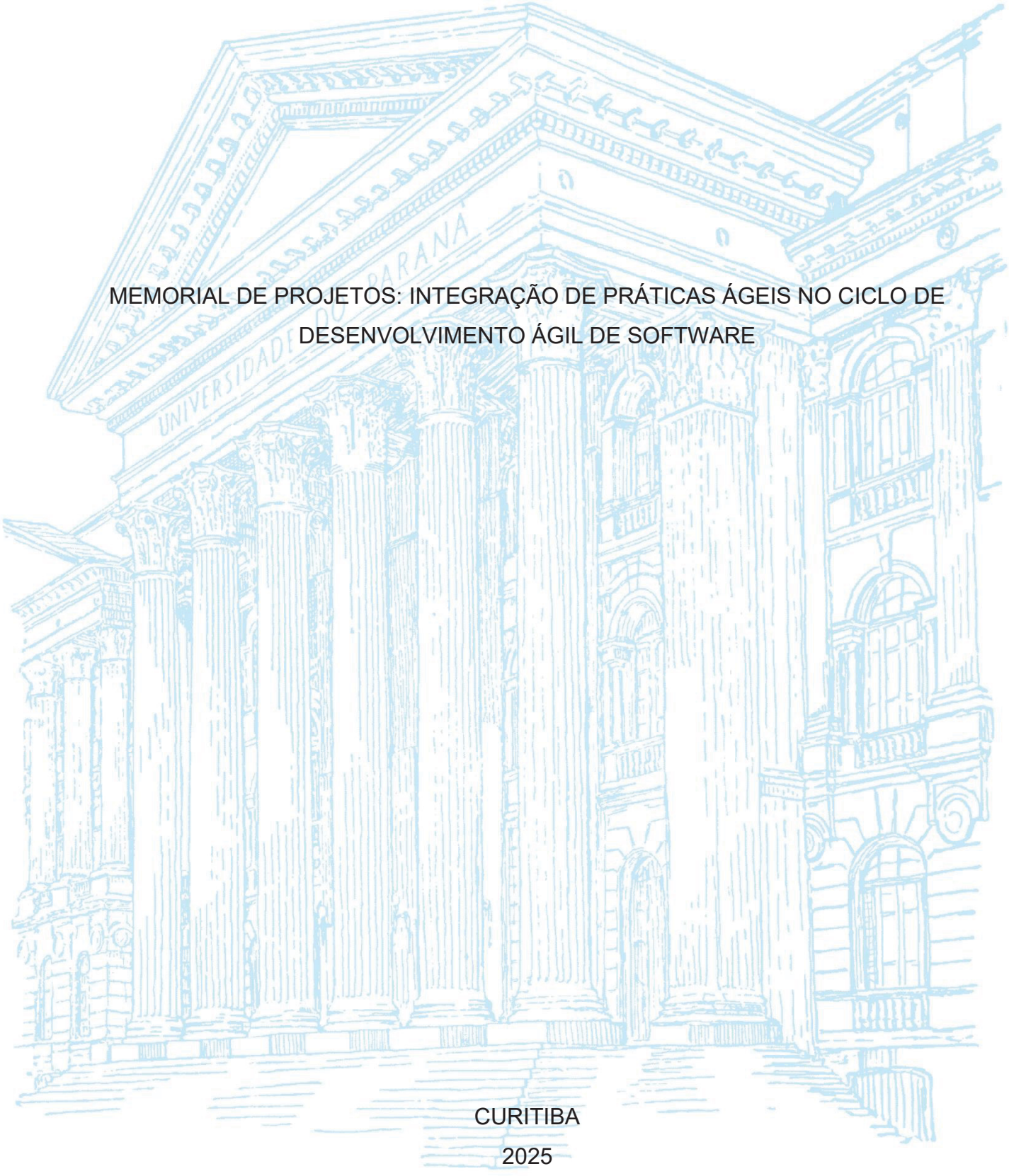
UNIVERSIDADE FEDERAL DO PARANÁ

LUIS FELIPE ORTEGA LYNG



MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO CICLO DE DESENVOLVIMENTO ÁGIL DE SOFTWARE

CURITIBA

2025

LUIS FELIPE ORTEGA LYNG

MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO CICLO DE DESENVOLVIMENTO ÁGIL DE SOFTWARE

Memorial de Projetos apresentado ao curso de Especialização 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

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 **LUIS FELIPE ORTEGA LYNG**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO CICLO DE 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, 04 de Dezembro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial de projetos consolida os trabalhos desenvolvidos durante a Especialização em Desenvolvimento Ágil de Software da Universidade Federal do Paraná, apresentando um portfólio integrado das quinze disciplinas cursadas ao longo de 2024. O objetivo principal é demonstrar a aplicação prática de metodologias ágeis no ciclo completo de desenvolvimento de *software*, desde a concepção até a entrega. O trabalho está estruturado em cinco eixos temáticos: Fundamentos Ágeis (metodologias e gerenciamento de projetos), Modelagem e Arquitetura (visões funcional e estrutural, além de UX), Fundamentos de Programação (introdução à programação, banco de dados e aspectos ágeis), Desenvolvimento de Aplicações (*web* e *mobile*) e Qualidade e Infraestrutura (*DevOps* e testes automatizados). Os projetos apresentados demonstram a evolução do aprendizado por meio de artefatos práticos que integram conceitos de *Scrum*, *Kanban*, histórias de usuário, diagramas UML, desenvolvimento *full stack* e práticas de CI/CD. A análise técnica realizada evidencia a importância da integração entre as diferentes disciplinas para a construção de soluções de software robustas e alinhadas aos princípios ágeis. Como principais desafios identificados, destacam-se a sincronização entre as atividades de UX e desenvolvimento, a implementação de testes automatizados em ambientes de entrega contínua e a adaptação de práticas ágeis em contextos com restrições organizacionais. Este memorial contribui para a consolidação do conhecimento sobre desenvolvimento ágil de *software*, oferecendo uma visão integrada e prática das competências adquiridas durante a especialização.

Palavras-chave: desenvolvimento ágil de software; metodologias ágeis; desenvolvimento web e mobile; DevOps; testes automatizados.

ABSTRACT

This project memorial consolidates the work developed during the Specialization in Agile Software Development at the Federal University of Paraná, presenting an integrated portfolio of the fifteen courses completed throughout 2024. The main objective is to demonstrate the practical application of agile methodologies in the complete software development cycle, from conception to delivery. The work is structured into five thematic axes: Agile Fundamentals (methodologies and project management), Modeling and Architecture (functional and structural views, as well as UX), Programming Fundamentals (introduction to programming, databases, and agile aspects), Application Development (web and mobile), and Quality and Infrastructure (DevOps and automated testing). The projects presented demonstrate the evolution of learning through practical artifacts that integrate concepts of Scrum, Kanban, user stories, UML diagrams, full stack development, and CI/CD practices. The technical analysis conducted highlights the importance of integration among different disciplines for building robust software solutions aligned with agile principles. The main challenges identified include synchronization between UX activities and development, implementation of automated tests in continuous delivery environments, and adaptation of agile practices in contexts with organizational constraints. This memorial contributes to the consolidation of knowledge about agile software development, offering an integrated and practical view of the competencies acquired during the specialization.

Keywords: agile software development; agile methodologies; web and mobile development; DevOps; automated testing.

SUMÁRIO

1 PARECER TÉCNICO

O presente parecer técnico sintetiza, de forma integrada, como as quinze disciplinas da Especialização em Desenvolvimento Ágil de Software da Universidade Federal do Paraná contribuíram para formar uma visão sistêmica do ciclo de vida de *software*. De acordo com Beck *et al.* (2001), o desenvolvimento ágil prioriza colaboração, adaptação e entrega contínua de valor, substituindo planos rígidos por ciclos de aprendizado. Bourque e Fairley (2014) ressaltam que a engenharia de *software* envolve áreas que devem atuar de maneira coordenada, da elicitação de requisitos à manutenção, articulação que esteve no centro das atividades práticas e reflexões propostas ao longo do curso.

Na disciplina de Métodos Ágeis de Desenvolvimento de Software (MADS), foram estudados *Scrum*, *Kanban* e *Extreme Programming* (XP) como estruturas para organizar o trabalho em ciclos curtos. Beck *et al.* (2001) destacam que iterações frequentes, *backlog* priorizado e retrospectivas fortalecem a transparência e a capacidade de adaptação da equipe. Cohn (2011) reforça o papel das histórias de usuário na comunicação entre clientes e desenvolvedores, favorecendo o entendimento de valor de negócio e a estimativa de esforço. Os princípios de fluxo contínuo apresentados por Anderson (2011) apoiaram a limitação de trabalho em progresso e a identificação de gargalos, prática experimentada nos quadros de tarefas e nas reuniões de acompanhamento dos projetos.

Nos módulos de Gerenciamento Ágil de Projetos I e II (GAP1 e GAP2), esses conceitos foram combinados com técnicas de planejamento, negociação e monitoramento. O Project Management Institute - PMI (2021) e Vargas (2017) destacam que escopo, prazos e riscos precisam ser revistos de forma sistemática para manter alinhamento com os objetivos do negócio, inclusive em ambientes ágeis. Kupiainen, Mäntylä e Itkonen (2015) indicam que métricas simples, como *lead time* e taxa de entrega, apoiam decisões rápidas e baseadas em dados. Com esse embasamento, foram construídos cronogramas de *release*, mapas de *stakeholders* e indicadores de fluxo que tornaram visível o avanço dos produtos e facilitaram a priorização colaborativa das entregas.

As disciplinas de Modelagem Ágil de Software I e II (MAG1 e MAG2) abordaram o uso de modelos visuais como apoio à comunicação e à análise de

alternativas de solução. Sommerville (2019) afirma que diagramas de casos de uso, de classes e de sequência reduzem ambiguidades e auxiliam na validação de requisitos com usuários e especialistas de domínio. Poppendieck e Poppendieck (2003) defendem que a documentação deve ser enxuta e produzida apenas quando agrega valor real ao processo, o que levou à produção de artefatos focados em decisões-chave. Em consonância, Girvan e Paul (2021) tratam a modelagem como linguagem compartilhada para discutir arquitetura; na especialização, os modelos foram elaborados de maneira incremental e utilizados como base para discussão técnica, em vez de documentação meramente burocrática.

A disciplina de Introdução à Programação (INTRO) consolidou bases de lógica, estruturas de controle e algoritmos, conforme a abordagem didática apresentada por Deitel e Deitel (2016). Aspectos Ágeis de Programação (AAP) retomou esse conteúdo com foco em práticas profissionais de *clean code*, testes automatizados e refatoração. Martin (2017) defende que código simples, coeso e com baixo acoplamento reduz riscos de defeitos e facilita a evolução contínua. Fowler (2020) descreve a refatoração como instrumento de melhoria incremental, desde que apoiado por uma boa suíte de testes. Pressman e Maxim (2021) apontam que a qualidade interna de um sistema resulta da combinação entre boas decisões arquiteturais e disciplina de programação no dia a dia.

As disciplinas técnicas voltadas a *back-end*, *front-end* e banco de dados aproximaram os alunos de ferramentas amplamente utilizadas no mercado e mostraram como integrar diferentes camadas de um sistema. Jandl Junior (2021) destaca que padrões de projeto e princípios de orientação a objetos favorecem extensibilidade e reuso de componentes. Silberschatz, Korth e Sudarshan (2020) e Elmasri e Navathe (2019) apresentam fundamentos para modelagem de dados, normalização e uso adequado de *SQL*, essenciais para garantir integridade e desempenho. A articulação entre arquitetura, interface e persistência permitiu estruturar soluções completas, nas quais decisões de banco de dados, camadas de serviço e interface gráfica foram avaliadas em conjunto com as necessidades de negócio.

A disciplina de *UX* no Desenvolvimento Ágil de Software apresentou métodos centrados no usuário e aproximou o desenvolvimento de atividades de pesquisa e prototipação. Sharp, Preece e Rogers (2019) defendem que compreender objetivos, tarefas e contexto de uso é condição para criar interfaces eficazes. Norman (2024)

enfatiza que uma boa experiência resulta da combinação de utilidade, clareza e respostas emocionais positivas. Lowdermilk (2013) e Gothelf e Seiden (2021) apontam que a validação contínua por protótipos e testes em ciclos curtos é indispensável em ambientes ágeis. Com esse suporte teórico, foram conduzidas entrevistas exploratórias, fluxos de navegação e protótipos de baixa fidelidade, incorporando o *feedback* coletado diretamente ao *backlog* dos projetos.

Infraestrutura para Desenvolvimento e Implantação de Software (INFRA) e Testes Automatizados (TEST) completaram a formação com práticas voltadas à qualidade e confiabilidade. Humble e Farley (2013) apresentam princípios de *DevOps* e integração contínua (*CI/CD*) como essenciais para reduzir riscos de implantação e permitir pequenas mudanças frequentes com segurança. Kim *et al.* (2021) reforçam a importância da automação e do monitoramento para manter estabilidade em ambientes complexos. Beck (2002) introduz o *Test-Driven Development* (*TDD*) como forma de projetar software a partir de testes, enquanto North (2006) apresenta o *Behavior-Driven Development* (*BDD*), aproximando linguagem técnica e de negócio. Esses conceitos sustentaram a construção de *pipelines* de automação e conjuntos de testes que acompanharam as entregas incrementais desenvolvidas na especialização.

A especialização também abordou métricas, pontos de função e contratos ágeis, relacionando aspectos técnicos e organizacionais. Vazquez, Simões e Albert (2013) destacam a utilidade da contagem por pontos de função em estimativas e no acompanhamento de escopo em projetos de software. Zijdemans e Stettina (2014) analisam contratos que procuram equilibrar flexibilidade e previsibilidade, tema sensível em organizações que buscam adotar métodos ágeis. Amaral *et al.* (2013) e Camargo e Ribas (2019) discutem liderança servidora e aprendizagem organizacional como fatores para sustentar mudanças de cultura. Em síntese, as disciplinas convergiram para uma visão prática e integrada da engenharia de software. Pressman e Maxim (2021) e Sommerville (2019) afirmam que a maturidade depende da combinação entre processos adequados e competências técnicas; apoiado nas contribuições de Beck *et al.* (2001), Kim *et al.* (2021) e Gothelf e Seiden (2021), o egresso passa a dispor de conhecimento consistente para atuar em equipes multidisciplinares e orientar decisões com base em agilidade, qualidade e geração de valor.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

A disciplina de Métodos Ágeis para Desenvolvimento de Software apresentou os fundamentos essenciais que sustentam a filosofia ágil na engenharia de software. Foram estudados os valores e princípios do Manifesto Ágil de Beck *et al.* (2001), enfatizando a colaboração entre pessoas, a entrega contínua de valor e a adaptação a mudanças. O conteúdo explorou os *frameworks Scrum*, *Kanban* e *Extreme Programming* (XP), destacando práticas de iteração curta, priorização de *backlog* e *feedback* constante, em linha com as orientações de Cohn (2011) para aplicação do *Scrum* em projetos de *software*.

