ágil. Os aprendizados dessas disciplinas se

relacionam diretamente com WEB1/WEB2, no consumo de APIs e CRUDs; BD, pela persistência de dados; e AAP/TEST, pela ênfase em práticas de qualidade, cobertura de testes e escrita clara de código.

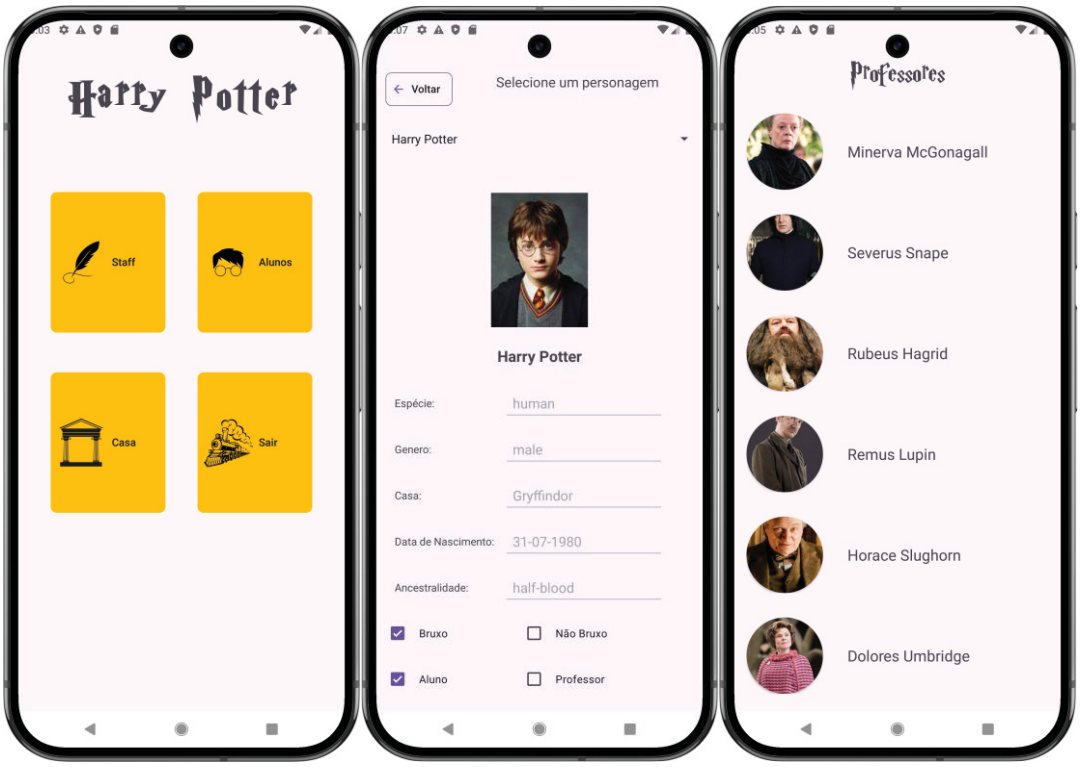
10.1 ARTEFATOS DO PROJETO

FIGURA 27 - TELAS DO APLICATIVO FINAPP



FONTE: O Autor (2025)

FIGURA 28 - TELAS DO APLICATIVO HARRY POTTER



FONTE: O Autor (2025)

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

A disciplina de Infraestrutura para Desenvolvimento e Implantação de Software teve como objetivo apresentar os fundamentos Ciclo de Vida do Desenvolvimento de Software (Software Development Life Cycle - SDLC) e do Desenvolvimento e Operações - *DevOps*. Segundo Humble et al. (2020), o *DevOps* promove práticas que integram desenvolvimento e operações, viabilizando entregas contínuas e maior confiabilidade no ciclo de vida do software. Foram trabalhados temas como versionamento de código com *Git* (Git, 2025), métricas de desempenho (DORA Metrics), maturidade de processos, conteinerização com *Docker* (Docker, 2025) e *Kubernetes* (Kubernetes, 2025), além da automação de *pipelines* de Integração Contínua e Entrega Contínua (CI/CD) utilizando ferramentas como *GitLab* (Gitlab, 2025) e *Jenkins* (Jenkins, 2025).

