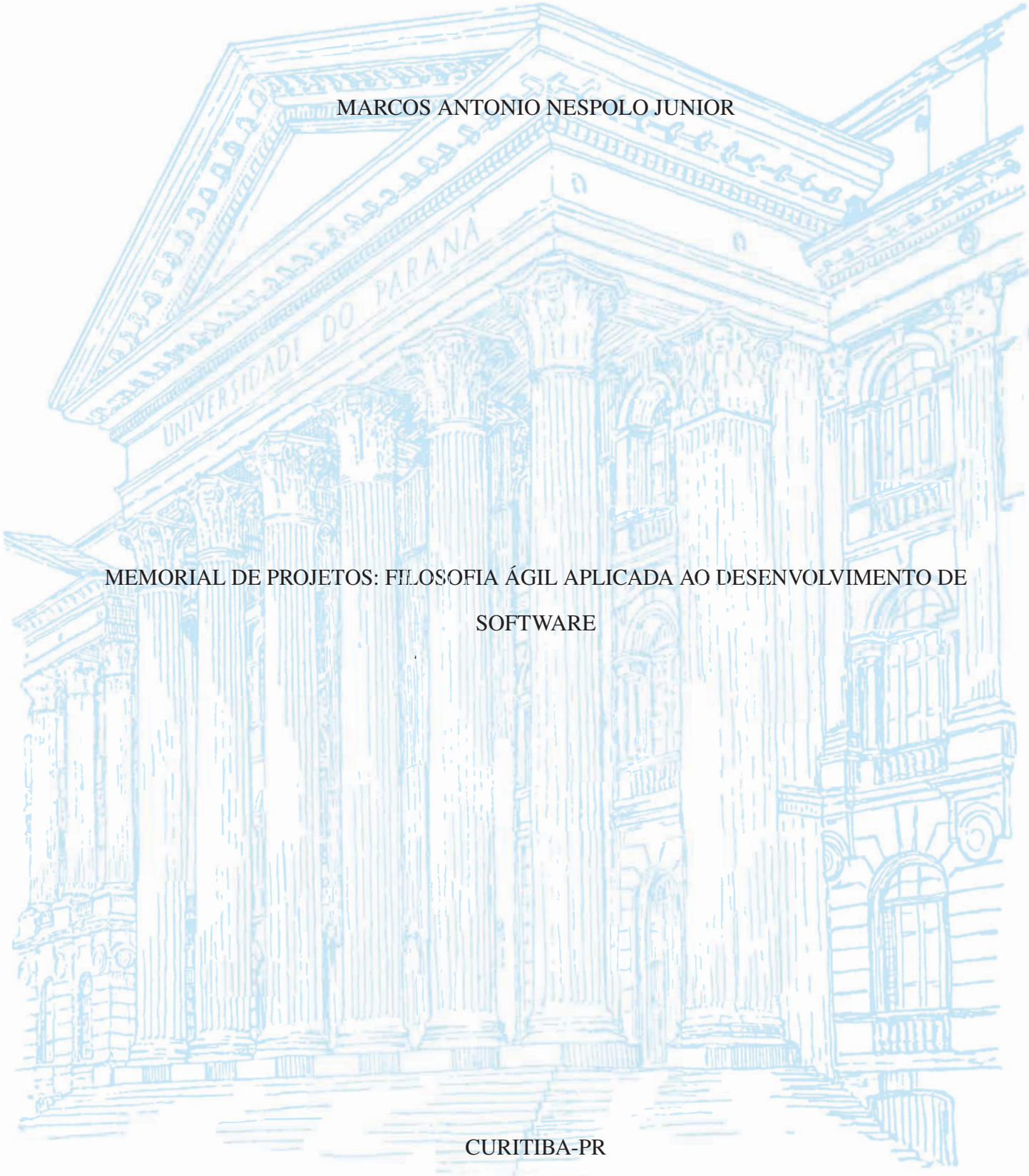
UNIVERSIDADE FEDERAL DO PARANÁ

MARCOS ANTONIO NESPOLO JUNIOR



MEMORIAL DE PROJETOS: FILOSOFIA ÁGIL APLICADA AO DESENVOLVIMENTO DE SOFTWARE

CURITIBA-PR

2025

MARCOS ANTONIO NESPOLO JUNIOR

MEMORIAL DE PROJETOS: FILOSOFIA ÁGIL APLICADA AO DESENVOLVIMENTO 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.

Área de concentração: *Desenvolvimento Ágil de Software*.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño.

CURITIBA-PR

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 **MARCOS ANTONIO NESPOLO JUNIOR**, intitulada: **MEMORIALDEPROJETOS:FILOSOFIAÁGILAPLICADAAODESENVOLVIMENTODE 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, 13 de Novembro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

AGRADECIMENTOS

Agradeço primeiramente à minha família, em especial aos meus avós, Almerinda e Antonio, que, com amor e sacrifício, dedicaram suas vidas para suavizar o peso da minha e abriram caminhos que sozinho eu não conseguiria trilhar.

Aos meus amigos, que foram apoio essencial nesta caminhada, agradeço pela companhia, pelo incentivo e pelos momentos de descontração que me ajudaram a manter o equilíbrio enquanto conciliava trabalho e duas pós-graduações ao mesmo tempo.

Por fim, agradeço à vida, ao destino — ou a Deus — por ter colocado no meu caminho os desafios que me lapidaram, as oportunidades que me abriram horizontes e as conquistas que deram sentido à jornada, permitindo-me concluir mais esta etapa da minha trajetória.

“Viver é se adaptar ao que vem. É fazer da mudança um modo de ser.”
Clóvis de Barros Filho

RESUMO

O conceito de Desenvolvimento Ágil de Software consolidou-se como uma resposta às limitações dos modelos tradicionais de engenharia, priorizando adaptação, colaboração e entrega contínua de valor. A partir das disciplinas do curso de pós-graduação, foram exploradas diferentes dimensões do ágil, incluindo fundamentos conceituais, práticas de modelagem, gestão de projetos, desenvolvimento *web* e *mobile*, *UX* e *DevOps*. Essas abordagens evidenciaram como a filosofia ágil se manifesta tanto no nível técnico, por meio de integração contínua, testes automatizados e ciclos iterativos, quanto no organizacional, com ênfase em equipes auto-organizadas e *feedback* constante. Além dos aspectos metodológicos, discutiu-se a relevância da postura ágil como orientação ética e profissional diante de cenários de mudança, incerteza e complexidade. Em síntese, o memorial destaca que o ágil não é apenas um conjunto de práticas, mas uma filosofia que integra pessoas, processos e tecnologia para formar profissionais preparados a entregar soluções sustentáveis e de impacto real. Os projetos de MADS e MAG consolidaram modelagem essencial, casos de uso, histórias de usuário, critérios claros, diagramas e classes. GAP estruturou *releases*, *sprints*, velocidade, fluxo e métricas. *Web* e *Mobile* transformaram requisitos em interfaces e serviços; *UX* promoveu pesquisa, protótipos e usabilidade. Infra e *DevOps* automatizaram *pipelines*, versionamento, integração e entrega. Testes implementaram verificação automatizada, cobertura e regressão. Integração demonstrou aprendizado colaborativo, comunicação efetiva e sustentabilidade.

Palavras-chave: desenvolvimento ágil de software; filosofia ágil; métodos ágeis; entrega de valor; integração contínua.