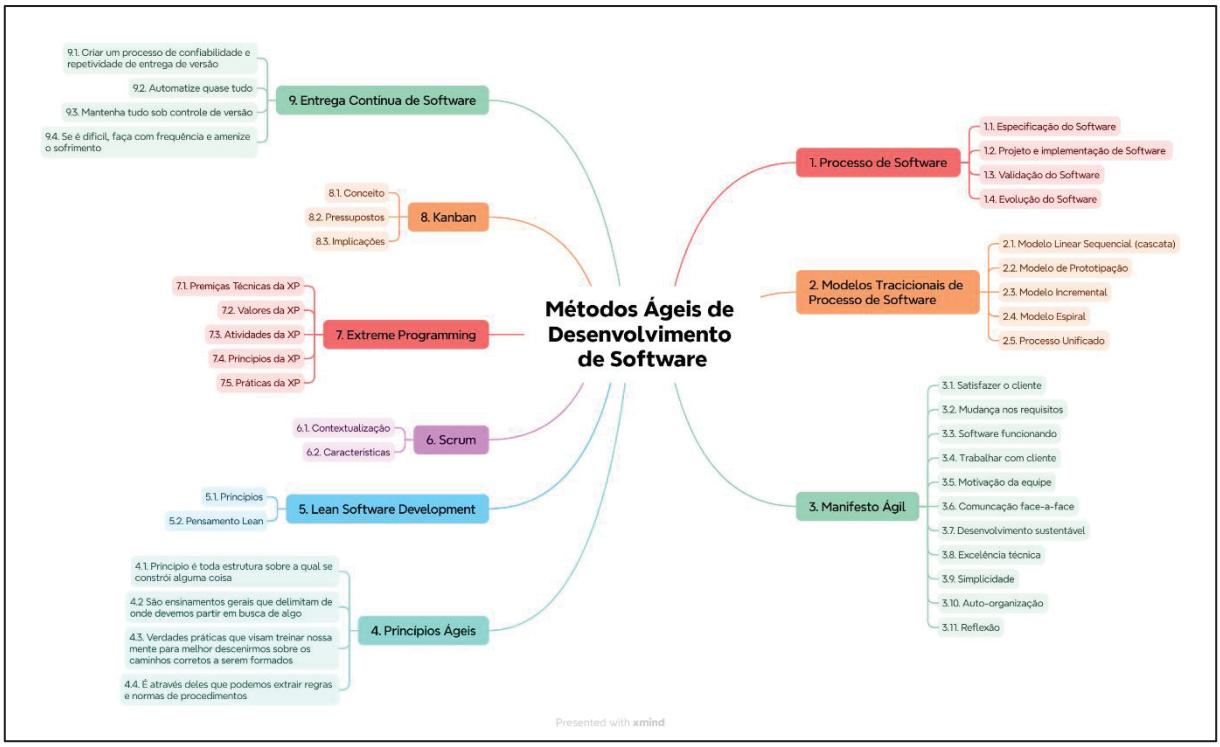
A análise dos papéis, artefatos e cerimônias do *Scrum* permitiu compreender a importância do planejamento incremental e da transparência no trabalho em equipe. Aplicou-se *Kanban* para o controle de fluxo e identificação de gargalos, conforme Anderson (2011), enquanto o XP, inspirado em Beck (2004), reforçou a integração entre codificação, refatoração e testes automatizados. Também foram introduzidos princípios de integração e entrega contínua (CI/CD), com base em Humble e Farley (2013), evidenciando o papel da automação e da colaboração entre desenvolvimento e operações.

A disciplina serviu de base para as demais unidades do curso, sendo constantemente referenciada em projetos de gerenciamento, modelagem, testes e infraestrutura. Os conceitos de iteração e melhoria contínua apoiaram a organização das atividades de Gerenciamento Ágil de Projetos (GAP1 e GAP2), enquanto a noção de adaptação e inspeção influenciou a Modelagem Ágil (MAG1 e MAG2) e o desenvolvimento *Web* e *Mobile*. Além disso, o foco no valor entregue ao usuário aproximou a disciplina de UX e da disciplina de Infraestrutura para Desenvolvimento e Implantação de *Software* (INFRA), reforçando a integração entre planejamento, design e implementação.

Como produto, elaborou-se um mapa mental no aplicativo *X-Mind*, que sintetiza de forma visual os principais tópicos abordados, o Manifesto Ágil, *frameworks* e práticas integradas, representando a base conceitual que orienta o desenvolvimento ágil de *software* e a aplicação prática dos princípios estudados ao longo da especialização.

2.1 ARTEFATOS DO PROJETO

FIGURA 1 - MAPA MENTAL DA DISCIPLINA MADS



FONTE: O Autor, 2025.

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

As disciplinas de Modelagem Ágil de Software I e II tiveram como objetivo desenvolver competências voltadas à concepção e representação de sistemas de *software* de forma integrada e iterativa, em consonância com os princípios do Manifesto Ágil de Beck *et al.* (2001).

A primeira etapa (MAG1) concentrou-se na modelagem funcional, abordando o levantamento de requisitos, a definição de casos de uso, histórias de usuário e a prototipação. O projeto aplicado consistiu na modelagem de um sistema para gestão condominial, no qual foram elaborados diagramas de casos de uso de níveis 1 e 2, identificando atores, funcionalidades e as principais interações do sistema, conforme a abordagem de especificação proposta por Sommerville (2019). A construção de histórias de usuário e critérios de aceitação proporcionou clareza nas funcionalidades e alinhamento entre cliente e equipe, sustentando a criação do *backlog* e as práticas de planejamento incremental, em linha com as recomendações de Cohn (2011).

Na segunda etapa (MAG2), a modelagem estrutural aprofundou os conceitos técnicos, utilizando diagramas de classes e de sequência para representar entidades, relacionamentos e fluxos de operação. As estruturas criadas, baseadas em UML (*Unified Modeling Language*), consolidaram o entendimento de herança, encapsulamento e dependência entre objetos, garantindo a coerência entre análise e codificação, como discutido por Pressman e Maxim (2021).

A integração entre as etapas funcional e estrutural fortaleceu o vínculo entre requisitos e implementação, servindo de base para disciplinas subsequentes, como Banco de Dados (BD), Aspectos Ágeis de Programação (AAP) e Testes Automatizados (TEST). Para Sommerville (2019), a modelagem é essencial para reduzir ambiguidades e promover uma visão compartilhada do sistema, permitindo que o desenvolvimento ocorra de forma incremental e continuamente validada. Poppendieck e Poppendieck (2003) ressaltam, ainda, que os modelos devem permanecer enxutos e focados em decisões relevantes, evitando documentação excessiva. Em complemento, Girvan e Paul (2021) tratam a modelagem como uma linguagem comum entre negócio e tecnologia, reforçando seu papel colaborativo no contexto ágil.

Essas disciplinas demonstraram, assim, a importância da modelagem como ferramenta de apoio ao pensamento ágil, ao fornecer artefatos leves, comunicativos e evolutivos. A síntese obtida entre análise, *design* e *feedback* contínuo resultou em um conjunto de modelos funcionais e estruturais que orientaram a implementação do projeto de forma integrada, eficiente e alinhada aos valores do desenvolvimento ágil de *software*.

3.1 ARTEFATOS DO PROJETO

FIGURA 2 - REQUISITOS DO SISTEMA DE GESTÃO DE CONDOMÍNIO

Sistema de Gestão de Condomínio

Um condomínio deseja um sistema para automatizar seus serviços. As solicitações iniciais do Síndico são as seguintes:

a) Cadastrar Apartamento: Número do Apto, Bloco;

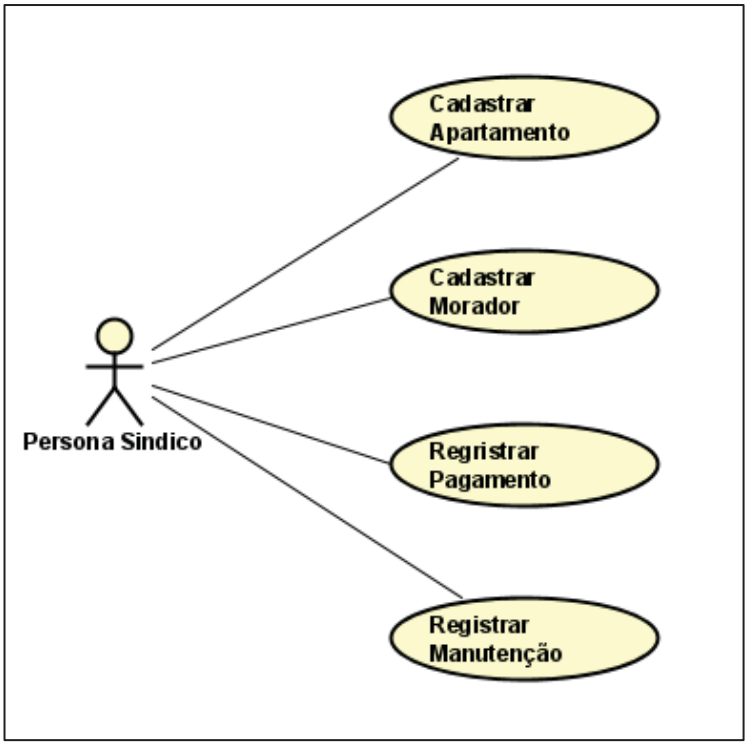
b) Cadastrar Moradores: CPF, Nome, Telefone, Apartamento, Responsável (S/N), Proprietário(S/N); - Ao incluir um novo morador, verificar se já existe o CPF. - Consistir se o número do apartamento já existe no cadastro. Guardar os dados dos veículos dos moradores, bem como suas vagas de garagem

c) Registrar o Pagamento do Condomínio: Apartamento, Mês/Ano Referência, Data Vencimento, Data Pagamento, Valor Condomínio; - Ao digitar o número do apartamento, o sistema deve apresentar o nome do morador responsável e o valor do condomínio. - Um pagamento só pode ser registrado se não houver valores anteriores vencidos.

d) Registrar as manutenções no prédio: As manutenções podem ser de pintura, limpeza de caixa d'agua, jardinagem etc.;

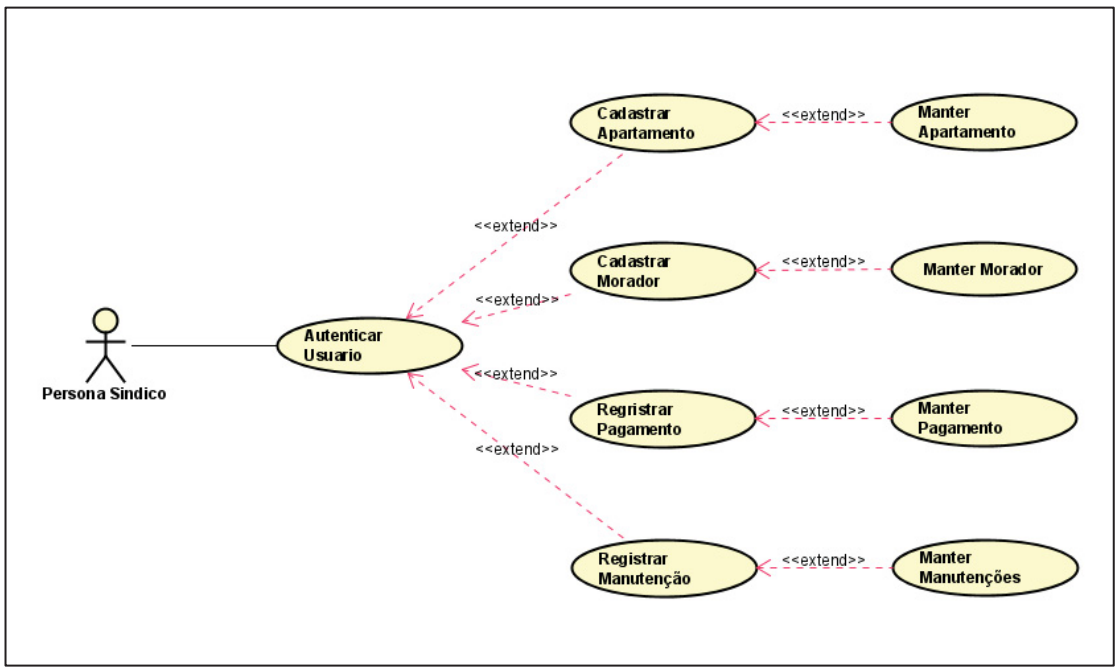
FONTE: O Autor, 2025.

FIGURA 3 – DIAGRAMA DE CASO DE USO NÍVEL 1



FONTE: O Autor, 2025.

FIGURA 4 – DIAGRAMA DE CASO DE USO NÍVEL 2



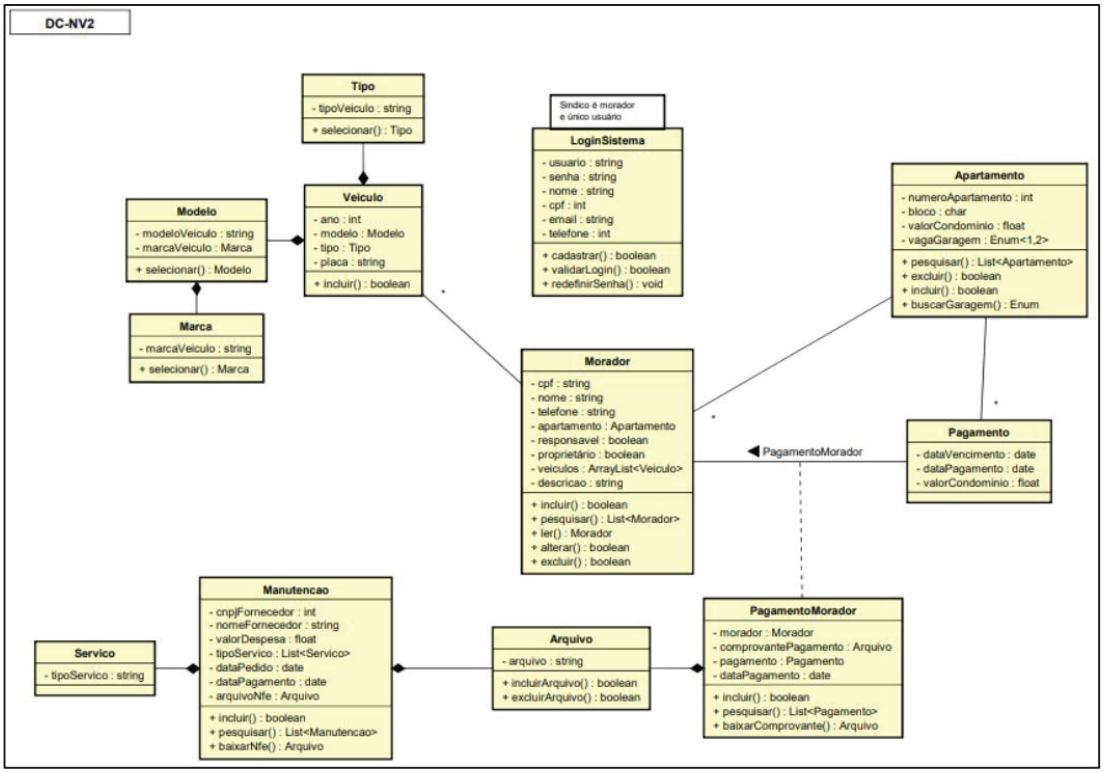
FONTE: O Autor, 2025.