O trabalho final consistiu na implantação de um container *Docker* (Figura 26) baseado na imagem `dfwandarti/gitlab_jenkins:3`, configurado para publicar as portas de acesso 22, 80, 443 e 9091. A atividade incluiu a inicialização do container com identificação personalizada, acesso ao *GitLab* via credenciais armazenadas no próprio container (Figura 27), e a realização de operações práticas de versionamento, como `commit` e `push` em um projeto pré-configurado (Figura 28). A execução dessa prática consolidou os conceitos de infraestrutura como código, observabilidade, *storage* e replicação de serviços, enfatizando como a automação de processos de *build*, integração e entrega contínua contribui para a redução de falhas e para a aceleração do ciclo de desenvolvimento.

No contexto do curso, essa disciplina possui forte conexão com MADS, que apresentou os princípios de entrega contínua, e com AAP e TEST, que destacaram a importância de qualidade técnica e cobertura de testes. Também dialoga diretamente com os projetos de WEB e MOBILE, ao demonstrar como aplicações construídas em etapas anteriores podem ser integradas em pipelines automatizados, testadas e disponibilizadas em ambientes de produção. Dessa forma, INFRA reforçou o papel do *DevOps* como extensão prática do desenvolvimento ágil, assegurando que a entrega de software seja sustentável, escalável e de valor para o cliente.

11.1 ARTEFATOS DO PROJETO

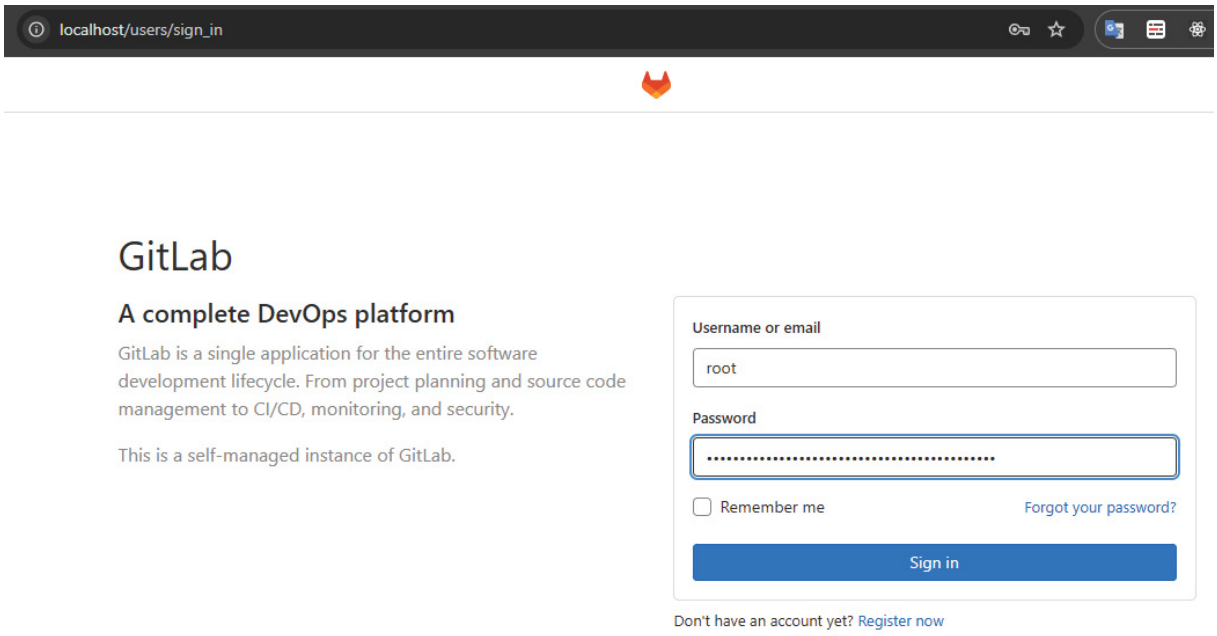
FIGURA 29 - CONSTRUÇÃO DO CONTAINER COM DOCKER

```
C:\Users\wagne>docker run --name 40001016398E1 --detach --hostname localhost --publish 443:443
--publish 80:80 --publish 22:22 --publish 9091:9091 --shm-size 256m dfwandarti/gitlab_jenkins:3

Unable to find image 'dfwandarti/gitlab_jenkins:3' locally
3: Pulling from dfwandarti/gitlab_jenkins
d7bfe07ed847: Already exists
25bb9590d9c9: Already exists
d46df4aa614d: Already exists
c695fd85ed38: Already exists
3f36ee87421e: Already exists
c0436bfc19a7: Already exists
061a5147aada: Already exists
5d4e92472abe: Already exists
5cf33ab37f3b: Already exists
b42364568e18: Already exists
Digest: sha256:c8f217311114d5e795315534e827d13a09a2c63d8956eedac53cca475c6e3780
Status: Downloaded newer image for dfwandarti/gitlab_jenkins:3
d4255b838c85c8a1931bcb9c402e2bc00ffac0c98c666c66de02d61e645e98b2
```