ABSTRACT

The concept of Agile Software Development has consolidated itself as a response to the limitations of traditional engineering models, prioritizing adaptation, collaboration, and continuous delivery of value. Based on the disciplines of the postgraduate course, different dimensions of agility were explored, including conceptual foundations, modeling practices, project management, web and mobile development, UX, and DevOps. These approaches showed how the agile philosophy manifests itself both at the technical level—through continuous integration, automated testing, and iterative cycles—and at the organizational level, with an emphasis on self-organizing teams and constant feedback. Beyond methodological aspects, the relevance of the agile stance was discussed as an ethical and professional orientation in the face of change, uncertainty, and complexity. In summary, the memorial highlights that agility is not just a set of practices, but a philosophy that integrates people, processes, and technology to prepare professionals to deliver sustainable solutions with real impact. The MADS and MAG projects consolidated essential modeling, use cases, user stories, clear criteria, diagrams, and classes. GAP structured releases, sprints, velocity, flow, and metrics. Web and Mobile transformed requirements into interfaces and services; UX promoted research, prototypes, and usability. Infra and DevOps automated pipelines, versioning, integration, and delivery. Tests implemented automated verification, coverage, and regression. Integration demonstrated collaborative learning, effective communication, and sustainability.

Keywords: agile software development; agile philosophy; agile methods; value delivery; continuous integration.

LISTA DE FIGURAS

SUMÁRIO

1 PARECER TÉCNICO

O Desenvolvimento Ágil de Software representa uma mudança de paradigma na forma de conceber, construir e evoluir sistemas. Mais do que um conjunto de métodos, ele propõe uma cultura de colaboração e adaptação contínua, em que o valor entregue ao usuário orienta cada decisão técnica. Ao longo do curso, essa abordagem foi explorada de modo integrado, unindo fundamentos conceituais, práticas de gestão e experimentação em projetos reais.

1.1 DESENVOLVIMENTO ÁGIL DE SOFTWARE

O termo Desenvolvimento Ágil de Software refere-se a um conjunto de práticas, valores e princípios que surgiram no início dos anos 2000 como resposta às limitações dos métodos tradicionais de desenvolvimento, conhecidos como modelos preditivos ou em cascata. Sua consolidação ocorreu com a publicação do Manifesto Ágil (Beck *et al.*, 2001), que estabeleceu como prioridade a entrega contínua de valor ao cliente por meio da colaboração, da adaptação a mudanças e da simplicidade no processo de trabalho.

Diferentemente das abordagens tradicionais, que enfatizam planejamento extensivo e documentação detalhada, o ágil privilegia ciclos curtos de desenvolvimento, nos quais requisitos, *design*, implementação e validação ocorrem de forma incremental. Esse modelo iterativo busca reduzir riscos, aumentar a qualidade do produto e permitir que as equipes respondam de maneira eficaz a contextos dinâmicos.

Segundo Highsmith (2002), o ágil deve ser entendido não apenas como um método de gestão de projetos, mas como um ambiente de trabalho que promove flexibilidade e confiança entre os participantes. Essa perspectiva reforça o caráter filosófico do movimento, que transcende práticas isoladas e orienta a cultura de equipes de *software*.

Ao longo de sua evolução, o Desenvolvimento Ágil incorporou influências de diferentes correntes, como a manufatura enxuta, a engenharia de *software* incremental e a programação extrema. Essas influências ajudaram a formar um paradigma no qual a entrega de valor é contínua, o aprendizado é constante e a adaptação é considerada parte natural do ciclo de desenvolvimento.

Em síntese, o Desenvolvimento Ágil de Software pode ser definido como uma abordagem que busca alinhar pessoas, processos e tecnologia em um fluxo de trabalho iterativo e colaborativo, priorizando a adaptação a mudanças e a entrega de software funcional em curtos intervalos

de tempo. Trata-se de um conceito que combina filosofia, práticas e princípios técnicos para enfrentar a complexidade e a incerteza características do desenvolvimento moderno de sistemas.

1.2 FILOSOFIA ÁGIL APLICADA AO DESENVOLVIMENTO DE SOFTWARE

Ao longo do curso de pós-graduação em Desenvolvimento Ágil de Software, foram desenvolvidos projetos que permitiram integrar conceitos desde a análise de requisitos até a implementação, testes e implantação contínua. Esses trabalhos mostraram como os conteúdos das disciplinas se conectam para compor um processo de desenvolvimento robusto e iterativo, em sintonia com os valores do Manifesto Ágil (Beck *et al.*, 2001).

Dentro desse panorama, falar em filosofia ágil significa reconhecer que agilidade é, antes de tudo, postura: ela coloca colaboração, simplicidade e adaptação contínua no centro do trabalho diário. Highsmith (2002, p. 123) lembra que agilidade é "mais atitude do que processo, mais ambiente do que metodologia" , enquanto Martin (2008) aponta que a dedicação permanente à excelência técnica torna possível mudar sem medo. Shore e Warden (2007) reforçam essa ideia ao mostrar que equipes que aprendem juntas e se auto-organizam avançam de maneira sustentável justamente porque adotam esse modo de trabalhar. Tomados em conjunto, esses autores explicam por que falamos em filosofia: trata-se de um conjunto de valores que atravessa o planejamento, a construção e a evolução do *software*.

A jornada formativa teve início com Introdução à Programação e Banco de Dados, quando se consolidaram fundamentos de lógica, estruturas de dados e modelagem relacional. Essa base permitiu, nas iterações subsequentes, produzir código funcional em ciclos curtos e manipular informações com segurança, premissas que, como mostra Larman (2004), reduzem riscos porque expõem o sistema a testes reais desde cedo.

Na sequência, Métodos Ágeis para Desenvolvimento de Software e Modelagem Ágil aprofundaram valores como iterações curtas, entregas incrementais e *feedback* constante. A modelagem adotou a lógica *just-enough*: diagramas UML serviram para comunicar, não burocratizar, em consonância com Booch, Rumbaugh e Jacobson (2005), que defendem integrar modelagem e programação num fluxo evolutivo .

O Gerenciamento Ágil de Projetos consolidou práticas como *backlog* priorizado, quadros Kanban e retrospectivas. Poppendieck e Poppendieck (2003) lembram que visualizar o fluxo revela gargalos e elimina desperdícios, acelerando a entrega de valor. Nos projetos, tais artefatos

permitiram revisar prioridades a cada ciclo e ajustar escopo conforme o retorno dos usuários, reais ou simulados.

As disciplinas de Desenvolvimento Web e Desenvolvimento Mobile traduziram os modelos em aplicações concretas, enquanto UX no Desenvolvimento Ágil trouxe prototipagem rápida e testes de usabilidade, garantindo que cada incremento responda às necessidades do usuário.

Na dimensão de infraestrutura, a disciplina de DevOps destacou como *pipelines* de integração e entrega contínuas mantêm o *software* em um estado sempre funcional e potencialmente liberável. Essa visão dialoga com Fowler (2006), que defende que a automação de *build*, testes e *deploys* não apenas reduz o risco de falhas de integração, mas também garante *feedback* rápido a cada modificação, permitindo que a equipe trabalhe com segurança em ciclos curtos. Em paralelo, a disciplina de Testes Automatizados reforçou esse princípio ao assegurar qualidade consistente a cada compilação, sustentando a ideia de que a excelência técnica é o motor da agilidade (Martin, 2008).

Desse modo, análise, modelagem, gestão, programação e operação se entrelaçam num ciclo único, confirmando que o ágil não é um receituário, mas uma filosofia que orienta escolhas técnicas e comportamentais. Ciclos iterativos facilitam o ajuste permanente do produto ao contexto, como descreve Larman (2004), e o acompanhamento dos requisitos ao longo de todo o projeto assegura a relevância do sistema, conforme enfatiza Sommerville (2011).