FIGURA 5 – HISTÓRIA DE USUÁRIO

HU001 - Logar no sistema	SENDO	um síndico
	QUERO	me logar no sistema
	PARA	ter acesso às informações do condomínio
HU002 - Pesquisar Apartamento	SENDO	um síndico
	QUERO	pesquisar um novo apartamento
	PARA	fazer manutenções nos seus dados
HU003 - Manter Apartamento	SENDO	um síndico
	QUERO	manter os dados de um apartamento
	PARA	que seus dados fiquem atualizados
HU004 - Pesquisar Morador	SENDO	um síndico
	QUERO	pesquisar um novo morador
	PARA	fazer manutenções nos seus dados
HU005 - Manter Morador	SENDO	um síndico
	QUERO	manter os dados de um morador
	PARA	que seus dados fiquem atualizados
HU006 - Pesquisar Pagamento	SENDO	um síndico
	QUERO	pesquisar um pagamento de um morador
	PARA	fazer manutenções nos seus dados
HU007 - Registrar Pagamento	SENDO	um síndico
	QUERO	incluir um pagamento de condomínio
	PARA	manter os dados registrados no sistema
HU008 - Pesquisar Manutenção	SENDO	um síndico
	QUERO	pesquisar uma manutenção no condomínio
	PARA	fazer manutenções nos seus dados
HU009 - Registrar Manutenção	SENDO	um síndico
	QUERO	incluir uma manutenção no condomínio
	PARA	manter os dados registrados no sistema

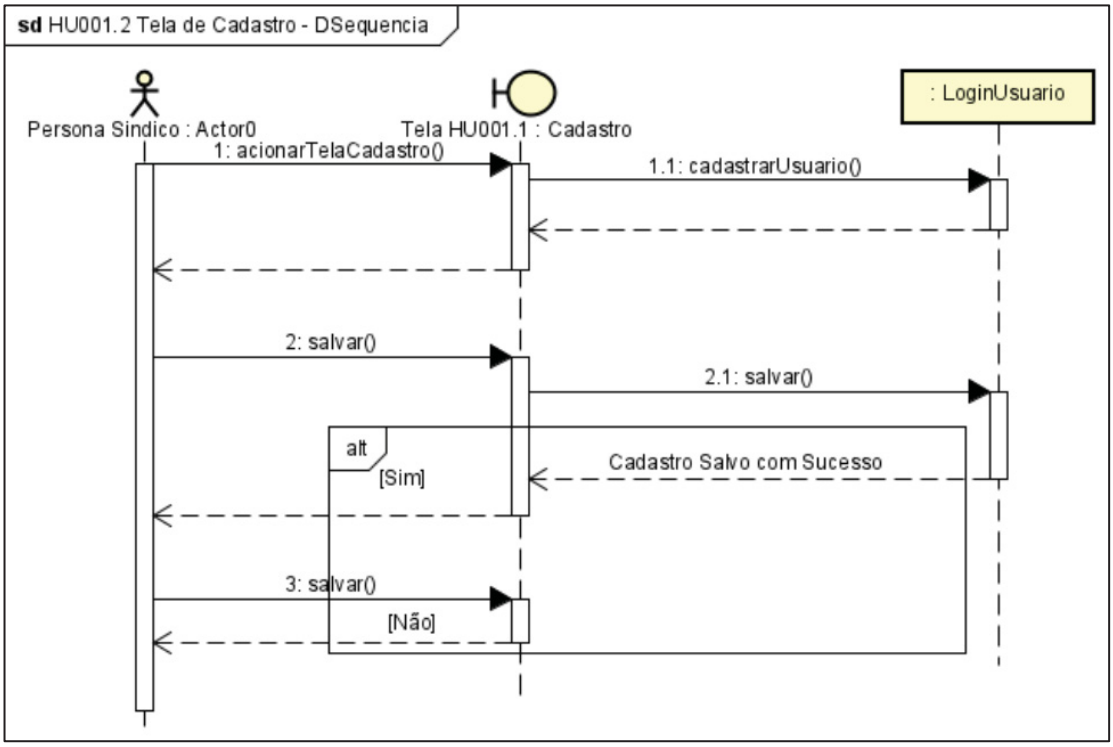
FONTE: O Autor, 2025.

FIGURA 6 - DIAGRAMA DE CASO DE USO NÍVEL 2



FONTE: O Autor, 2025.

FIGURA 7 – DIAGRAMA DE SEQUÊNCIA – TELA DE CADASTRO



FONTE: O Autor, 2025.

FIGURA 8 – CRITÉRIOS DE ACEITAÇÃO NA TELA CADASTRO

1.2.4 Critérios de aceitação – Detalhamento

1) Deve permitir cadastrar.

Dado que	estou na tela **HU001.3** e estou realizando o cadastro
Quando	acessou a tela de cadastro
Então	o usuário é cadastrado e direcionado para tela de login

2) Não deve permitir cadastrar.

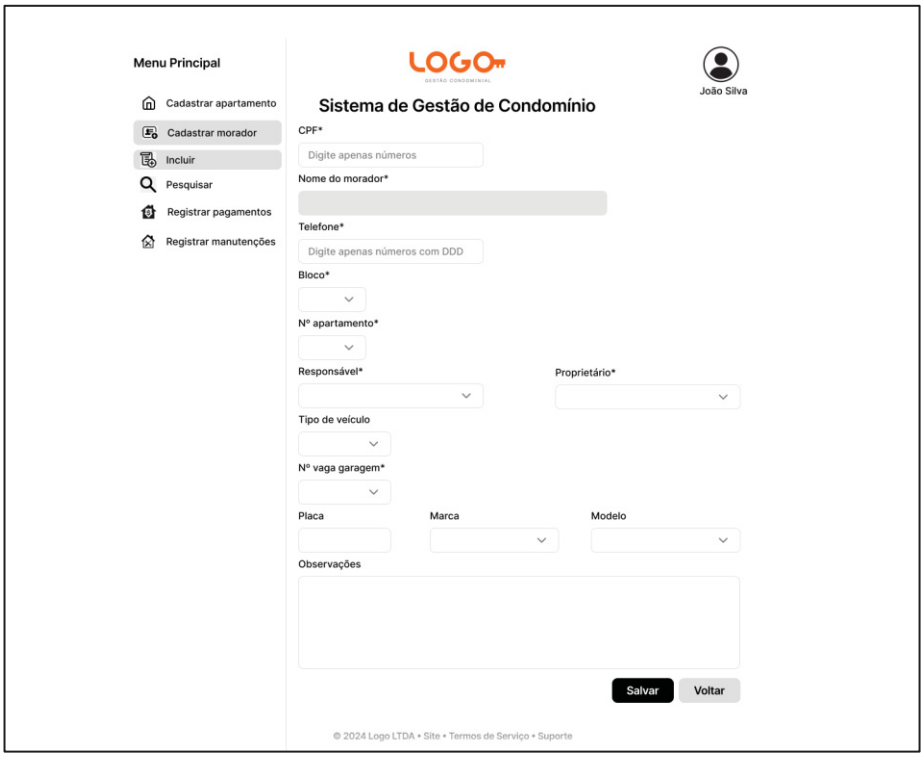
Dado que	estou na tela **HU001.3** realizando o cadastro
Quando	o usuário informou um CPF já existente
Então	apresenta uma mensagem: “CPF informado já está cadastrado”

3) Não deve permitir o cadastro quando os campos com (*) não forem preenchidos.

Dado que	estou na tela **HU001.3** realizando o cadastro
Quando	o usuário não preencheu os campos com (*)
Então	apresenta uma mensagem: ‘’preencher todos os campos com (*)”

FONTE: O Autor, 2025.

FIGURA 9 – TELA DE CADASTRO DE MORADOR



FONTE: O Autor, 2025.

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

As disciplinas de Gerenciamento Ágil de Projetos de Software I e II abordaram os fundamentos e práticas de planejamento, execução e controle de projetos de software sob a ótica dos métodos ágeis. Na primeira etapa (GAP1), o foco esteve na elaboração de um plano de *release* para o sistema de gestão condominial, consolidando a aplicação de ferramentas como o *Product Backlog* e o *Sprint Planning*. Foram definidas e priorizadas histórias de usuário considerando os papéis de síndico, morador e administrador, com estimativas baseadas na velocidade média da equipe e capacidade de entrega. Essa atividade possibilitou compreender a importância da transparência e da previsibilidade na gestão de projetos, alinhando-se às recomendações de Cohn (2011) e do PMI (2021) quanto à priorização de valor, gestão de riscos e entregas incrementais.

Durante o desenvolvimento, foram aplicadas práticas como *Planning Poker*, *backlog grooming* (refinamento do backlog) e definição de critérios de aceite, que viabilizaram a criação de um cronograma enxuto e flexível, promovendo entregas contínuas e *feedbacks* frequentes. A disciplina enfatizou o papel do gerente ágil como facilitador do processo e apoiador do time, reforçando o valor da comunicação efetiva e da colaboração multidisciplinar, em consonância com o papel de facilitador descrito para o *Scrum Master* por Schwaber e Sutherland (2020).

Na segunda etapa (GAP2), a ênfase voltou-se à melhoria contínua e à otimização de fluxo, por meio de simulações práticas, como a dinâmica *Kanban*, que evidenciaram a relevância da limitação do trabalho em progresso (*Work in Progress* – WIP), da identificação de gargalos e da importância da inspeção constante. Conforme Anderson (2011), o *Kanban* permite visualizar o processo e adaptar o fluxo de trabalho de acordo com a capacidade da equipe, o que contribui para maior eficiência e qualidade nas entregas.

A integração das disciplinas GAP1 e GAP2 com Modelagem Ágil (MAG1 e MAG2), Desenvolvimento *Web* e *Mobile*, Testes e Infraestrutura (*DevOps*) foi essencial para garantir a coerência entre planejamento, execução e validação das entregas. O gerenciamento estruturado permitiu que o curso reproduzisse, em escala acadêmica, o ciclo real de desenvolvimento ágil de forma iterativa, incremental e

colaborativa, assegurando a entrega contínua de valor ao usuário final e a evolução sustentável do projeto, em consonância com os valores do Manifesto Ágil de Beck *et al.* (2001).

4.1 ARTEFATOS DO PROJETO

FIGURA 10 - ESTIMATIVA DE PONTOS POR HISTÓRIA DE USUÁRIO

Nome	Descrição Resumida		Estimativa em Pontos (Dias Ideais)
HU001 - Logar no sistema	SENDO QUERO PARA	um síndico me logar no sistema ter acesso às informações do condomínio	5
HU002 - Pesquisar Apartamento	SENDO QUERO PARA	um síndico pesquisar um novo apartamento fazer manutenções nos seus dados	5
HU003 - Manter Apartamento	SENDO QUERO PARA	um síndico manter os dados de um apartamento que seus dados fiquem atualizados	5
HU004 - Pesquisar Morador	SENDO QUERO PARA	um síndico pesquisar um novo morador fazer manutenções nos seus dados	5
HU005 - Manter Morador	SENDO QUERO PARA	um síndico manter os dados de um morador que seus dados fiquem atualizados	10
HU006 - Pesquisar Pagamento	SENDO QUERO PARA	um síndico pesquisar um pagamento de um morador fazer manutenções nos seus dados	5
HU007 - Registrar Pagamento	SENDO QUERO PARA	um síndico incluir um pagamento de condomínio manter os dados registrados no sistema	5
HU008 - Pesquisar Manutenção	SENDO QUERO PARA	um síndico pesquisar uma manutenção no condomínio fazer manutenções nos seus dados	5
HU009 - Registrar Manutenção	SENDO QUERO PARA	um síndico incluir uma manutenção no condomínio manter os dados registrados no sistema	5
		TOTAL PONTOS PROJETO	50

FONTE: O Autor, 2025.

FIGURA 11 – CÁLCULO DE VELOCIDADE E PLANO DE RELEASE

Cálculo da Velocidade:

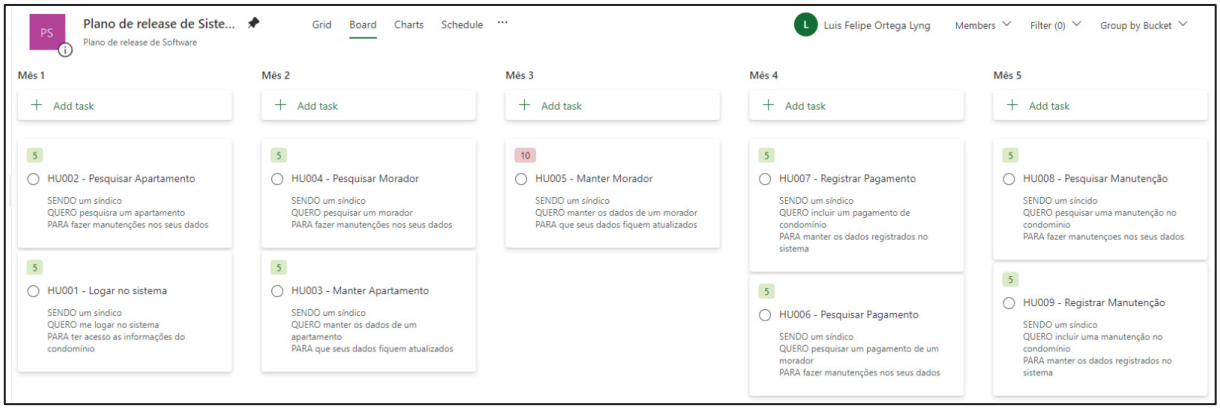
Horas disponíveis por dia: 4 horas	Tamanho da Sprint: 50 pontos	Duração total: 5 meses
Horas disponíveis por Sprint: 80 horas	Velocidade: 10	

Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4	Iteração/Sprint 5
Data Início: 29/04/24	Data Início: 28/05/24	Data Início: 02/07/24	Data Início: 30/07/24	Data Início: 27/08/24
Data Fim: 27/05/24	Data Fim: 01/07/24	Data Fim: 29/07/24	Data Fim: 26/08/24	Data Fim: 23/09/24
<HU001 – Logar no Sistema> ESTIMATIVA (5)	<HU003 – Manter Apartamento > ESTIMATIVA (5)	<HU005 – Manter Morador> ESTIMATIVA (10)	<HU006 – Pesquisar Pagamento> ESTIMATIVA (5)	<HU008 – Pesquisar Manutenção> ESTIMATIVA (5)
<HU002 – Pesquisar Apartamento> ESTIMATIVA (5)	<HU004 – Pesquisar Morador > ESTIMATIVA (5)		<HU007 – Registrar Pagamento> ESTIMATIVA (5)	<HU009 – Registrar Manutenção> ESTIMATIVA (5)

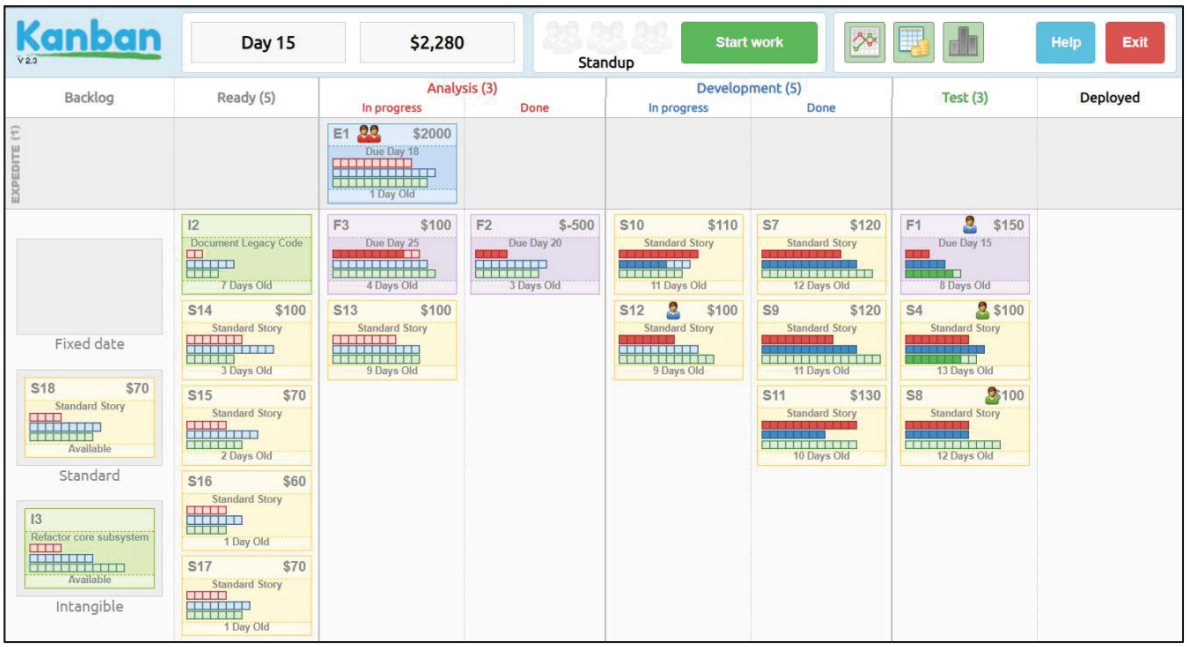
FONTE: O Autor, 2025.

FIGURA 12 – PLANO RELEASE ELABORADO NO MICROSOFT PLANNER



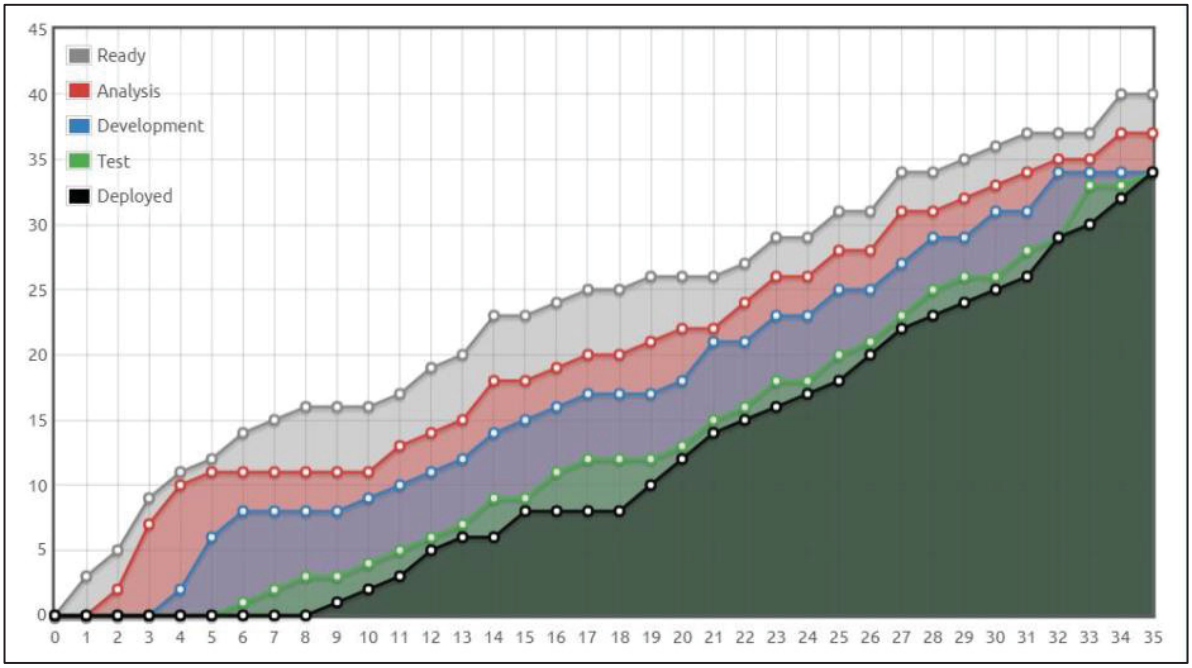
FONTE: O Autor, 2025.

FIGURA 13 – TELA DO KANBAN CARD GAME



FONTE: O Autor, 2025.

FIGURA 14 – CUMULATIVE FLOW DIAGRAM



FONTE: O Autor, 2025.

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação apresentou os conceitos fundamentais da lógica computacional e das estruturas básicas de programação, fornecendo a base necessária para a formação de profissionais na área de desenvolvimento de *software*. Utilizando a linguagem Java e a IDE NetBeans, os estudantes aprenderam a desenvolver algoritmos estruturados, aplicar estruturas condicionais e de repetição, manipular dados e compreender o funcionamento de variáveis e funções. Essa fundamentação prática foi essencial para introduzir o pensamento lógico e a decomposição de problemas, princípios indispensáveis em metodologias ágeis voltadas à construção incremental de soluções.

O projeto proposto consistiu no desenvolvimento do *back-end* de um sistema bancário simplificado, que contemplava operações de cadastro de clientes, criação de contas correntes e de investimento, movimentações de crédito e débito e persistência de dados em um banco MySQL. Foram aplicados os princípios de orientação a objetos e práticas de desenvolvimento orientado por testes (TDD), por meio do *framework* JUnit, garantindo a validação automatizada das funcionalidades. A meta de aprovação simultânea dos testes reforçou a importância da qualidade e da verificação contínua do código, em consonância com a abordagem de desenvolvimento orientado a testes descrita por Beck (2002) e com a necessidade de integração entre codificação e validação destacada por Pressman e Maxim (2021).

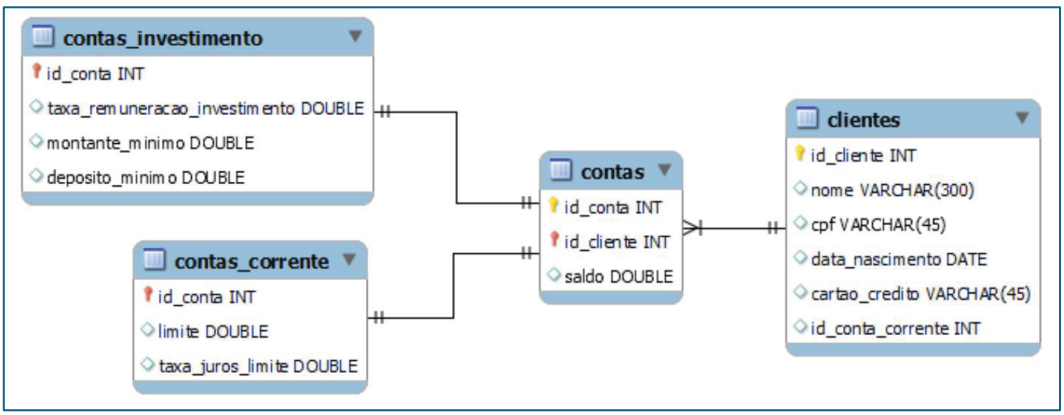
A atividade prática proporcionou uma visão realista do ciclo de vida do *software*, destacando o papel dos testes desde as etapas iniciais do desenvolvimento, um conceito alinhado à visão de Sommerville (2019) sobre a importância da verificação precoce para redução de falhas e retrabalho. Esse entendimento seria ampliado nas disciplinas de Aspectos Ágeis de Programação (AAP), Testes Automatizados (TEST) e Infraestrutura (INFRA), nas quais a automação de testes e a integração contínua ganham papel central. Além disso, o aprendizado sobre modularização e reuso de código contribuiu para a transição natural às disciplinas de Desenvolvimento *Web* e *Mobile*, bem como à Modelagem Ágil de Software (MAG1 e MAG2), que dependem da clareza na estruturação de classes e métodos.

Assim, a disciplina INTRO teve papel estratégico na formação do raciocínio lógico e na aplicação disciplinada de práticas de codificação, consolidando as bases

do desenvolvimento iterativo, incremental e orientado à qualidade que sustentam o desenvolvimento ágil de *software*.

5.1 ARTEFATOS DO PROJETO

FIGURA 15 – DIAGRAMA ER BANCO RW



FONTE: O Autor, 2025.

FIGURA 16 – TESTES UNITÁRIOS JUNIT APROVADOS

FONTE: O Autor, 2025.

6 DISCIPLINA: BD – BANCO DE DADOS

A disciplina de Banco de Dados teve como objetivo introduzir os conceitos, técnicas e práticas fundamentais de modelagem e implementação de sistemas de persistência de dados, essenciais para o desenvolvimento de *software* sob a perspectiva ágil. Foram abordados tópicos como modelagem conceitual e lógica, utilização do modelo Entidade-Relacionamento (E-R), normalização, integridade referencial e construção de esquemas relacionais em SQL. Esses conhecimentos forneceram a base necessária para a criação de sistemas consistentes, escaláveis e alinhados às regras de negócio.

O projeto prático foi dividido em duas etapas, ambas centradas no domínio de controle de empréstimos de livros em uma biblioteca. Na primeira etapa, foi desenvolvido o modelo conceitual por meio de um diagrama Entidade-Relacionamento, no qual foram identificadas as principais entidades do sistema (Usuário, Funcionário, Departamento, Obra, Editora, Empréstimo e Reserva) e seus respectivos relacionamentos. Esse diagrama evidenciou, por exemplo, que um usuário pode realizar vários empréstimos e reservas, que cada empréstimo é registrado por um único funcionário vinculado a um departamento e que cada obra pertence a uma única editora. As cardinalidades associadas a cada relacionamento (1:1, 1:N e N:N) foram definidas para representar com fidelidade as regras de negócio da biblioteca e garantir a consistência dos dados.

Na segunda etapa, esse modelo conceitual foi refinado em um diagrama de esquema relacional, detalhando as tabelas correspondentes às entidades e relacionamentos e especificando atributos, tipos de dados, chaves primárias e chaves estrangeiras. Foram estruturadas tabelas como `Usuario`, `Funcionario`, `Departamento`, `Obra`, `Editora`, `Emprestimo`, `Reserva` e `Emprestimo_Obra`, esta última responsável por representar a relação N:N entre empréstimos e obras. Essa etapa permitiu discutir a integridade referencial e a implementação prática das cardinalidades definidas no modelo conceitual, aproximando a modelagem dos cenários reais de implementação em sistemas gerenciadores de banco de dados relacionais.

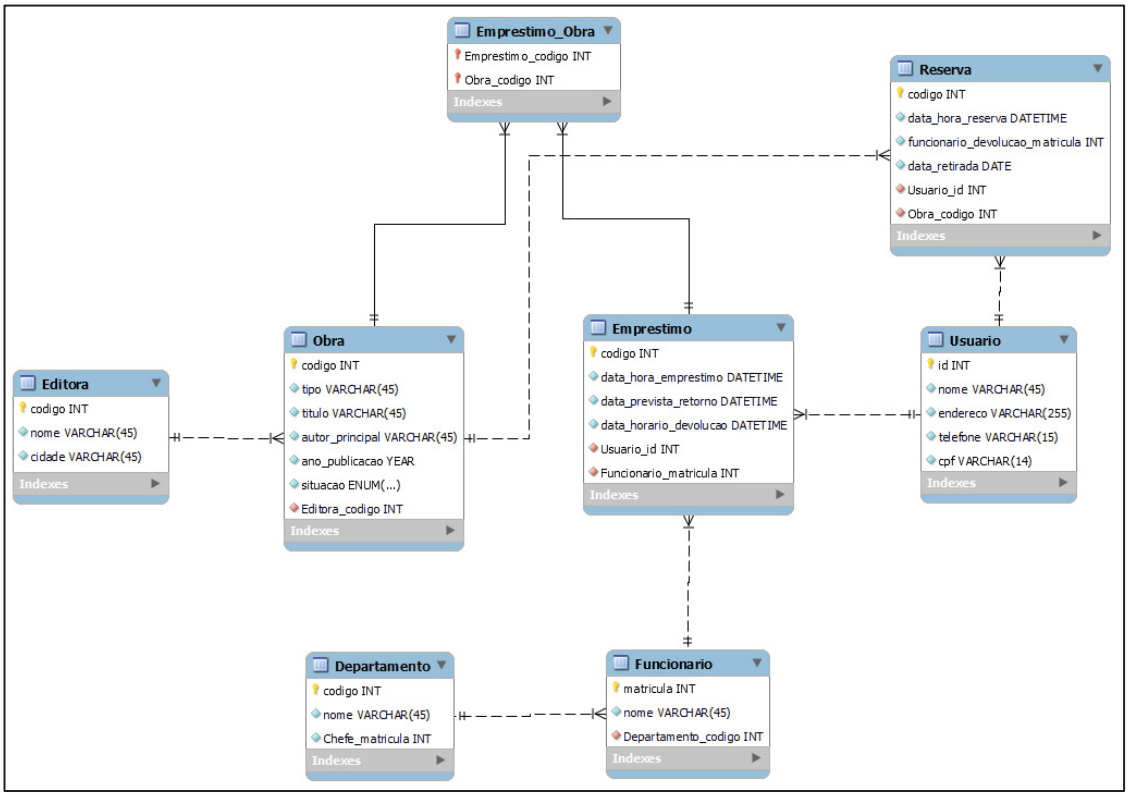
A disciplina teve papel central na integração das demais áreas do curso, servindo como elo entre a modelagem (MAG1 e MAG2), a programação (INTRO e

AAP) e a infraestrutura (INFRA). A clareza na definição dos modelos de dados possibilitou a conexão direta com os projetos desenvolvidos nas disciplinas de Desenvolvimento *Web* e *Mobile*, garantindo que as aplicações pudessem acessar e manipular dados de maneira eficiente e segura. Autores clássicos de Banco de Dados, como Elmasri e Navathe (2019) e Silberschatz, Korth e Sudarshan (2020), destacam que um projeto de banco de dados bem estruturado é fundamental para assegurar a integridade e a consistência das informações, além de suportar a evolução das aplicações de software.