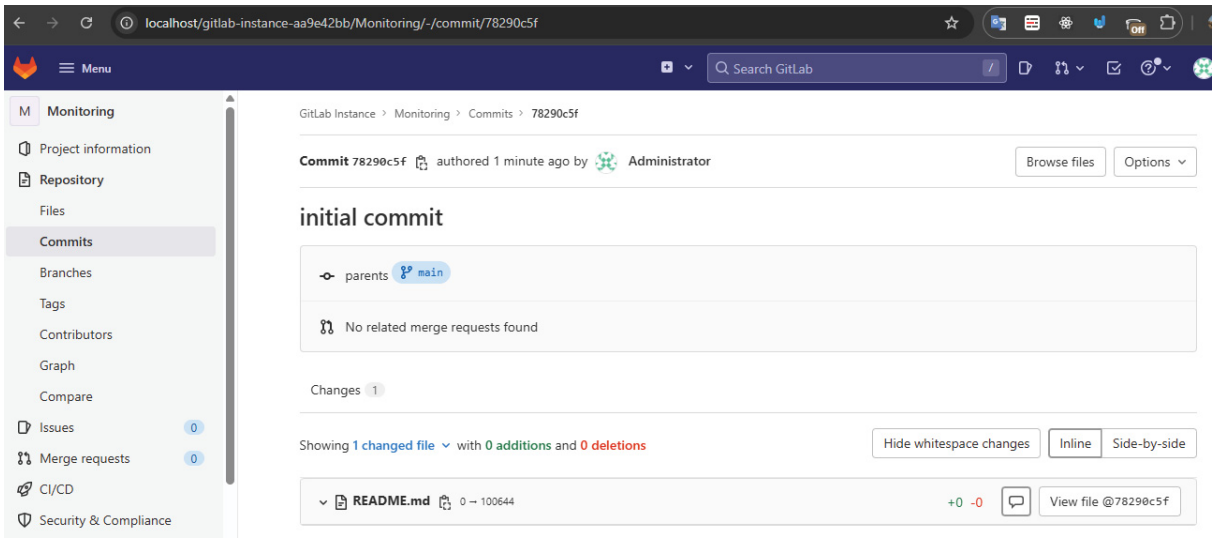
FONTE: O Autor (2025)

FIGURA 30 - CREDENCIAIS DE ACESSO AO CONTAINER



FONTE: O Autor (2025)

FIGURA 31 - COMMIT INICIAL COM GITLAB



FONTE: O Autor (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados teve como foco apresentar o papel dos testes no ciclo de vida do desenvolvimento de software, enfatizando sua importância na garantia de qualidade e na entrega contínua de valor — pilares centrais do desenvolvimento ágil. Foram abordados conceitos essenciais como os tipos de testes, os papéis do testador, o modelo da Pirâmide de Testes (Fowler, 2012), e a estratégia *Shift Left*, que propõe a antecipação das atividades de teste para as etapas iniciais do desenvolvimento, reduzindo custos e riscos associados a falhas tardias.

De acordo com Fowler (2012), a Pirâmide de Testes recomenda uma distribuição equilibrada entre testes unitários, de integração e de interface, priorizando testes unitários por serem mais rápidos e baratos. O princípio *Shift Left*, aplicado em métodos ágeis e *DevOps*, sugere antecipar as atividades de teste para fases iniciais do desenvolvimento. Isso está alinhado à visão de Aniche (2022), que defende a criação de testes desde os requisitos para orientar o design e garantir que o código atenda ao comportamento esperado.

Nas atividades práticas, foram exploradas ferramentas de mercado como *JUnit* (JUnit, 2025) para testes unitários e *Mockito* (Mockito, 2025) para a criação de *stubs* e *mocks*, simulando dependências externas e validando o comportamento isolado dos componentes. Conceitos como asserções, anotações, cobertura de testes e integração contínua (CI) foram aplicados para automatizar a execução dos testes em pipelines, fortalecendo a rastreabilidade e a confiabilidade das entregas.