O memorial “Filosofia ágil aplicada ao desenvolvimento de software” evidencia a passagem da teoria para a prática em um fluxo contínuo e adaptativo que atravessa todas as disciplinas do curso. Mais do que o domínio de técnicas, essa trajetória ressalta a colaboração, a adaptabilidade e a entrega contínua de valor como fundamentos essenciais do desenvolvimento ágil. Também revela os desafios de aplicar tais princípios em contextos reais, marcados por incertezas, restrições e mudanças constantes. Dessa forma, a filosofia ágil se afirma não apenas como um conjunto de práticas, mas como uma postura profissional capaz de orientar escolhas éticas, técnicas e organizacionais. Em última análise, trata-se de formar profissionais preparados para atuar em cenários de transformação permanente, contribuindo para soluções sustentáveis e de impacto positivo na sociedade.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

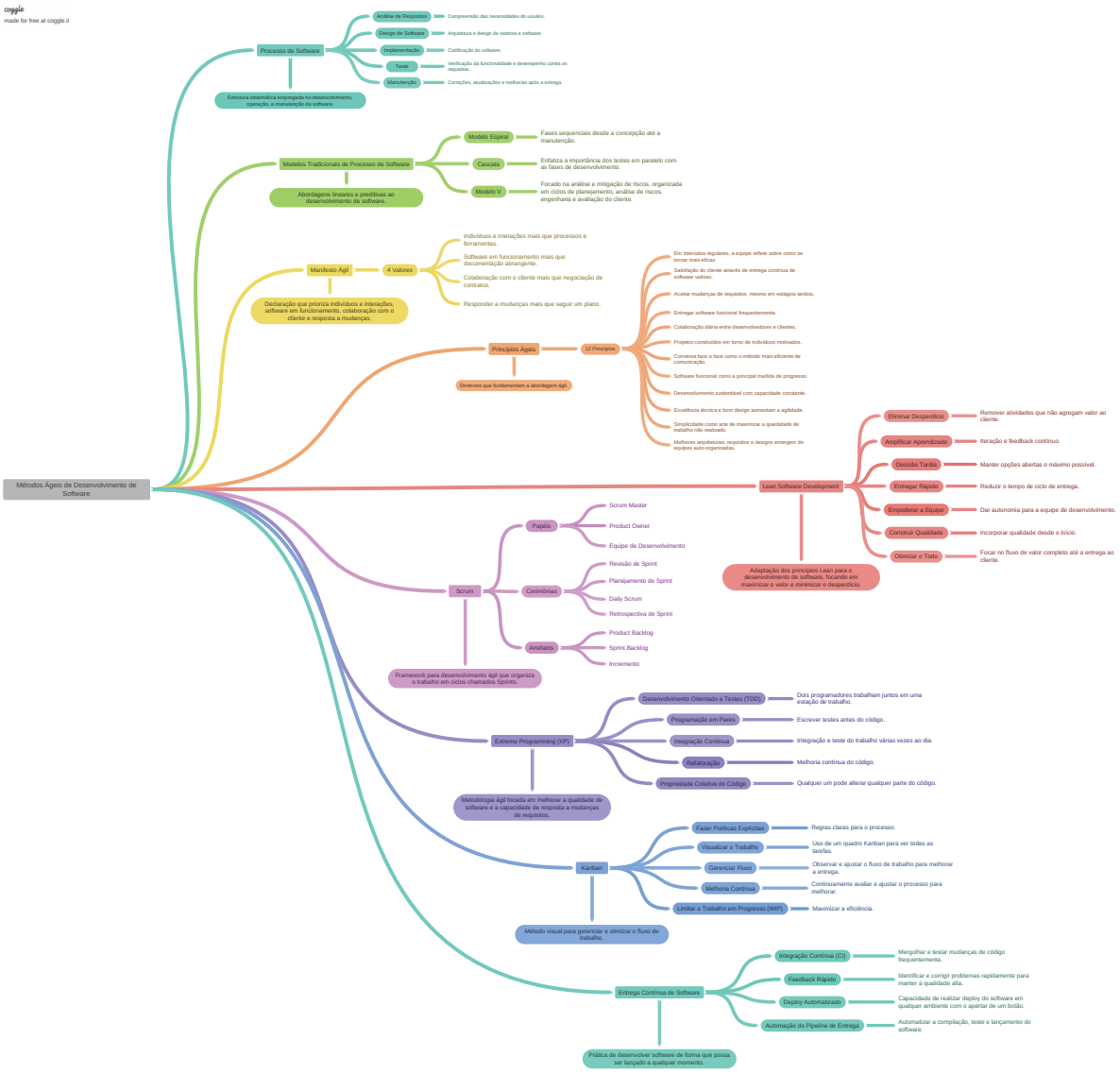
Na disciplina de Métodos Ágeis de Desenvolvimento de Software, o projeto teve como objetivo elaborar um mapa mental que reunisse e organizasse os principais conceitos ligados ao processo de software e às abordagens ágeis. O artefato deveria representar, em uma única estrutura visual, desde os modelos tradicionais até o surgimento do Manifesto Ágil e de seus princípios, incluindo tópicos como *Lean Software Development* (Poppendieck e Poppendieck, 2003), *Scrum* (Schwaber, 2004), *Extreme Programming* (Beck e Andres, 2004), e *Kanban* (Anderson, 2010), além da prática de entrega contínua. Cada tópico precisava conter definição e detalhamento, de modo a deixar claras não apenas as denominações, mas também atividades, papéis e forma de trabalho associados.

A importância desse projeto para o desenvolvimento ágil de software está em oferecer uma visão histórica e estruturada: primeiro se observa como o processo de software foi pensado de forma preditiva e sequencial, depois se compreende por que o ágil surgiu para lidar com mudanças, entrega de valor e colaboração. Ter esse panorama facilita, nas disciplinas seguintes, a escrita de histórias de usuário, a priorização de *backlogs* e a participação em *sprints* com clareza sobre o propósito das práticas adotadas.

Além disso, esse trabalho se integra às demais disciplinas do curso porque serve como base conceitual para todas elas. As ideias de iteração curta, *feedback* e entrega incremental aparecem em gerenciamento ágil de projetos, em modelagem ágil, em desenvolvimento *web* e *mobile*, em testes automatizados e também em *DevOps*, que fornece o suporte técnico para entregar continuamente. Assim, o mapa mental funciona como referência inicial: sempre que uma disciplina introduz uma nova prática, é possível posicioná-la dentro do conjunto maior de métodos e princípios ágeis.

2.1 ARTEFATOS DO PROJETO

Figura 2.1 – RESOLUÇÃO EXERCÍCIO: MAPA MENTAL - MADS



Fonte: O autor (2025)

3 DISCIPLINA: MAG 1 E MAG 2 - MODELAGEM ÁGIL DE SOFTWARE 1 E 2

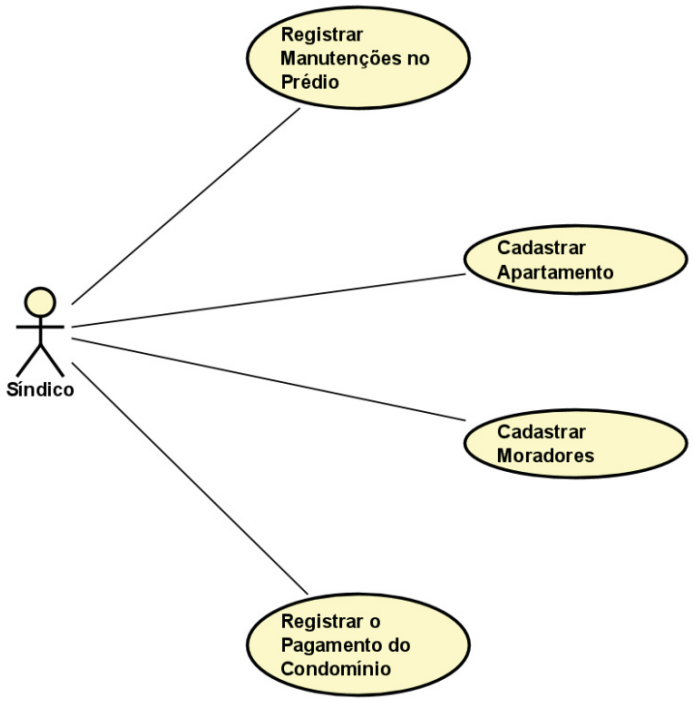
Nas disciplinas de Modelagem Ágil de Software 1 e 2, o projeto teve como propósito modelar um Sistema de Gestão de Condomínio a partir de uma especificação realista. O trabalho exigiu a elaboração do diagrama de casos de uso em dois níveis e o conjunto completo de histórias de usuário, contemplando funções típicas de um condomínio, como cadastro de apartamentos, cadastro de moradores com validação de CPF e vínculo ao apartamento, registro de pagamentos condicionados à inexistência de débitos anteriores e registro de manutenções periódicas.