Dessa forma, a disciplina BD consolidou as bases para a persistência e integridade dos dados, nas quais o domínio das técnicas de SQL e da estruturação lógica dos dados contribuiu para a qualidade, consistência e manutenção contínua dos sistemas desenvolvidos ao longo do curso.

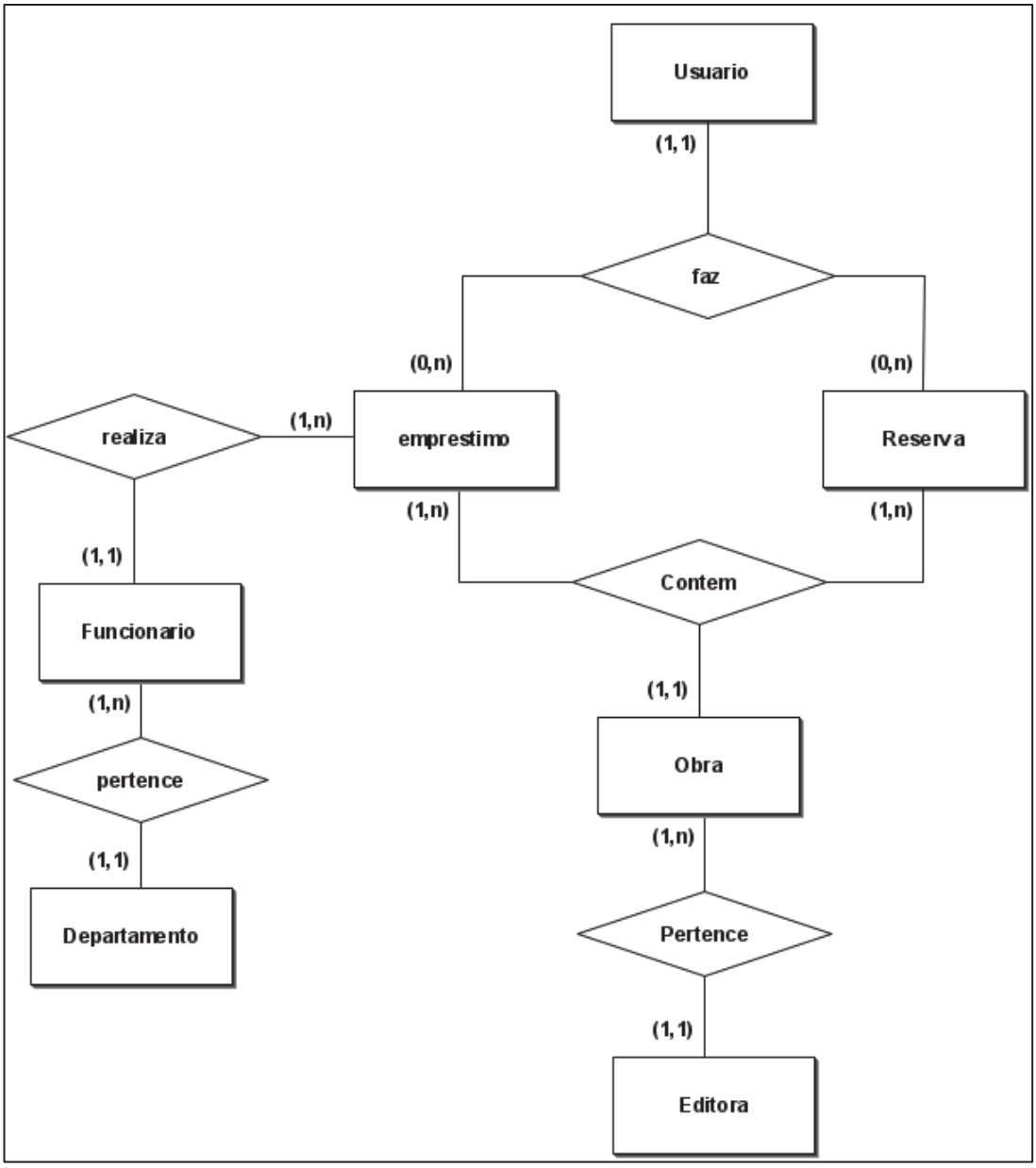
6.1 ARTEFATOS DO PROJETO

FIGURA 17 – DIAGRAMA ER CONTROLE BIBLIOTECA



FONTE: O Autor, 2025.

FIGURA 18 – MODELO CONCEITUAL CONTROLE BIBLIOTECA



FONTE: O Autor, 2025.

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como objetivo consolidar as boas práticas de codificação e o uso disciplinado de técnicas que sustentam a qualidade e a adaptabilidade do *software* em ambientes ágeis. Foram estudados os princípios SOLID, a refatoração contínua, o uso de testes automatizados, o controle de versionamento com Git, o desenvolvimento orientado a testes (TDD) e os padrões de projeto aplicáveis a contextos iterativos e incrementais. Esses conhecimentos reforçam a importância de um código limpo, modular e de fácil manutenção, pois são requisitos essenciais para sustentar o ciclo de entrega contínua, em linha com a defesa do *Clean Code* e da responsabilidade bem definida apresentada por Martin (2017) e com a visão de refatoração incremental proposta por Fowler (2020).

O projeto prático envolveu a aplicação dos princípios de *clean code* e TDD sobre um algoritmo de ordenação (*Bubble Sort*), exigindo refatorações significativas no código original. O método `bubbleSort(int arr[], int n)` foi reestruturado como um método `sort(int[] values)`, eliminando o parâmetro redundante "`n`" e utilizando diretamente o tamanho do vetor, o que reduziu o acoplamento e simplificou a assinatura. Entre as melhorias implementadas, destacam-se a extração de um método específico `swap(int[] values, int firstIndex, int secondIndex)` para realizar a troca de elementos, a reorganização da lógica de ordenação em um método dedicado, a renomeação de variáveis para torná-las mais descritivas (como `values`, `currentIndex` e `lastIndexToCheck`) e o uso do método `Arrays.toString()` para a exibição do resultado da ordenação, em substituição ao laço manual de impressão. Essas alterações ilustraram como pequenas intervenções incrementais, apoiadas por testes automatizados, podem melhorar a legibilidade, reduzir a complexidade e aumentar a confiabilidade do código, promovendo maior colaboração e transparência entre os membros da equipe.

Além de fortalecer o domínio técnico sobre boas práticas de programação, a disciplina teve papel fundamental na integração com outras áreas do curso. O aprendizado sobre modularidade e testes complementou as bases estabelecidas em Introdução à Programação (INTRO) e em Banco de Dados (BD), enquanto o uso de TDD e versionamento se estendeu às disciplinas de Testes Automatizados (TEST) e Infraestrutura (INFRA). Os princípios de qualidade contínua e inspeção constante

também se refletiram nas atividades de Gerenciamento Ágil de Projetos (GAP1 e GAP2), promovendo entregas incrementais e sustentáveis.

Assim, Aspectos Ágeis de Programação consolidou os fundamentos técnicos e culturais necessários para o desenvolvimento ágil de software, demonstrando que a qualidade do código é um componente inseparável da agilidade e da capacidade de adaptação do processo de desenvolvimento.

7.1 ARTEFATOS DO PROJETO

CÓDIGO ORIGINAL BUBBLE SORT

```
// Código extraído de https://www.geeksforgeeks.org/bubble-sort/

import java.io.*;

class BubbleSort {

    // An optimized version of Bubble Sort
    static void bubbleSort(int arr[], int n)
    {
        int i, j, temp;
        boolean trocado;
        for (i = 0; i < n - 1; i++) {
            trocado = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {

                    // Swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    trocado = true;
                }
            }

            // If no two elements were
            // trocado by inner loop, then break
            if (trocado == false)
                break;
        }
    }

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

    // Driver program
    public static void main(String args[])
    {
        int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
        int n = arr.length;
        bubbleSort(arr, n);
        System.out.println("Array ordenado: ");
        printArray(arr, n);
    }
}
```

CÓDIGO BUBBLE SORT - REFATORADO

```
import java.util.Arrays;

public class BubbleSort {

    public static void sort(int[] values) {
        boolean swapped;
        int lastIndexToCheck = values.length - 1;

        do {
            swapped = false;
            for (int currentIndex = 0; currentIndex < lastIndexToCheck;
currentIndex++) {
                if (values[currentIndex] > values[currentIndex + 1]) {
                    swap(values, currentIndex, currentIndex + 1);
                    swapped = true;
                }
            }
            lastIndexToCheck--;
        } while (swapped);
    }

    private static void swap(int[] values, int firstIndex, int secondIndex) {
        int temp = values[firstIndex];
        values[firstIndex] = values[secondIndex];
        values[secondIndex] = temp;
    }

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

        sort(numbers);

        System.out.println("Array ordenado: " + Arrays.toString(numbers));
    }
}
```

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

As disciplinas de Desenvolvimento Web I e II foram fundamentais para consolidar a aplicação prática dos conceitos de *front-end* e *back-end* no desenvolvimento de sistemas *web* completos. Em WEB1, o foco esteve na introdução à plataforma Angular, passando pela estruturação de projetos, configuração do ambiente, uso de TypeScript e HTML, e construção de interfaces responsivas. Inicialmente, foram desenvolvidos exemplos simples, como o projeto "Soma", para explorar variáveis, funções, componentes, *templates* HTML e diretivas básicas, em linha com a abordagem introdutória apresentada por Machado (2021) para o uso integrado de Angular e serviços de *back-end*. Na sequência, evoluindo para um projeto de CRUD, no qual foram implementados dois módulos: Alunos e Cursos com telas de listagem, inserção, edição e remoção, utilizando *Local Storage* para persistência dos dados e aplicando padrões de *layout* com Bootstrap e Material Design. Essa etapa consolidou a compreensão de componentes, serviços, roteamento e organização modular do código, em consonância com a arquitetura baseada em componentes descrita na documentação oficial do Angular (ANGULAR, 2025).

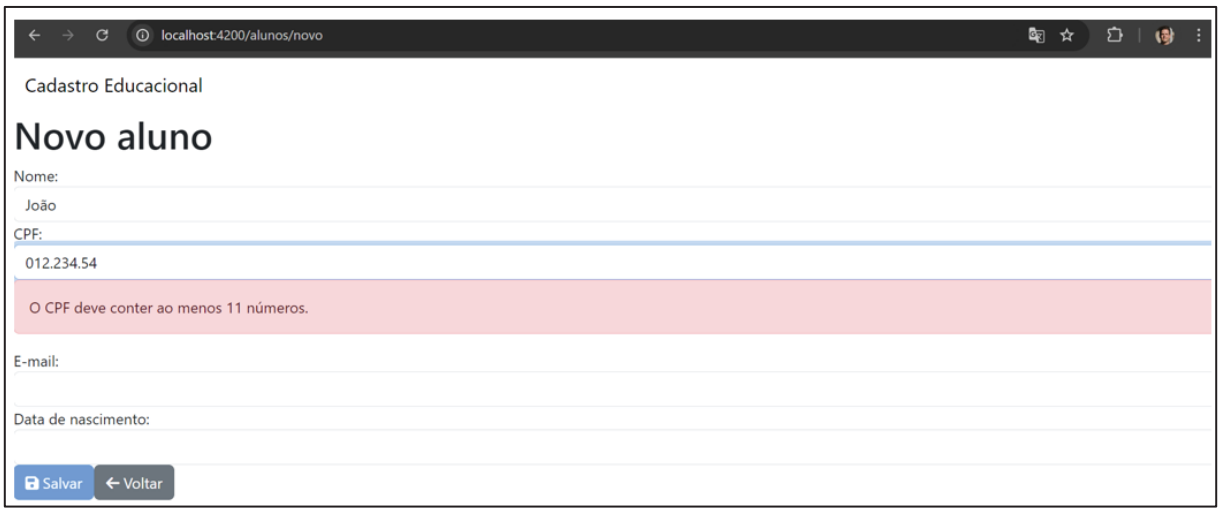
Na disciplina WEB2, o escopo foi ampliado para contemplar formulários avançados, validação de campos, diretivas personalizadas, máscaras, *pipes*, uso de *modals* e criação de menus de navegação. A disciplina enfatizou o consumo de APIs REST a partir do *front-end* Angular, introduzindo conceitos de programação reativa, autenticação básica e integração com serviços remotos. No *back-end*, foi desenvolvido um servidor em Java com o *framework* Spring Boot, responsável por expor APIs REST para três módulos: Alunos, Cursos e Matrículas. O banco de dados PostgreSQL foi modelado com três tabelas principais: `Aluno`, `Curso` e `Matricula`, sendo utilizado o Spring Data JPA para o mapeamento objeto-relacional. O projeto foi organizado em camadas de apresentação, serviço e persistência, reforçando princípios de arquitetura limpa e evolução sustentável do código, em linha com as recomendações de Pressman e Maxim (2021) e Sommerville (2019) sobre separação de responsabilidades e projeto de *software* orientado à manutenção.

Assim, WEB1 e WEB2 representaram um marco no curso, ao permitir que os alunos aplicassem de forma integrada os conceitos estudados, resultando em um sistema funcional completo que reflete os princípios do desenvolvimento ágil —

colaboração, entrega incremental, qualidade contínua e foco no valor entregue ao usuário.