O artefato final da disciplina consistiu na elaboração de um teste ponta a ponta (E2E) utilizando o *Playwright* (Microsoft, 2025), cujo objetivo foi automatizar a interação com o editor de texto online Anotepad (Anotepad, 2025). O teste simulou a criação de uma anotação com o título *“Entrega trabalho TEST DAS 2024”*, incluindo o nome e a matrícula no corpo do texto. Essa prática demonstrou a aplicação de *codegen* para geração automatizada de scripts e o uso de testes de interface para validação completa do comportamento da aplicação (Figura 29), consolidando o entendimento sobre automação e confiabilidade em ambientes reais.

A disciplina demonstrou que o teste é uma prática contínua e colaborativa, não restrita à fase final do ciclo de desenvolvimento. Sua integração com disciplinas como AAP, INFRA e MADS reforça a perspectiva de que qualidade e valor de negócio

são resultados diretos de práticas iterativas e incrementais de desenvolvimento e validação contínua. Assim, TEST consolidou a base técnica para um ciclo ágil sustentável, no qual a automação e a verificação permanente garantem entregas consistentes e de alto padrão.

12.1 ARTEFATOS DO PROJETO

FIGURA 32 - CÓDIGO DE TESTE

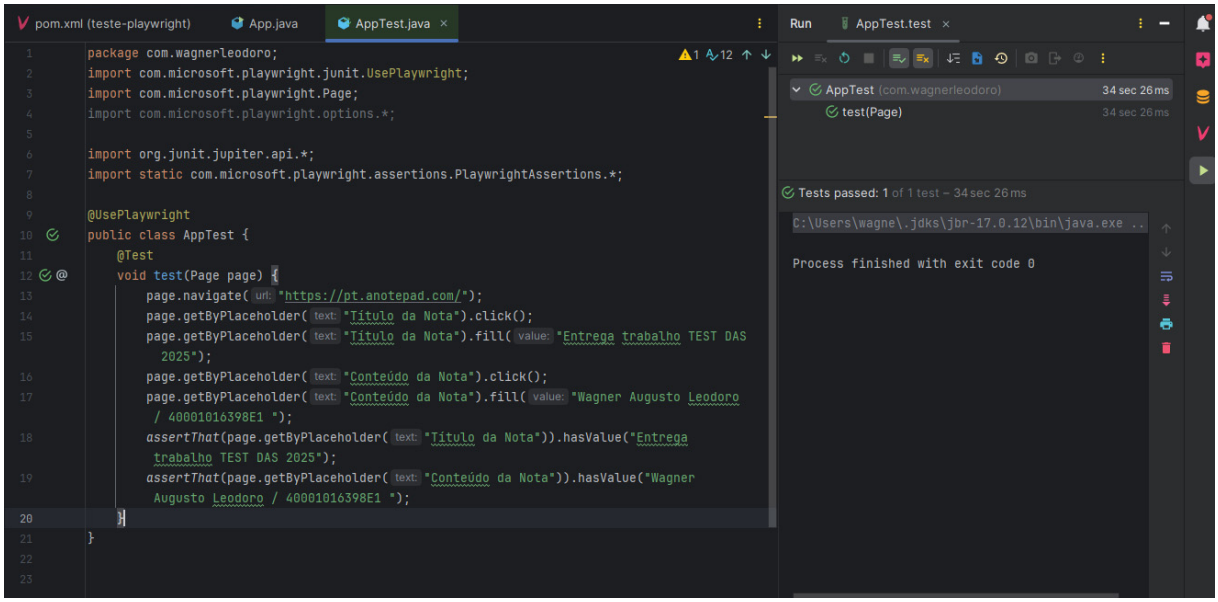
```
package com.wagnerleodoro;
import com.microsoft.playwright.junit.UsePlaywright;
import com.microsoft.playwright.Page;
import com.microsoft.playwright.options.*;

import org.junit.jupiter.api.*;
import static
com.microsoft.playwright.assertions.PlaywrightAssertions.*;

@UsePlaywright
public class AppTest {
    @Test
    void test(Page page) {
        page.navigate("https://pt.anotepad.com/");
        page.getByPlaceholder("Título da Nota").click();
        page.getByPlaceholder("Título da Nota").fill("Entrega
trabalho TEST DAS 2025");
        page.getByPlaceholder("Conteúdo da Nota").click();
        page.getByPlaceholder("Conteúdo da Nota").fill("Wagner
Augusto Leodoro / 40001016398E1 ");
        assertThat(page.getByPlaceholder("Título da
Nota")).hasValue("Entrega trabalho TEST DAS 2025");
        assertThat(page.getByPlaceholder("Conteúdo da
Nota")).hasValue("Wagner Augusto Leodoro / 40001016398E1 ");
    }
}
```