A importância desse projeto para o desenvolvimento ágil está em evidenciar que agilidade não significa ausência de modelagem, mas sim modelagem essencial, orientada a valor e fácil de atualizar. Ao transformar os requisitos do síndico em histórias de usuário, o sistema passa a ser descrito no formato de quem usa, o que facilita a priorização em *backlogs* e a negociação com o cliente. Já os diagramas de caso de uso, em níveis, permitem visualizar rapidamente o escopo e decompor o sistema em incrementos entregáveis, algo indispensável quando o domínio possui muitas regras de consistência, como cadastros e cobranças (Booch, Rumbaugh e Jacobson, 2005).

Além disso, esse projeto se integra às demais disciplinas do curso porque fornece a base de requisitos sobre a qual as outras atividades podem trabalhar. As histórias de usuário geradas aqui podem ser reutilizadas em gerenciamento ágil de projetos para montar *sprints*, em desenvolvimento *web* e *mobile* para implementar telas e serviços, e em testes e *DevOps* para derivar casos de teste automatizados e cenários de integração contínua.

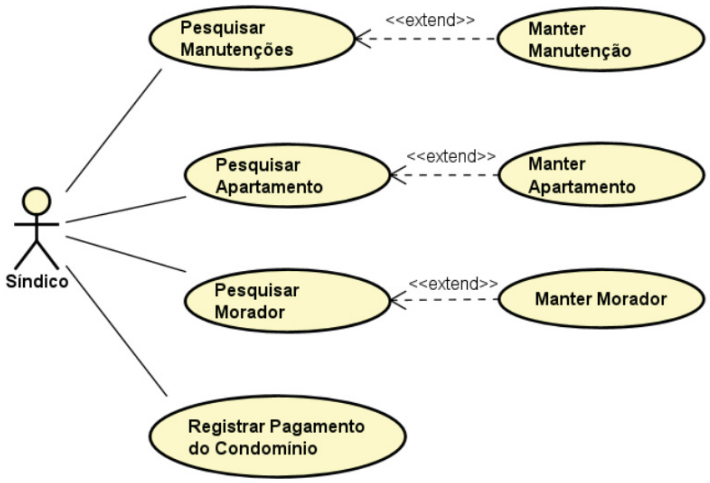
3.1 ARTEFATOS DO PROJETO

Figura 3.1 – DIAGRAMA DE CASO DE USO NÍVEL 1



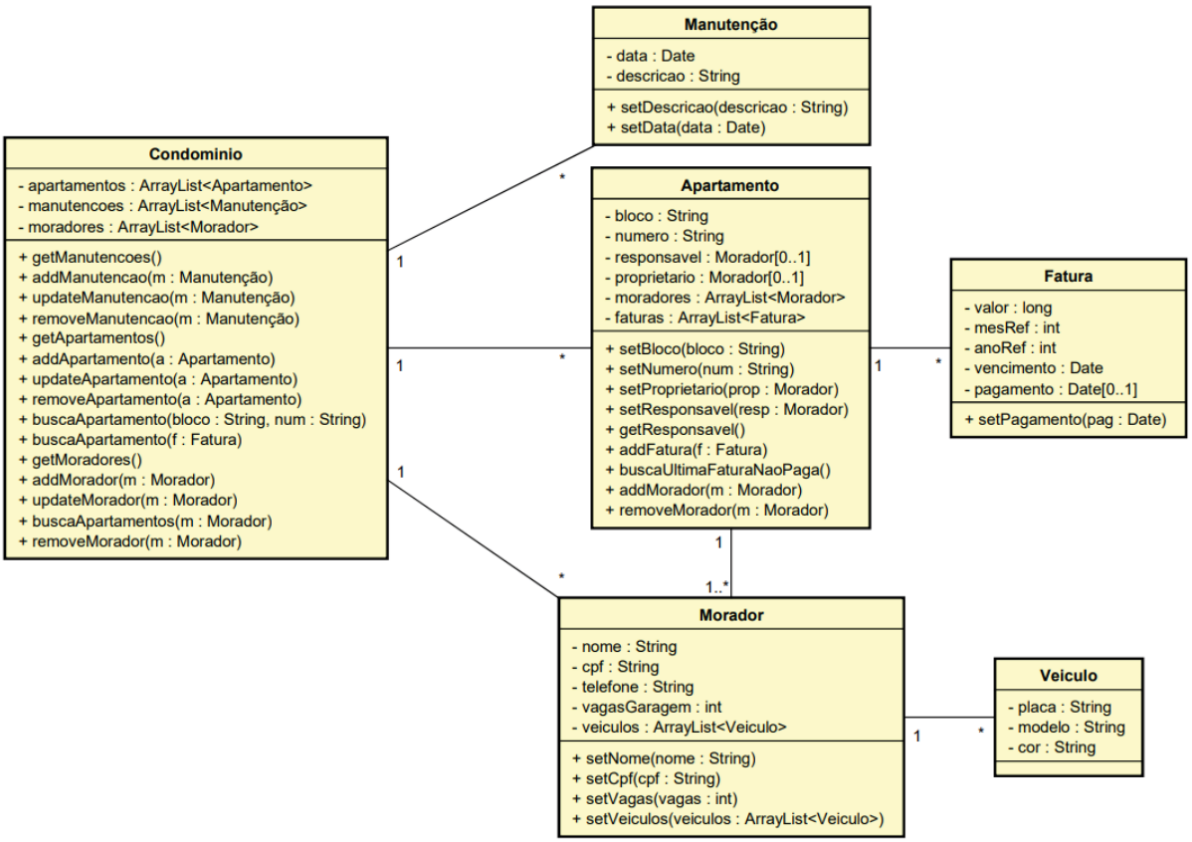
Fonte: O autor (2025)

Figura 3.2 – DIAGRAMA DE CASO DE USO NÍVEL 2



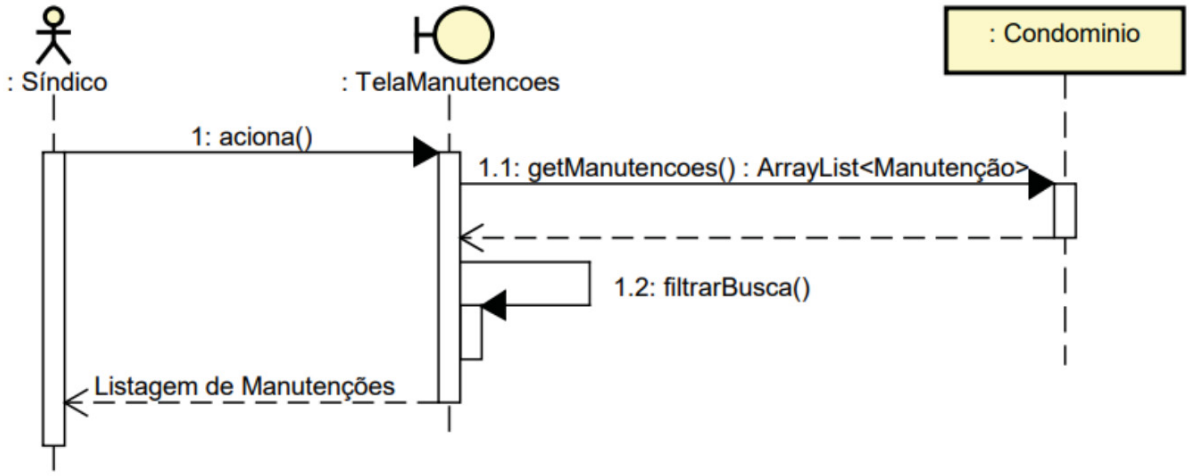
Fonte: O autor (2025)