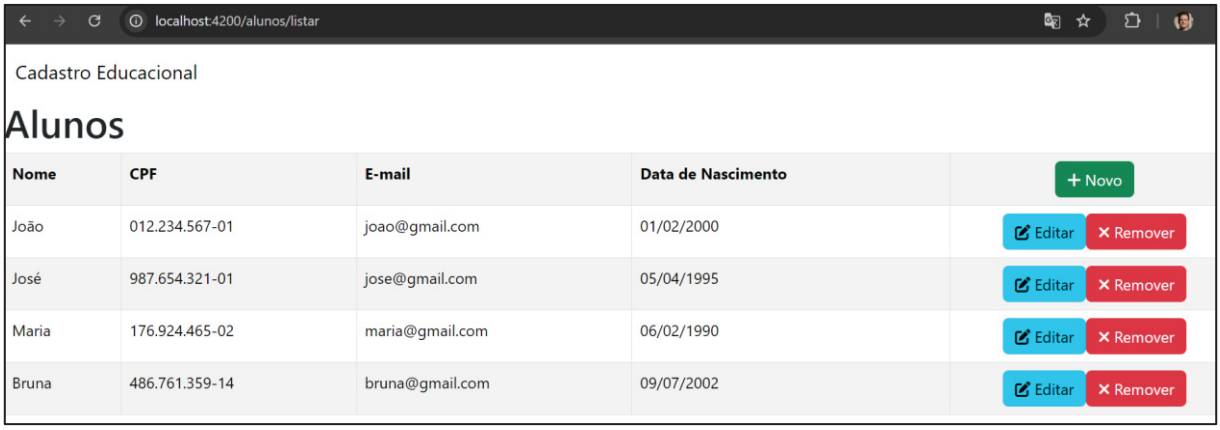
8.1 ARTEFATOS DO PROJETO

FIGURA 19 – TELA CADASTRO DE ALUNO



FONTE: O Autor, 2025.

FIGURA 20 – TELA CONSULTA CADASTRAL DE ALUNOS



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 a integração da experiência do usuário ao processo iterativo de desenvolvimento, destacando a importância de projetar soluções centradas nas necessidades reais das pessoas. Foram abordados princípios de usabilidade, arquitetura da informação, *design* de interação e acessibilidade, apoiando-se em fundamentos de *design* centrado no usuário e interação humano–computador apresentados por Norman (2024) e por Sharp, Preece e Rogers (2019), bem como nas diretrizes de usabilidade para interfaces digitais discutidas por Nielsen e Budiu (2013). A abordagem evidenciou que a experiência de uso deve ser considerada desde o planejamento do produto, garantindo que o *software* entregue valor contínuo e possa ser ajustado a partir do *feedback* dos usuários.

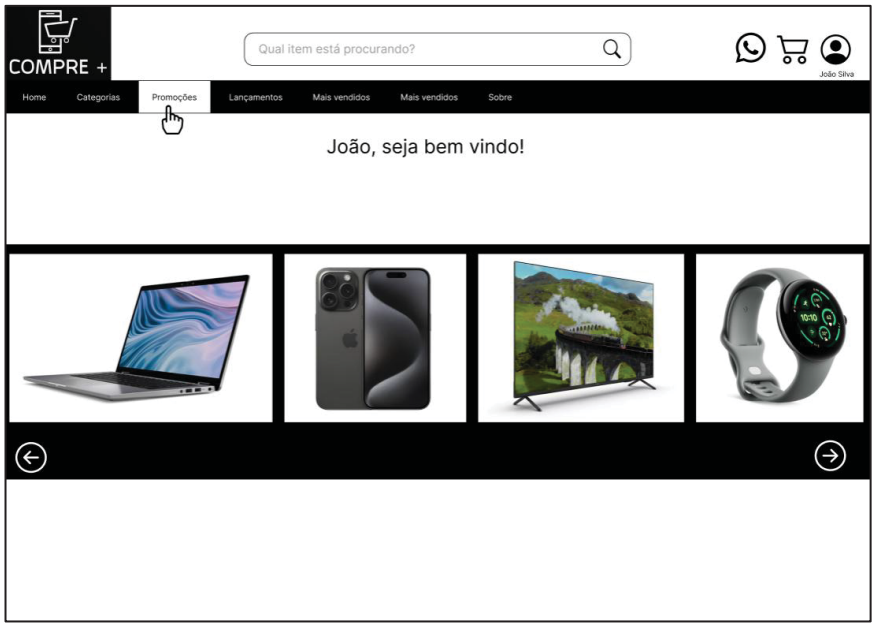
O projeto desenvolvido consistiu na elaboração de um protótipo de interface, construído na ferramenta Figma, para um *site* de *e-commerce* de artigos eletrônicos denominado Compre+. O sistema foi projetado para oferecer aos clientes acesso a produtos em promoção e lançamentos do mundo digital. Foram criadas cinco telas principais: *Login*, Cadastro de Usuário, Produtos em Promoção, Lançamentos e Carrinho de Compras, seguindo diretrizes de consistência, legibilidade, hierarquia visual e acessibilidade. O *design* utilizou cores contrastantes (elementos em preto sobre fundo claro) e fontes legíveis, alinhadas à identidade visual da marca, assegurando clareza e coerência em diferentes dispositivos. Essas escolhas ocorreram em consonância com as boas práticas de organização visual, hierarquia e psicologia cognitiva aplicadas ao *design* de interfaces discutidas por Grant (2019) e Yablonski (2020) e com a perspectiva de *design* centrado no usuário apresentada por Lowdermilk (2013).

A disciplina integrou-se diretamente às áreas de modelagem, desenvolvimento e gerenciamento de projetos. Os protótipos criados em UX serviram como referência para a implementação nas disciplinas de Desenvolvimento *Web* (WEB1 e WEB2) e *Mobile* (MOB1 e MOB2), enquanto a iteração e validação contínua reforçaram os princípios aplicados em Gerenciamento Ágil (GAP1 e GAP2). Em especial, a disciplina enfatizou a importância de incorporar práticas de UX em ciclos curtos de desenvolvimento, em linha com a proposta de integração entre UX e

métodos ágeis discutida por Curcio (2021) e com a abordagem de experimentação, aprendizado contínuo e *Lean UX* apresentada por Gothelf e Seiden (2021) e Levy (2021). Como destacam Pressman e Maxim (2021), o *design* centrado no usuário é elemento-chave para assegurar qualidade e satisfação em produtos de *software*. Assim, a disciplina de UX consolidou-se como elo entre empatia, *design* e tecnologia, fortalecendo a entrega de valor no contexto do desenvolvimento ágil.

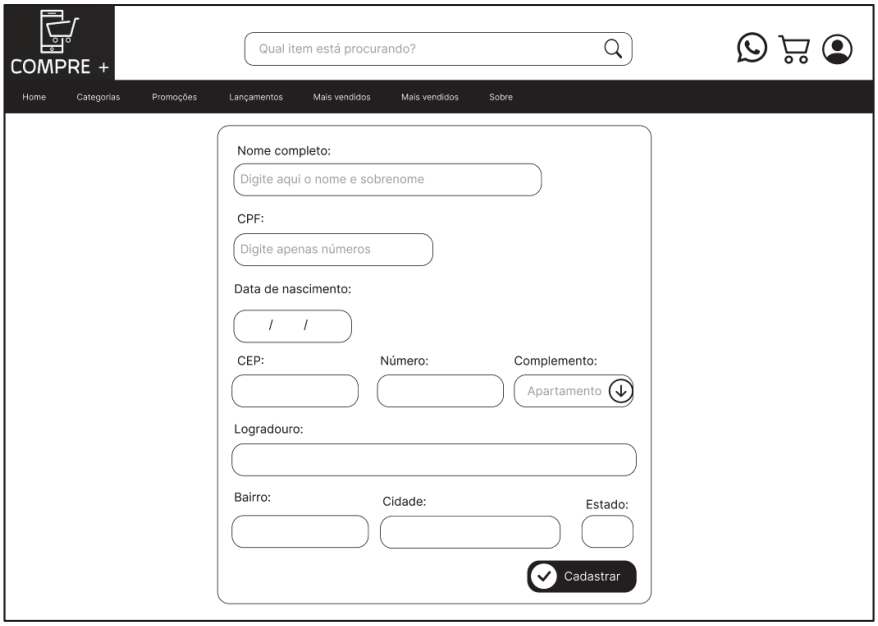
9.1 ARTEFATOS DO PROJETO

FIGURA 21 – TELA PROTÓTIPO PRINCIPAL E-COMMERCE



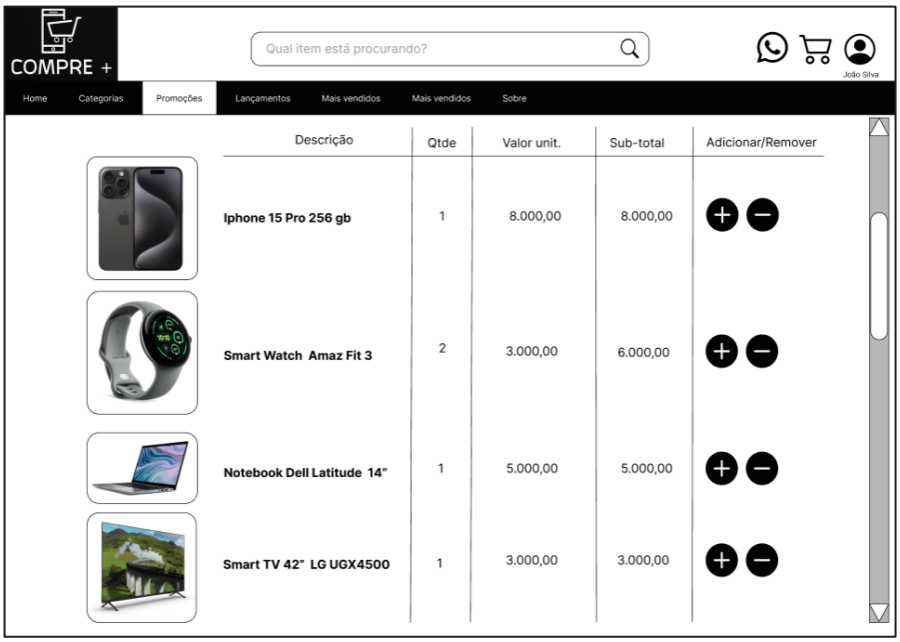
FONTE: O Autor, 2025.

FIGURA 22 – TELA PROTÓTIPO – CADASTRO CLIENTE



FONTE: O Autor, 2025.

FIGURA 23 – TELA PROTÓTIPO – CARRINHO DE COMPRA



FONTE: O Autor, 2025.

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

As disciplinas de Desenvolvimento Mobile I e II tiveram como propósito capacitar os alunos na criação de aplicativos nativos para Android, utilizando a linguagem Kotlin e o ambiente Android Studio. A partir dos princípios do desenvolvimento ágil, as atividades enfatizaram a prototipação rápida, a entrega incremental e o refinamento contínuo de funcionalidades, favorecendo a adaptação a mudanças de requisitos e o aprendizado iterativo. Os conteúdos dialogaram com boas práticas de desenvolvimento apresentadas em obras voltadas à plataforma Android com Kotlin, como Glauber (2019), Jemerov e Isakova (2017) e Lecheta (2018), bem como com a documentação oficial da plataforma Android, fornecida pela equipe do Android (ANDROID, 2025).

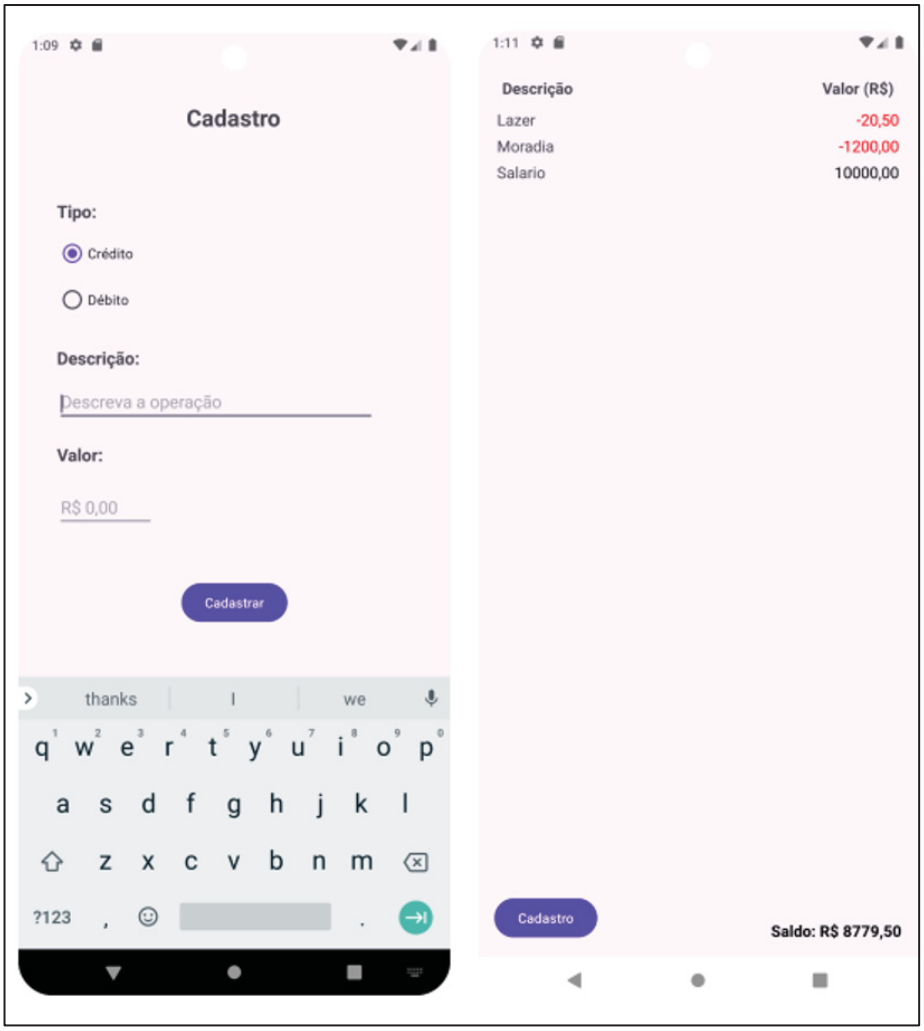
Em MOB1, foi desenvolvido o projeto FinApp, um aplicativo de controle financeiro pessoal que permitia cadastrar e visualizar movimentações de crédito e débito, simulando um extrato bancário simplificado. O desenvolvimento envolveu o uso de `RecyclerView` para listagem de itens, `SharedPreferences` para persistência local de dados, navegação entre telas por meio de *intents* e a aplicação de boas práticas de *layout* responsivo. Esse projeto reforçou conceitos de modularidade, organização de código em camadas e reutilização de componentes, além de integrar noções de *design* centrado no usuário estudadas em UX e conceitos de estruturação de informações e modelos de dados trabalhados em Banco de Dados.

Em MOB2, o projeto teve como base o consumo da HP-API, uma API pública com informações sobre o universo de Harry Potter. O desafio consistiu em realizar requisições HTTP assíncronas com o uso de corrotinas em Kotlin e consumir *endpoints* REST por meio da biblioteca *Retrofit*, alinhado às recomendações das documentações oficiais de Android e Kotlin para programação assíncrona e acesso a serviços remotos. A aplicação permitia listar personagens, professores e alunos por casa, além de realizar buscas por ID. Essa experiência aprofundou a compreensão de padrões arquiteturais como MVC e MVVM, enfatizando a separação de responsabilidades entre camadas de interface, lógica de negócio e acesso a dados, e complementou a vivência de integração com *back-ends* REST iniciada nas disciplinas WEB1 e WEB2.