FONTE: O Autor (2025)

FIGURA 33 - RESULTADO DOS TESTES



FONTE: O Autor (2025)

13 CONCLUSÃO

O memorial apresentou a integração conceitual e prática das disciplinas do curso de Desenvolvimento Ágil de Software, evidenciando como a aplicação de métodos, frameworks e ferramentas se articula em um ciclo contínuo de planejamento, construção, validação e entrega de valor. Ao longo dos projetos, observou-se que o desenvolvimento ágil não se limita à adoção de técnicas como *Scrum*, *Kanban*, *TDD* ou *DevOps*, mas representa uma mudança de paradigma que envolve mentalidade, cultura e colaboração.

A consolidação dos conhecimentos adquiridos demonstrou que a agilidade requer disciplina, transparência e comunicação constante entre os integrantes da equipe. A ênfase na entrega incremental e na melhoria contínua mostrou-se essencial para garantir qualidade técnica, rastreabilidade e alinhamento com as necessidades do cliente. Entretanto, a implementação efetiva dessa abordagem enfrenta desafios relevantes. A flexibilidade característica das metodologias ágeis, embora seja uma vantagem, pode gerar confusão em equipes não adaptadas ou com papéis e responsabilidades pouco definidos.

Outro desafio identificado é a mudança de hábitos e de cultura organizacional, uma vez que a agilidade pressupõe autonomia, confiança e colaboração multidisciplinar. O ambiente de trabalho deve oferecer condições que estimulem a experimentação, o aprendizado e o feedback constante. Além disso, a natureza iterativa dos processos pode, em um primeiro momento, transmitir a impressão de lentidão ou repetição, quando na verdade representa o aprimoramento contínuo e o fortalecimento da qualidade do produto, especialmente no que se refere aos testes e à validação incremental.

Conclui-se que a adoção do desenvolvimento ágil exige mais do que o domínio de ferramentas: requer um compromisso coletivo com a entrega de valor, a melhoria dos processos e a aprendizagem contínua. A experiência relatada ao longo das disciplinas evidencia que a agilidade é tanto um conjunto de práticas quanto uma filosofia de trabalho, que, quando aplicada de forma coerente e integrada, resulta em softwares mais eficientes, sustentáveis e alinhados às reais necessidades dos usuários.

REFERÊNCIAS

ANDROID *Developers. Android Studio Documentation*. Disponível em: https://developer.android.com/. Acesso em: 12 set. 2025.

ANGULAR. *Angular: deliver web apps with confidence*. Google, 2025. Software. Disponível em: https://angular.io/. Acesso em: 12 set. 2025.

ANICHE, M. *Effective Software Testing: A Developer's Guide*. Shelter Island: Manning Publications, 2022.

ANOTEPAD. *Anotepad: Online Notepad App*. Disponível em: https://anotepad.com/. Acesso em: 12 set. 2025.

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

BECK, K. *TDD: desenvolvimento guiado por testes* [recurso eletrônico]. Tradução: CHEIRAN J. F. P. Revisão técnica: PIMENTA M. S. Porto Alegre: Bookman, 2010. Dados eletrônicos.

BOOTSTRAP. *Bootstrap: the most popular HTML, CSS, and JS library in the world*. The Bootstrap Authors, 2025. Software. Disponível em: https://getbootstrap.com/. Acesso em: 12 set. 2025.

CÂNDIDO, C. H. *brModelo: versão 3.32*. Florianópolis, abr. 2024. Software. Disponível em: https://www.sis4.com/brModelo/download.html. Acesso em: 12 set. 2025.

CHANGE Vision, Inc. *Astah UML*. 2025. Software. Disponível em: https://astah.net/products/astah-uml/. Acesso em: 12 set. 2025.