Figura 3.3 – DIAGRAMA DE CLASSES



Fonte: O autor (2025)

Figura 3.4 – DIAGRAMA DE SEQUÊNCIA HU001 - PESQUISAR MANUTENÇÕES



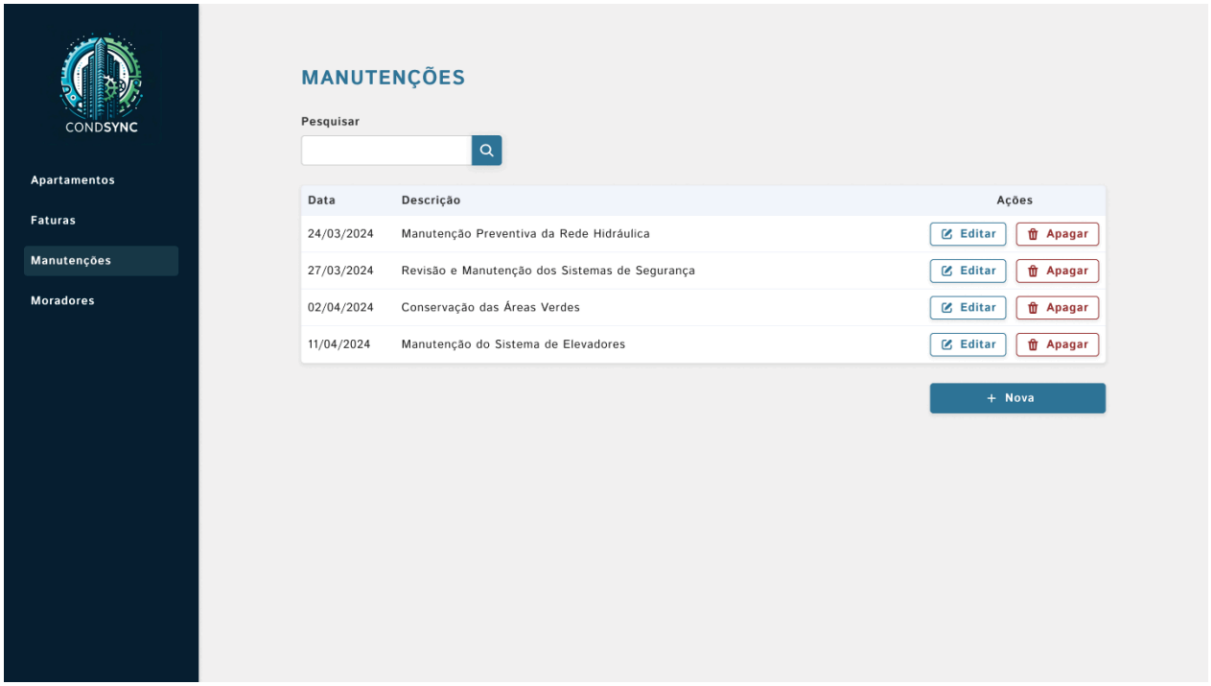
Fonte: O autor (2025)

HISTÓRIAS DE USUÁRIO

HU001 - Pesquisar Manutenções

SENDO	O Síndico
QUERO	Pesquisar as manutenções
PARA	Fazer manutenções nos seus dados

Figura 3.5 – TELA HU001 - PESQUISAR MANUTENÇÕES



Fonte: O autor (2025)

CRITÉRIOS DE ACEITAÇÃO:

1. Deve permitir ao Síndico acessar a funcionalidade de pesquisa de manutenções.
2. Deve exibir uma interface de pesquisa que permita ao Síndico inserir critérios de busca, como tipo de manutenção, data ou descrição.
3. Deve fornecer resultados relevantes de acordo com os critérios de busca inseridos pelo Síndico.
4. Deve apresentar as informações das manutenções encontradas, incluindo tipo, data, descrição e qualquer outra informação relevante.

5. Deve permitir ao Síndico selecionar uma manutenção específica para visualização detalhada.

CRITÉRIOS DE ACEITAÇÃO - DETALHAMENTO:

1. Deve permitir ao Síndico acessar a funcionalidade de pesquisa de manutenções:

Dado que o Síndico está autenticado no sistema

Quando ele acessa a seção de manutenção

Então ele deve ter a opção de iniciar uma pesquisa de manutenções.

2. Deve exibir uma interface de pesquisa que permita ao Síndico inserir critérios de busca:

Dado que o Síndico está na página de pesquisa de manutenções

Quando ele visualiza a interface de pesquisa

Então deve ser apresentado um formulário com campos para inserir critérios de busca, como tipo de manutenção, data e descrição.

3. Deve fornecer resultados relevantes de acordo com os critérios de busca inseridos pelo Síndico:

Dado que o Síndico preencheu os critérios de busca e

Quando o sindico pressionar o botão de pesquisa

Então devem ser exibidos os resultados relevantes de acordo com os critérios inseridos.

4. Deve apresentar as informações das manutenções encontradas:

Dado que o Síndico visualiza os resultados da pesquisa de manutenções

Quando o sistema apresenta os resultados

Então o sistema deve exibir as informações das manutenções encontradas, incluindo tipo, data, descrição e outras informações pertinentes.

5. Deve permitir ao Síndico selecionar uma manutenção específica para visualização detalhada:

Dado que o Síndico visualiza os resultados da pesquisa de manutenções

Quando ele seleciona uma manutenção específica na lista de resultados

Então o sistema deve permitir a visualização detalhada dessa manutenção.

4 DISCIPLINA: GAP 1 E GAP 2 - GERENCIAMENTO ÁGIL DE PROJETOS DE SOFTWARE 1 E 2

Na disciplina de Gerenciamento Ágil de Projetos de Software 1, o projeto consistiu na elaboração de um plano de *release* para um software definido na própria disciplina, considerando o desenvolvimento realizado por apenas uma pessoa (conforme apresentado na Figura 4.1). O trabalho exigiu o cálculo explícito da velocidade, adotando que 1 ponto equivale a 8 horas de trabalho, e a organização de todas as *sprints* com datas de início e fim previamente planejadas. Cada *sprint* deveria conter histórias de usuário escritas no formato SENDO/QUERO/PARA, com suas estimativas registradas, reforçando a prática de especificar requisitos orientados a valor e de planejar entregas em pequenos incrementos. Esse exercício é importante para o desenvolvimento ágil porque conecta esforço, tempo e escopo de forma transparente, permitindo negociar prazos sem perder o controle do que será entregue.

Na disciplina de Gerenciamento Ágil de Projetos de Software 2, o foco passou da previsão para a operação do fluxo, por meio da simulação de um ciclo de 35 dias na plataforma Kanban Board Game (Kanban, 2025). Nessa atividade foram aplicados conceitos como limite de trabalho em progresso, acompanhamento de fluxo e visualização do trabalho, registrando evidências em diferentes momentos da execução, inclusive o *Cumulative Flow Diagram* (*CFD*) (Anderson, 2010).