Ambos os projetos destacaram a importância da entrega iterativa e da integração contínua, pilares do desenvolvimento ágil. As disciplinas Mobile consolidaram o domínio técnico sobre *interfaces* gráficas, consumo de serviços e organização de código, além de reforçar a interdisciplinaridade com UX, *Web*, Banco de Dados e Gerenciamento Ágil de Projetos. Conforme Pressman e Maxim (2021), o desenvolvimento ágil requer ciclos curtos de *feedback* e entrega de valor contínua, princípios aplicados de forma prática em MOB1 e MOB2, que proporcionaram uma visão completa do desenvolvimento moderno de aplicativos móveis.

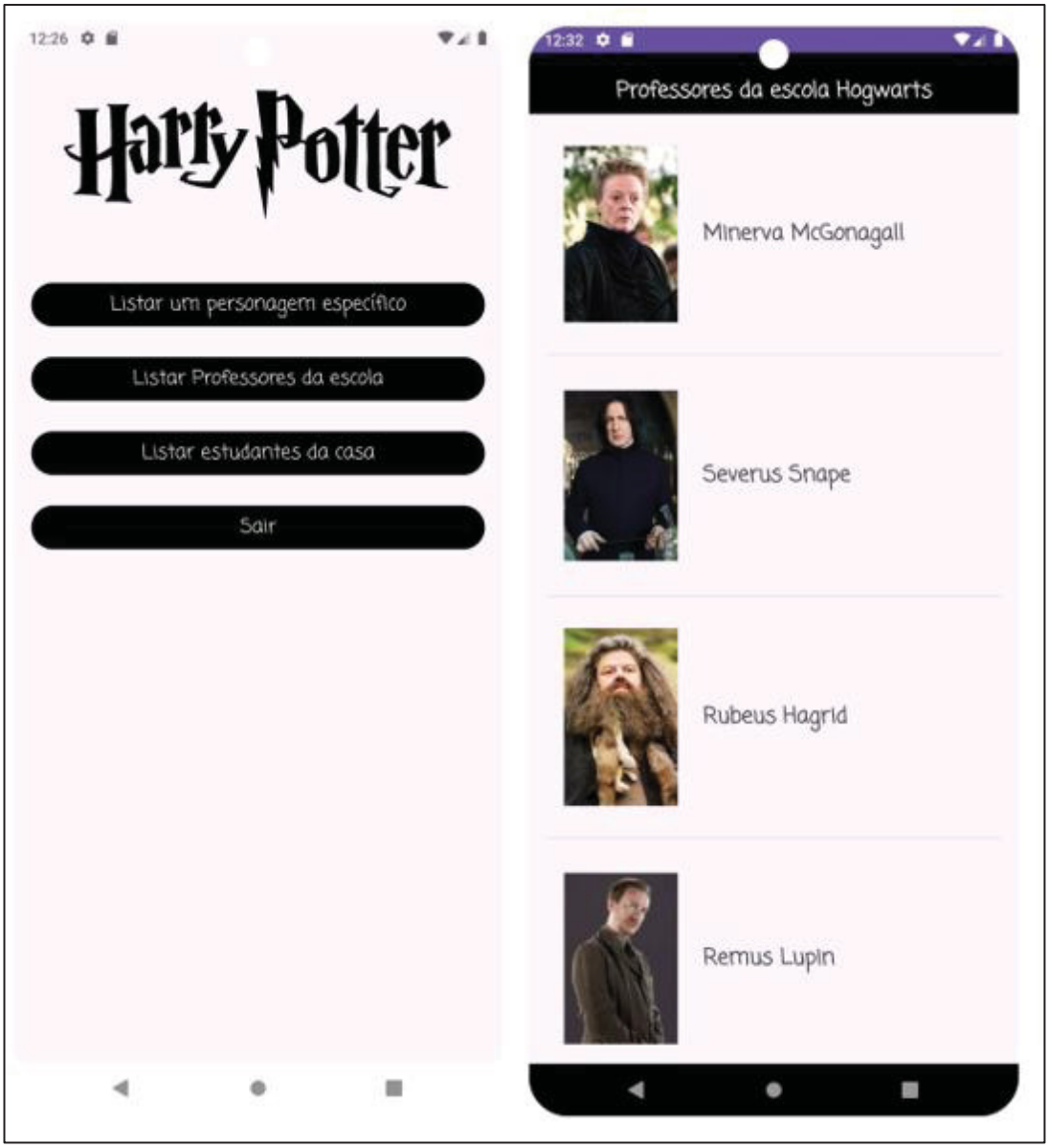
10.1 ARTEFATOS DO PROJETO

FIGURA 24 – TELAS DO APLICATIVO FINAPP



FONTE: O Autor, 2025.

FIGURA 25 – TELAS DO APLICATIVO HARRY POTTER API



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 foco a aplicação dos princípios de *DevOps* e da automação de ambientes no contexto do desenvolvimento ágil. Foram estudados o ciclo de vida de desenvolvimento de *software* (SDLC), conceitos de *DevOps*, métricas e maturidade de processos, sistemas de controle de versão, integração contínua (CI), entrega contínua (CD), conteinerização com Docker, orquestração de serviços com Kubernetes e práticas de *Infrastructure as Code* (IaC). Em consonância com Kim *et al.* (2021), a cultura *DevOps* foi apresentada como elo entre desenvolvimento, operações e qualidade, promovendo colaboração, automação e *feedback* contínuo ao longo de todo o ciclo de vida do *software*, enquanto Chacon e Straub (2020) reforçam o papel central do Git como base para versionamento e rastreabilidade de mudanças.

O projeto prático da disciplina consistiu na implantação de um ambiente integrado com GitLab e Jenkins executado via Docker, utilizando a imagem `dfwandarti/gitlab_jenkins:3`. O exercício envolveu a criação de um *container* identificado pela matrícula do aluno, a publicação das portas 22, 80, 443 e 9091 e a configuração do acesso `root` no GitLab. Após o *login* e recuperação da senha inicial, foi realizado o *commit* e *push* de um projeto para o repositório GitLab, seguido da configuração de *pipelines* de CI/CD e da integração com o Jenkins, consolidando o fluxo automatizado de *build*, teste e *deploy*. Essa prática reforçou os conceitos de conteinerização e automação descritos por Vitalino e Castro (2018) e evidenciou a importância da rastreabilidade, da integração contínua e do monitoramento de *pipelines* como indicadores de maturidade *DevOps*, em linha com as discussões de Kim *et al.* (2021).

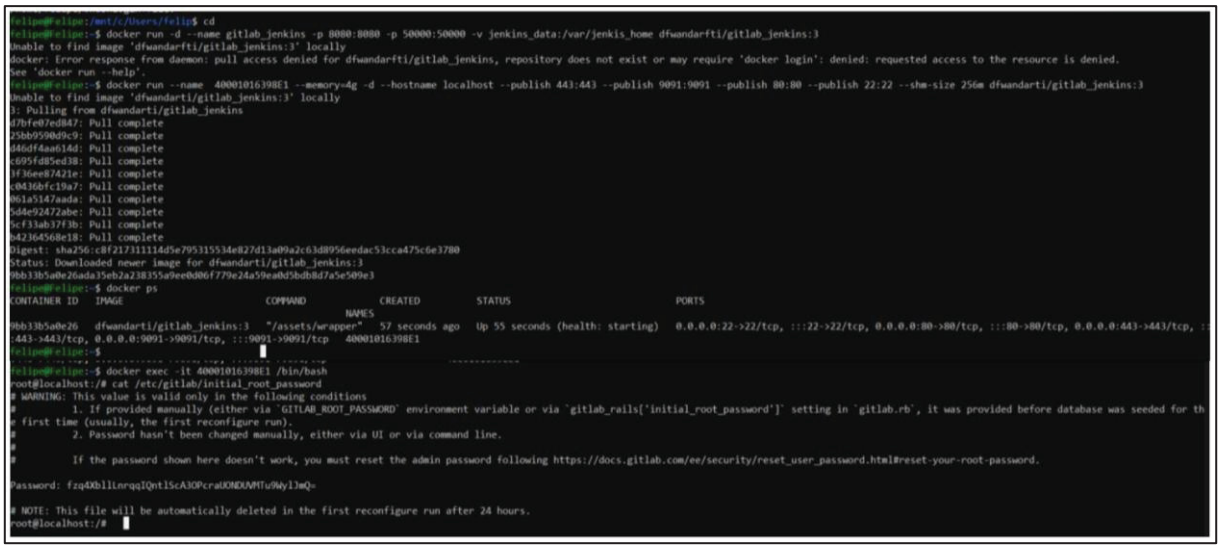
Além da implementação prática com GitLab, Jenkins e Docker, a disciplina introduziu conceitos de orquestração de *containers* com Kubernetes, incluindo *deployments*, *replicasets*, *services* e estratégias de armazenamento e configuração, com base nas abordagens de Burns, Beda e Hightower (2020). Foram discutidas também práticas de observabilidade: métricas, *logs* e *tracing*, bem como elementos fundamentais para compreender o comportamento de sistemas distribuídos e identificar gargalos, conforme ressaltam Majors, Jones e Miranda (2022). A

perspectiva de *Infrastructure as Code*, inspirada em Wang (2022), foi apresentada como forma de tratar a infraestrutura como artefato versionável, permitindo reprodutibilidade de ambientes, auditoria de mudanças e alinhamento entre código de aplicação e configuração de infraestrutura.

Essa visão unificada de desenvolvimento, testes, implantação e operação permitiu automatizar processos, reduzir erros humanos e aproximar as atividades de desenvolvimento das operações, criando um ambiente propício à integração com os projetos de Desenvolvimento *Web*, Desenvolvimento Mobile e Testes Automatizados, que puderam ser construídos e entregues por meio de *pipelines* reprodutíveis. Assim, a disciplina INFRA consolidou o entendimento de que o sucesso do desenvolvimento ágil depende não apenas da entrega rápida de funcionalidades, mas também da estabilidade, observabilidade e previsibilidade da infraestrutura. O *DevOps* se apresentou, portanto, como componente estratégico do processo ágil, sustentando a entrega contínua de valor e o aprimoramento colaborativo dos sistemas de *software*.

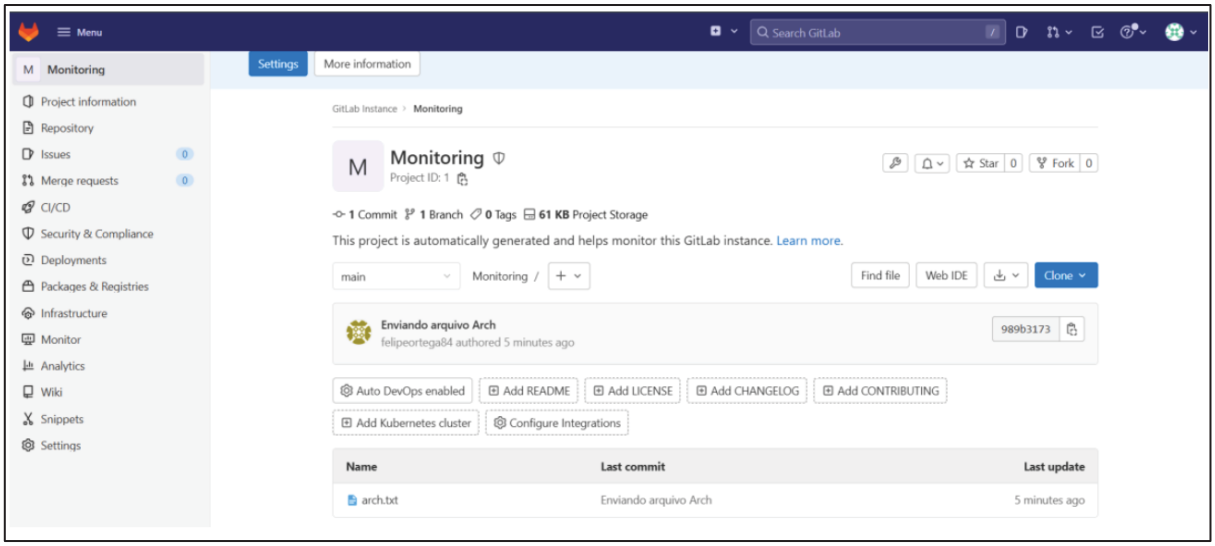
11.1 ARTEFATOS DO PROJETO

FIGURA 26 – CRIANDO CONTAINER E IDENTIFICANDO AS PORTAS



FONTE: O Autor, 2025.

FIGURA 27 – GIT COMMIT E PUSH



FONTE: O Autor, 2025.

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados teve como objetivo consolidar as práticas de verificação e validação de *software*, essenciais para a entrega de produtos com qualidade e confiabilidade dentro de ciclos de desenvolvimento ágil. Foram estudados diferentes níveis de teste: unitário, integração, sistema e aceitação, bem como o papel dos testes automatizados no suporte a TDD (*Test-Driven Development*) e BDD (*Behavior-Driven Development*). Beck (2002) apresenta o TDD como uma prática em que os testes orientam o *design* do código, enquanto North (2006) descreve o BDD como uma evolução focada em comportamentos observáveis e na colaboração entre negócio e desenvolvimento. Em complemento, Pressman e Maxim (2021) e Humble e Farley (2013) reforçam a importância da automação de testes e da sua integração aos processos de integração e entrega contínua.

Na etapa prática, inicialmente foram desenvolvidos testes unitários em Java utilizando o *framework* JUnit, com o uso de *asserts*, marcações, *stubs* e *mocks*, além da análise de cobertura para apoiar a escrita de código testável. Em seguida, foi proposto um projeto integrador de testes de interface, no qual foi desenvolvido um *script* automatizado capaz de interagir com a interface gráfica de um navegador para validar funcionalidades em uma aplicação *web*. Utilizando o *framework* Playwright e a linguagem JavaScript, o programa acessava a plataforma aNotepad, criava uma nova nota preenchendo automaticamente o nome do aluno e o número da matrícula, validava a operação por meio de asserções e finalizava a execução. Essa experiência permitiu aplicar os princípios de testes *end-to-end* (E2E), simulando o comportamento real do usuário e garantindo que os fluxos críticos do sistema estivessem operando corretamente, em linha com as recomendações da documentação oficial do Playwright (PLAYWRIGHT, 2025).

O exercício demonstrou a relevância da automação de testes no contexto ágil, em que a entrega contínua requer validações rápidas e confiáveis. A disciplina destacou que a automação reduz erros humanos, acelera a detecção de falhas e fortalece a integração entre equipes de desenvolvimento e operações, conforme defendem Kim *et al.* (2021) na cultura *DevOps* e na relação entre testes, integração contínua e entrega contínua. Além disso, promoveu a interdisciplinaridade com as

disciplinas de Programação, Infraestrutura e Desenvolvimento Web, ao permitir a execução automatizada de testes em ambientes integrados e controlados.