COHN, M. *Desenvolvimento de software com Scrum: aplicando métodos ágeis com sucesso*. Porto Alegre: Bookman, 2011.

DESIGN Sprint. *Design Sprint*. Disponível em: https://www.thesprintbook.com/the-design-sprint. Acesso em: 12 set. 2025.

DOCKER. *Docker Documentation*. Disponível em: https://www.docker.com/. Acesso em: 12 set. 2025.

EVANS, E. *Domain-driven design: tackling complexity in the heart of software*. Boston: Addison-Wesley, 2003.

FOWLER, M. *The Practical Test Pyramid*. 2012. Disponível em: https://martinfowler.com/articles/practical-test-pyramid.html. Acesso em: 12 set. 2025.

GITLAB. *GitLab Documentation*. Disponível em: https://about.gitlab.com/. Acesso em: 12 set. 2025.

HP-API. *HP-API: Harry Potter API*. Disponível em: https://hp-api.onrender.com/. Acesso em: 12 set. 2025.

HUMBLE, J.; FARLEY, D. *Entrega contínua: como entregar software de forma rápida e confiável*. Porto Alegre: Bookman, 2014.

INTERNATIONAL Organization for Standardization. ISO 9241-210:2010 – *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. Geneva: ISO, 2010.

JUNIT. *JUnit 5 User Guide*. Disponível em: https://junit.org/junit5/docs/current/user-guide/. Acesso em: 12 set. 2025.

KOTLIN Foundation. *Kotlin Language Documentation*. Disponível em: https://kotlinlang.org/. Acesso em: 12 set. 2025.

KUBECTL. *kubectl Command Reference*. Disponível em: https://kubernetes.io/docs/reference/kubectl/. Acesso em: 12 set. 2025.

KUBERNETES. *Kubernetes Documentation*. Disponível em: https://kubernetes.io/. Acesso em: 12 set.

MARTIN, R. C. *Arquitetura limpa: o guia do artesão para estrutura e design de software*. Tradução: NASCIMENTO, R. Rio de Janeiro: Alta Books, 2019.

MARTIN, R. C. *Código limpo: Habilidades práticas do Agile Software*. Rio de Janeiro: Editora Alta Books, 2009. *E-book.* p.4. ISBN 9788550816043. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788550816043/. Acesso em: 28 set. 2025.

MATERIAL Design. *Material Design*. Google, 2025. Framework. Disponível em: https://m3.material.io/. Acesso em: 12 set. 2025.

MICROSOFT Corporation. *Microsoft Planner*. 2025. Software. Disponível em: https://www.microsoft.com/pt-br/microsoft-365/planner/microsoft-planner. Acesso em: 12 set. 2025.

MICROSOFT. *Playwright: Fast and Reliable End-to-End Testing for Modern Web Apps*. Disponível em: https://playwright.dev/. Acesso em: 12 set. 2025.

MOCKITO. *Mockito Framework Documentation*. Disponível em: https://site.mockito.org/. Acesso em: 12 set. 2025.

POSTGRESQL**.** *PostgreSQL: the world's most advanced open source relational database*. PostgreSQL Global Development Group, 2025. Software. Disponível em: https://www.postgresql.org/. Acesso em: 12 set. 2025.

PRESSMAN, R. S. *Engenharia de Software: uma abordagem profissional*. 7. ed. São Paulo: McGraw-Hill, 2010.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. (org.). *Métodos Ágeis para Desenvolvimento de Software*. Porto Alegre: Bookman, 2014.

SCHANK, R. C. *What We Learn When We Learn by Doing*. Technical Report No. 60, Northwestern University, 1995. Disponível em: https://cogprints.org/637/. Acesso em: 12 set. 2025.

SMART, J. F. *BDD in action: behavior-driven development for the whole software lifecycle*. Shelter Island, NY: Manning Publications, 2015.

SOFTWARE Development Kanban Game. *Software Development Kanban Game*. Disponível em: http://www.kanbanboardgame.com/. Acesso em: 12 set. 2025.

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

SPRING**.** *Spring Boot*. Pivotal Software; VMware, 2025. Framework. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 12 set. 2025.

TYPESCRIPT. *TypeScript: JavaScript with syntax for types*. Microsoft, 2025. Software. Disponível em: https://www.typescriptlang.org/. Acesso em: 12 set. 2025.