A integração entre GAP 1 e GAP 2 e as demais disciplinas do curso ocorre porque os artefatos produzidos aqui podem se apoiar nas histórias modeladas em *MAG 1 e 2*, podem ser implementados em desenvolvimento *web* e *mobile* e podem servir de base para testes e *DevOps*, que dependem de entregas pequenas e frequentes para automatizar *build*, testes e *deploy*. Forma-se, assim, o ciclo completo: primeiro modela-se o sistema, depois transformam-se os requisitos em histórias priorizadas, em seguida planejam-se *releases* com *sprints* e, por fim, monitora-se o fluxo em *kanban*, garantindo que o que foi planejado realmente atravessa o sistema sem gargalos.

4.1 ARTEFATOS DO PROJETO

Figura 4.1 – PLANO DE RELEASE - SISTEMA DE GESTÃO DE CONDOMÍNIO

Gerenciamento Ágil de Projetos I
Profa. Dra. Rafaela Mantovani Fontana
Plano de Release - Sistema de Gestão de Condomínio
Marcos Antonio Nespolo Junior

Cálculo da Velocidade:

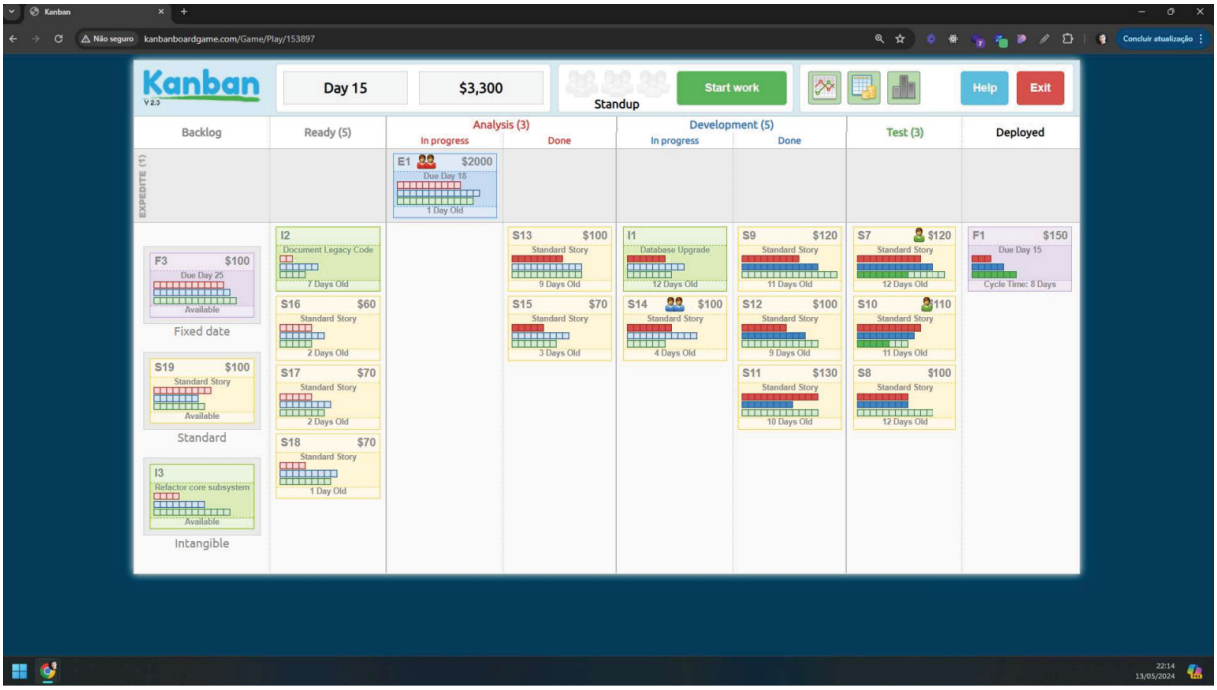
Horas disponíveis por dia:	8 horas	Tamanho da Sprint:	2 semanas
Horas disponíveis por Sprint:	80 horas	Velocidade:	10 pontos

Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4
Data Início: 06/05/2024	Data Início: 20/05/2024	Data Início: 03/06/2024	Data Início: 17/06/2024
Data Fim: 17/05/2024	Data Fim: 31/05/2024	Data Fim: 14/06/2024	Data Fim: 28/06/2024
<HU001 – Pesquisar Manutenções> SENDO O Síndico QUERO Pesquisar as manutenções PARA Fazer manutenções nos seus dados ESTIMATIVA (3)	<HU003 – Pesquisar Apartamento> SENDO O Síndico QUERO Pesquisar Apartamento PARA Fazer manutenções nos seus dados ESTIMATIVA (3)	<HU005 – Registrar Pagamento do Condomínio> SENDO O Síndico QUERO Registrar Pagamento do Condomínio PARA Que seus dados fiquem atualizados ESTIMATIVA (8)	<HU006 – Pesquisar Morador> SENDO O Síndico QUERO Pesquisar Morador PARA Fazer manutenções nos seus dados ESTIMATIVA (5)
<HU002 – Manter Manutenção> SENDO O Síndico QUERO Manter Manutenção PARA Que seus dados fiquem atualizados ESTIMATIVA (5)	<HU004 – Manter Apartamento> SENDO O Síndico QUERO Manter Apartamento PARA Que seus dados fiquem atualizados ESTIMATIVA (5)		<HU007 – Manter Morador> SENDO O Síndico QUERO Manter os dados do Morador PARA Que seus dados fiquem atualizados ESTIMATIVA (5)

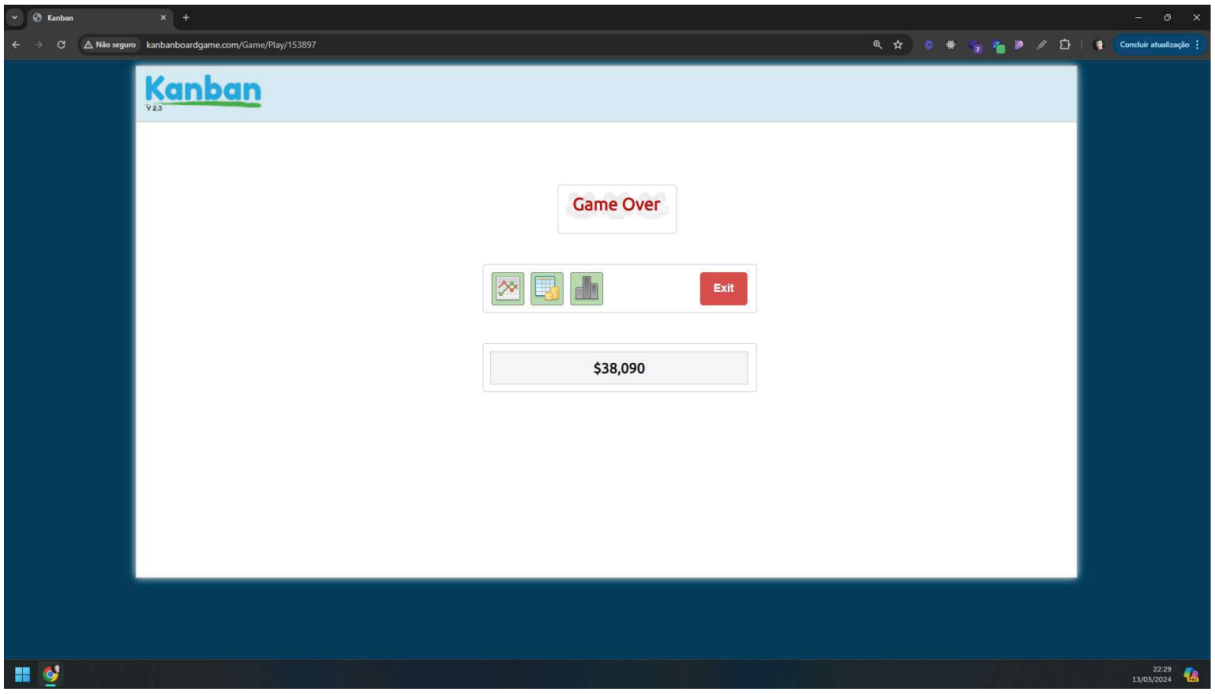
Fonte: O autor (2025)