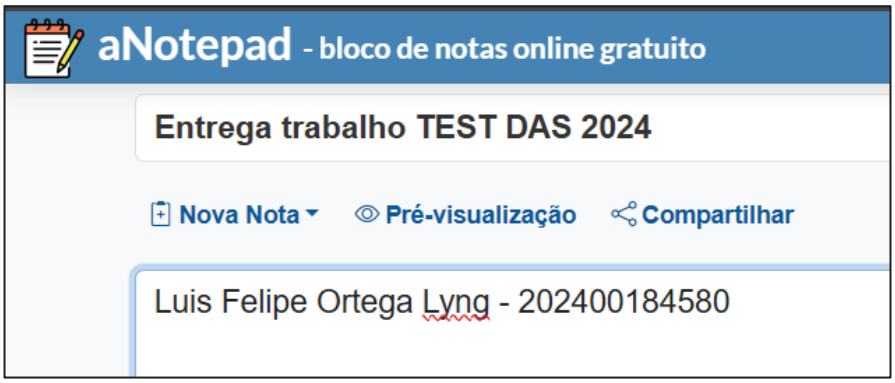
Assim, a disciplina de Testes Automatizados consolidou-se como um elo essencial entre desenvolvimento e qualidade, garantindo que cada incremento de *software* seja validado de forma contínua e previsível. A aplicação prática dos conceitos de TDD e BDD evidenciou que a automação não apenas assegura a robustez do código, mas também potencializa a agilidade e a confiança nos processos de entrega de *software*.

12.1 ARTEFATOS DO PROJETO

CÓDIGO PLAYWRIGHT - TESTE

```
const { test } = require('@playwright/test');

test('Entrega trabalho TEST DAS 2024', async ({ page }) => {
  await page.goto('https://pt.anotepad.com/', { waitUntil: 'domcontentloaded' });

  await page.getByRole('textbox', { name: 'Título da Nota' })
            .fill('Entrega trabalho TEST DAS 2024');

  await page.getByRole('textbox', { name: 'Conteúdo da Nota' })
            .fill('Luis Felipe Ortega Lyng - 202400184580');

  await page.getByRole('button', { name: 'Salvar' })
            .click();

  // Pausar antes de fechar (fica aberto até você fechar)
  await page.pause();
});
```

FIGURA 28 – EXECUÇÃO DO PLAYWRIGHT NO ANOTEPAD



FONTE: O Autor, 2025.

13 CONCLUSÃO

A especialização em Desenvolvimento Ágil de Software proporcionou uma formação integrada e prática, permitindo compreender como os princípios ágeis se aplicam de forma efetiva em todo o ciclo de desenvolvimento de sistemas. Em consonância com Beck *et al.* (2001), o curso evidenciou que valores como colaboração, adaptação e entrega contínua de valor são o eixo central da engenharia de *software* moderna, complementando a visão de processo e qualidade discutida por Pressman e Maxim (2021) e Sommerville (2019).

A disciplina de Métodos Ágeis (MADS) introduziu os valores do Manifesto Ágil e *frameworks* como *Scrum* e *Kanban*, que orientaram todas as demais etapas. A Modelagem Ágil (MAG1 e MAG2) reforçou a importância da análise evolutiva e da modelagem enxuta, enquanto o Gerenciamento Ágil de Projetos (GAP1 e GAP2) consolidou práticas de planejamento incremental, priorização e monitoramento contínuo do fluxo de trabalho, em linha com as contribuições de Cohn (2011) e Anderson (2011) sobre gestão de projetos em contextos ágeis.

As disciplinas técnicas aplicaram esses princípios de forma concreta. Introdução à Programação e Aspectos Ágeis de Programação (AAP) desenvolveram o raciocínio lógico, o código limpo e o uso disciplinado de testes, apoiando-se em práticas como refatoração contínua e TDD discutidas por Fowler (2020) e Beck (2002). Banco de Dados (BD) estruturou a persistência e a integridade das informações, enquanto Desenvolvimento *Web* e *Mobile* (WEB1/WEB2 e MOB1/MOB2) integraram as camadas de apresentação, serviço e dados, refletindo a visão de arquitetura em camadas e manutenibilidade defendida por Pressman e Maxim (2021). A disciplina de UX trouxe a perspectiva centrada no usuário, alinhando o *design* de interfaces aos conceitos de usabilidade, interação e experiência de uso apresentados por autores como Norman (2024), Sharp, Preece e Rogers (2019) e Nielsen e Budiu (2013).

Infraestrutura (INFRA) e Testes Automatizados (TEST) sustentaram a base técnica do ciclo ágil, com foco em automação, integração e entrega contínua. O estudo de práticas de *DevOps*, conteinerização, CI/CD e *Infrastructure as Code* aproximou a formação das recomendações de Kim *et al.* (2021) sobre fluxos contínuos de *build*, teste e *deploy*. Já a disciplina de Testes Automatizados reforçou a importância de TDD, BDD e testes E2E como mecanismos de verificação sistemática da qualidade,

em acordo com Beck (2002) e Humble e Farley (2013). Juntas, essas disciplinas formaram um ecossistema de aprendizado interdisciplinar e colaborativo, que espelha os princípios da engenharia moderna de *software.*

Entre os desafios para adoção prática do ágil, destacam-se a mudança cultural nas organizações, a resistência à adaptação contínua e a necessidade de maturidade técnica para aplicar práticas como CI/CD, automação de testes e monitoramento de sistemas. Esses pontos dialogam com as análises de Kim *et al.* (2021) e com as estratégias de melhoria contínua e *Lean Flow* discutidas por Walter *et al.* (2015), que evidenciam que a transição para o ágil exige ajustes graduais de processos, estruturas e comportamentos.

Conclui-se que o curso consolidou a compreensão de que o desenvolvimento ágil vai além de métodos e ferramentas: trata-se de uma mentalidade orientada à entrega de valor, ao aprendizado contínuo e à colaboração entre pessoas e equipes, conforme já apontado por Beck *et al.* (2001). Essa perspectiva representa o principal legado da especialização e o alicerce para uma prática profissional moderna, eficiente e genuinamente centrada nas pessoas.

REFERÊNCIAS

AMARAL, D. C.; CONFORTO, E. C.; BENASSI, J. L. G.; *et al.* **Gerenciamento ágil de projetos: aplicação em produtos inovadores**. São Paulo: Saraiva, 2013.

ANDERSON, D. J. **Kanban: mudança evolucionária para seu negócio de tecnologia**. Seattle, EUA: Blue Hole Press, 2011.

ANDROID. Desenvolver para Android. Disponível em: https://developer.android.com/develop?hl=pt-br. Acesso em: 26 nov. 2025.

ANGULAR. The framework for building scalable web apps with confidence. Disponível em: https://angular.io. Acesso em: 28 nov. 2025.

ANOTEPAD. aNotepad - free online notepad. Disponível em: https://anotepad.com/. Acesso em: 29 nov. 2025.

BECK, K. **Test-Driven Development: By Example**. Boston, EUA: Addison-Wesley Professional, 2002.

BECK, K. **Programação extrema (XP) aplicada: acolha as mudanças**. Porto Alegre: Bookman, 2004.

BECK, K.; FOWLER, M.; COCKBURN, A.; *et al.* Manifesto para o Desenvolvimento Ágil de Software. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 29 nov. 2025.

BOURQUE, P.; FAIRLEY, R. E. **Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 3.0**. Washington, EUA: IEEE Computer Society, 2014.

BURNS, B.; BEDA, J.; HIGHTOWER, K. **Kubernetes Básico: Mergulhe no futuro da infraestrutura**. São Paulo: Novatec, 2020.

CAMARGO, R.; RIBAS, T. **Gestão ágil de projetos: As melhores soluções para suas necessidades**. São Paulo: Saraiva Uni, 2019.

CHACON, S.; STRAUB, B. **Pro Git**. 2. ed. Nova York, EUA: Apress, 2020.

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

CURCIO, K. P. de C. **An approach for user experience design integration into agile software development**. 2021. 265 f. Tese (Doutorado em Informática), Pontifícia Universidade Católica do Paraná. Curitiba, PR, 2021. Disponível em: https://www.ppgia.pucpr.br/pt/arquivos/doutorado/teses/2021/Karina_Curcio_Tese_2021_12_10_final.pdf. Acesso em: 25 nov. 2025.

DEITEL, H.; DEITEL, P. **Java: Como Programar**. 10. ed. São Paulo: Pearson Education do Brasil, 2016.

DOCKER. Accelerated Container Application Development. Disponível em: https://www.docker.com/. Acesso em: 30 nov. 2025.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. São Paulo: Pearson Education do Brasil, 2019.

FOWLER, M. **Refatoração: aperfeiçoando o projeto de código existente**. 2. ed. São Paulo: Novatec, 2020.

GIRVAN, L.; PAUL, D. **Agile Analysis and Design: A Practical Guide for Business Analysts**. 2. ed. Sebastopol, EUA: O'Reilly Media, 2021.

GIT. Distributed is the new centralized. Disponível em: https://git-scm.com/. Acesso em: 30 nov. 2025.

GITLAB. A plataforma DevSecOps com tecnologia de IA. Disponível em: https://gitlab.com/gitlab-org/gitlab. Acesso em: 28 nov. 2025.

GLAUBER, N. **Dominando o Android com Kotlin**. São Paulo: Novatec, 2019.

GOTHELF, J.; SEIDEN, J. **Lean UX: Designing Great Products with Agile Teams**. 3. ed. Sebastopol, EUA: O'Reilly Media, 2021.

GRANT, W. **UX Design: Guia Definitivo com as Melhores Práticas de UX**. São Paulo: Novatec, 2019.

HUMBLE, J.; FARLEY, D. **Entrega contínua: como entregar software de forma rápida e confiável**. Porto Alegre: Bookman, 2013.

JANDL JUNIOR, P. **Java: Guia do Programador**. 4. ed. São Paulo: Novatec, 2021.

JAVA. Oracle Java is the #1 programming language and development platform. Disponível em: https://www.java.com/pt-BR/. Acesso em: 28 nov. 2025.

JEMEROV, D.; ISAKOVA, S. **Kotlin em Ação**. São Paulo: Novatec, 2017.

JENKINS. The leading open source automation server. Disponível em: https://www.jenkins.io/. Acesso em: 30 nov. 2025.

JUNIT. The programmer-friendly testing framework for Java and the JVM. Disponível em: https://junit.org/. Acesso em: 27 nov. 2025.

KIM, G.; HUMBLE, J.; DEBOIS, P.; *et al.* **The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations**. 2. ed. Portland, EUA: IT Revolution Press, 2021.

KOTLIN. Kotlin Documentation. 2025. Disponível em: https://kotlinlang.org/docs/home.html. Acesso em: 29 nov. 2025.

KUBERNETES. Orquestração de contêineres prontos para produção. Disponível em: https://kubernetes.io/pt-br/. Acesso em: 30 nov. 2025.

KUPIAINEN, E.; MÄNTYLÄ, M. V.; ITKONEN, J. Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. **Information and Software Technology**, Amsterdam, v. 62, p. 143–163, fev. 2015.

LECHETA, R. R. **Android Essencial com Kotlin**. 2. ed. São Paulo: Novatec, 2018.

LEVY, J. **Estratégia de UX: Técnicas de Estratégia de Produto Para Criar Soluções Digitais Inovadoras**. São Paulo: Novatec, 2021.

LOWDERMILK, T. **Design Centrado no Usuário**. São Paulo: Novatec, 2013.

MACHADO, K. K. **Angular 11 e Firebase: Construindo uma aplicação integrada com a plataforma do Google**. Casa do Código, 2021.

MAJORS, C.; JONES, L. F.; MIRANDA, G. **Observability Engineering: Achieving Production Excellence**. Sebastopol, EUA: O'Reilly Media, 2022.

MARTIN, R. C. **Código Limpo: Habilidades Práticas do Agile Software**. Rio de Janeiro: Alta Books, 2017.

NIELSEN, J.; BUDIU, R. **Usabilidade Móvel**. Rio de Janeiro: GEN LTC, 2013.

NORMAN, D. **O design do dia a dia**. Rio de Janeiro: Rocco, 2024.

NORTH, D. Introducing BDD. Disponível em: https://dannorth.net/blog/introducing-bdd/. Acesso em: 29 nov. 2025.

PLAYWRIGHT. Playwright enables reliable end-to-end testing for modern web apps. Disponível em: https://playwright.dev/. Acesso em: 30 nov. 2025.

PMI. **A Guide to the Project Management Body of Knowledge (PMBOK® Guide)**. 7. ed. Newtown Square, EUA: Project Management Institute, 2021.

POPPENDIECK, M.; POPPENDIECK, T. **Lean Software Development: An Agile Toolkit**. Boston, EUA: Addison-Wesley Professional, 2003.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: uma abordagem profissional**. 9. ed. Porto Alegre: AMG, 2021.

SCHWABER, K.; SUTHERLAND, J. **O guia do Scrum: o guia definitivo para o Scrum: as regras do jogo**. Scrum.org, 2020.

SHARP, H.; PREECE, J.; ROGERS, Y. **Interaction Design: Beyond Human Computer Interaction**. 5. ed. Hoboken, EUA: John Wiley & Sons, 2019.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 7. ed. Rio de Janeiro: GEN LTC, 2020.

SOMMERVILLE, I. **Engenharia de Software**. 10. ed. São Paulo: Pearson Education do Brasil, 2019.

VARGAS, R. **Gerenciamento de Projetos: estabelecendo diferenciais competitivos**. 8. ed. Rio de Janeiro: Brasport, 2017.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de pontos de função: Medição, estimativas e gerenciamento de projetos de software**. 13. ed. São Paulo: Érica, 2013.

VITALINO, J. F. N.; CASTRO, M. A. N. **Descomplicando o Docker**. 2. ed. Rio de Janeiro: Brasport, 2018.

WALTER, M.; TRAMONTINI, R.; FONTANA, R. M.; *et al.* From Sprints to Lean Flow: Management Strategies for Agile Improvement. In: INTERNATIONAL CONFERENCE ON AGILE SOFTWARE DEVELOPMENT, 16., 2015, Helsinki. **Anais [...]** Cham: Springer, 2015. v. 212. p. 310–318.

WANG, R. **Infrastructure as Code, Patterns and Practices**. Shelter Island, EUA: Manning Publications, 2022.

YABLONSKI, J. **Leis da Psicologia Aplicadas a UX**. São Paulo: Novatec, 2020.

ZIJDEMANS, S. H.; STETTINA, C. J. Contracting in Agile Software Projects: State of Art and How to Understand It. In: INTERNATIONAL CONFERENCE ON AGILE SOFTWARE DEVELOPMENT, 15., 2014, Roma. **Anais [...]** Cham: Springer, 2014. v. 179. p. 78–93.