Figura 4.2 – EVIDÊNCIA DO DIA 15 DO JOGO



Fonte: O autor (2025)

Figura 4.3 – EVIDÊNCIA DO TOTAL DA RECEITA OBTIDA



Fonte: O autor (2025)

Figura 4.4 – DIAGRAMA DE FLUXO ACUMULADO

Fonte: O autor (2025)

5 DISCIPLINA: INTRO - INTRODUÇÃO À PROGRAMAÇÃO

Na disciplina de Introdução à Programação, o projeto teve como objetivo implementar o *backend* de um sistema bancário simplificado, responsável por cadastrar clientes, contas correntes e contas de investimento, utilizando o diagrama de classes fornecido, os testes unitários prontos em *JUnit* (JUnit, 2025) e o *script DDL (Data Definition Language)*[1] (Elmasri e Navathe, 2016) para criação das tabelas no *MySQL* (Oracle, 2025). Mesmo sendo uma disciplina introdutória, o trabalho já foi proposto no formato de desenvolvimento orientado a testes: a atividade só é considerada concluída quando os testes ficam verdes, isto é, quando o código realmente faz o que o requisito descreve. Ao exigir que ao menos 40 dos 42 testes passem simultaneamente, a disciplina deixa claro que qualidade não é opcional e que pequenas quebras de regra de negócio precisam ser tratadas diretamente no código.

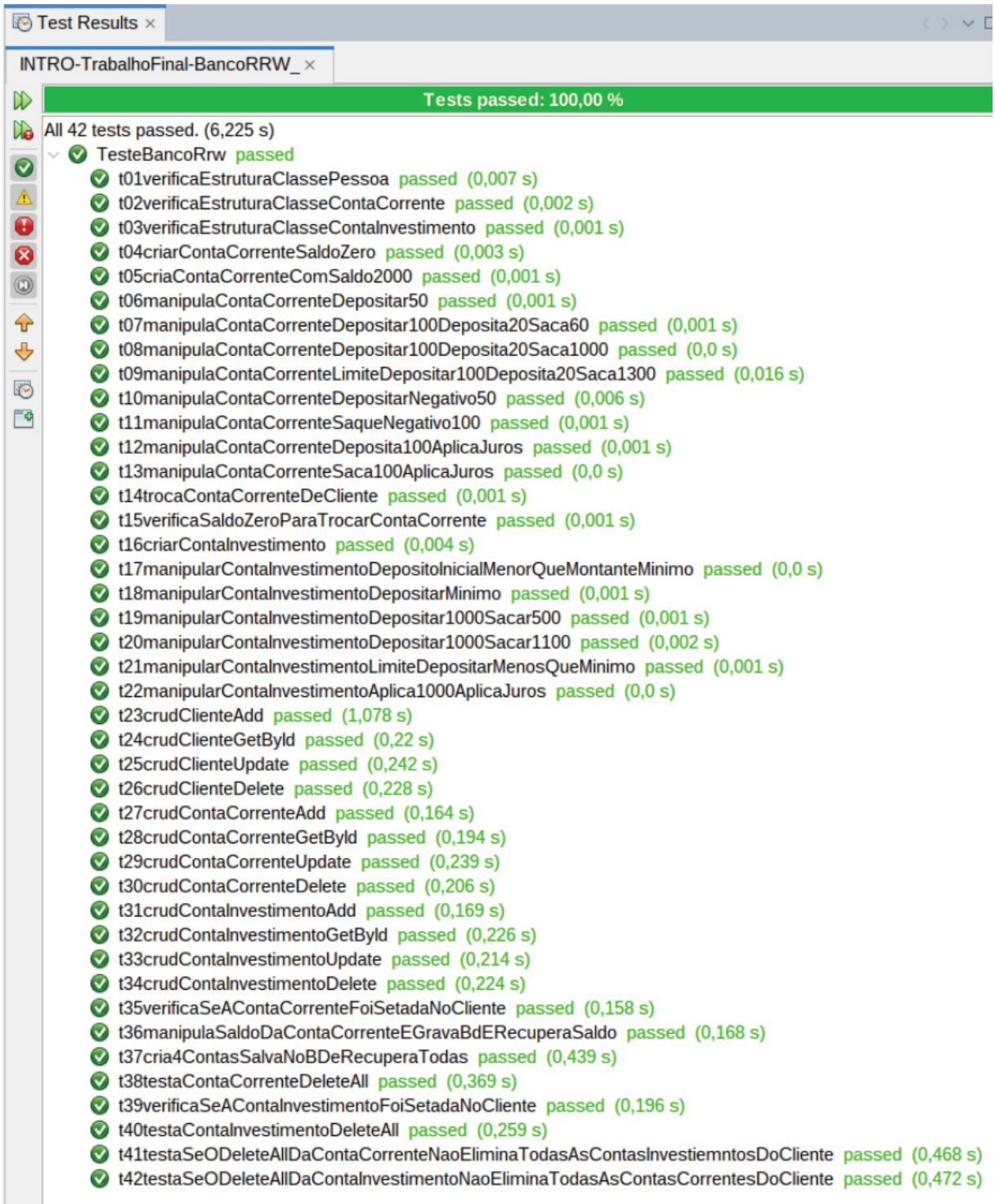
A importância desse projeto para o desenvolvimento ágil de software está em introduzir a ideia de que software ágil depende de código testável e evolutivo. Em métodos ágeis, cada incremento precisa estar funcional para poder ser entregue em uma *sprint* ou em um fluxo de *kanban*; ter uma suíte de testes automatizados, mesmo simples, permite refatorar, corrigir e acrescentar funcionalidades sem medo de quebrar o que já estava pronto.

O uso de um esquema relacional real, com criação de tabelas por *DDL*, validação de chaves e respeito às restrições, antecipa o que depois será detalhado na disciplina de Banco de Dados. Já a relação com Testes Automatizados é direta, porque o projeto já nasce com uma suíte de *JUnit* e com a lógica de desenvolver guiado pelos testes, aproximando-se de *TDD*: primeiro o comportamento esperado é conhecido, depois o código é escrito até que todos os testes fiquem verdes.

[1] Data Definition Language: conjunto de comandos SQL voltados à definição de estrutura do banco, como CREATE, ALTER e DROP.

5.1 ARTEFATOS DO PROJETO

Figura 5.1 – EVIDÊNCIA DOS TESTES



Fonte: O autor (2025)

6 DISCIPLINA: BD - BANCO DE DADOS

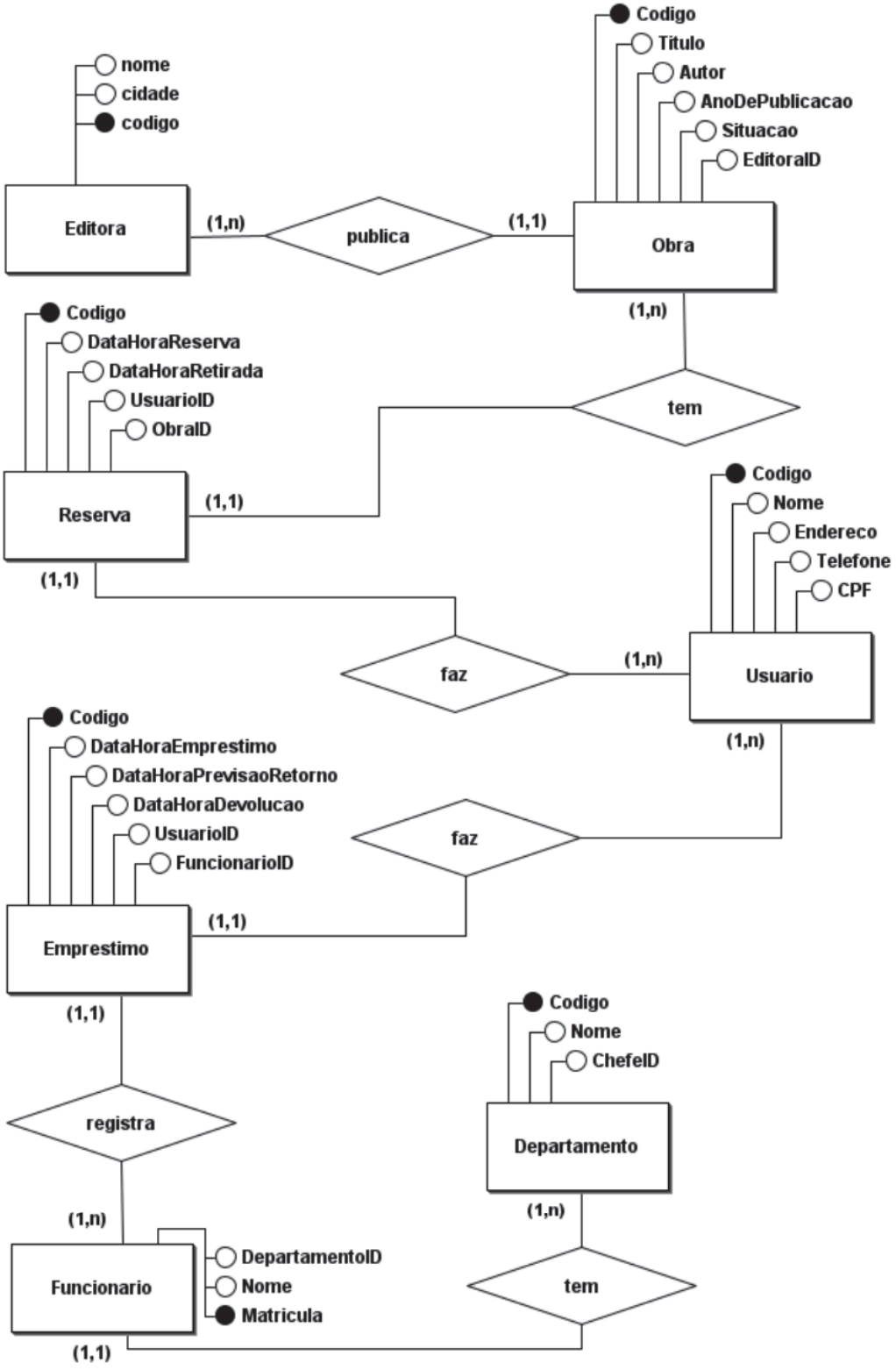
Na disciplina de Banco de Dados, foi necessário modelar um sistema de controle de biblioteca a partir de requisitos textuais, produzindo primeiro um modelo entidade-relacionamento conceitual e, em seguida, o modelo lógico com tabelas, chaves primárias, chaves estrangeiras e relacionamentos (Elmasri e Navathe, 2016).

Na segunda parte, a disciplina aproximou o exercício de um cenário de desenvolvimento real ao pedir a escolha de um domínio próprio e sua transformação em modelo lógico e *script SQL* completo, com exemplos de dados.

Esse trabalho se integra às demais disciplinas porque os modelos produzidos aqui podem ser reutilizados em modelagem ágil (*MAG 1 e 2*), em gerenciamento ágil de projetos (*GAP 1 e 2*) para estimar *backlogs* com mais realismo, e nas disciplinas de desenvolvimento *web* e *mobile*, que consomem exatamente essas tabelas.

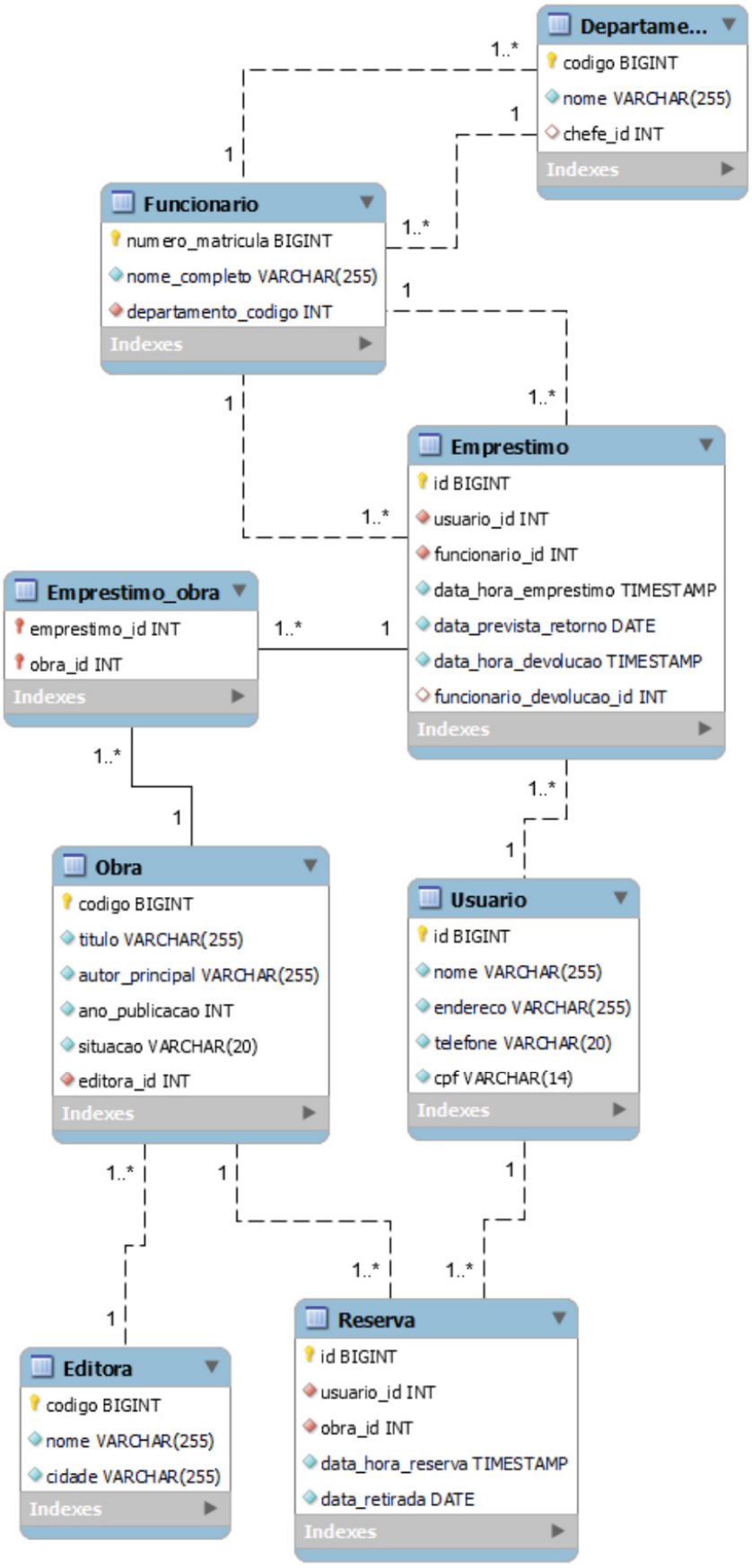
6.1 ARTEFATOS DO PROJETO

Figura 6.1 – MODELO ENTIDADE-RELACIONAMENTO CONCEITUAL



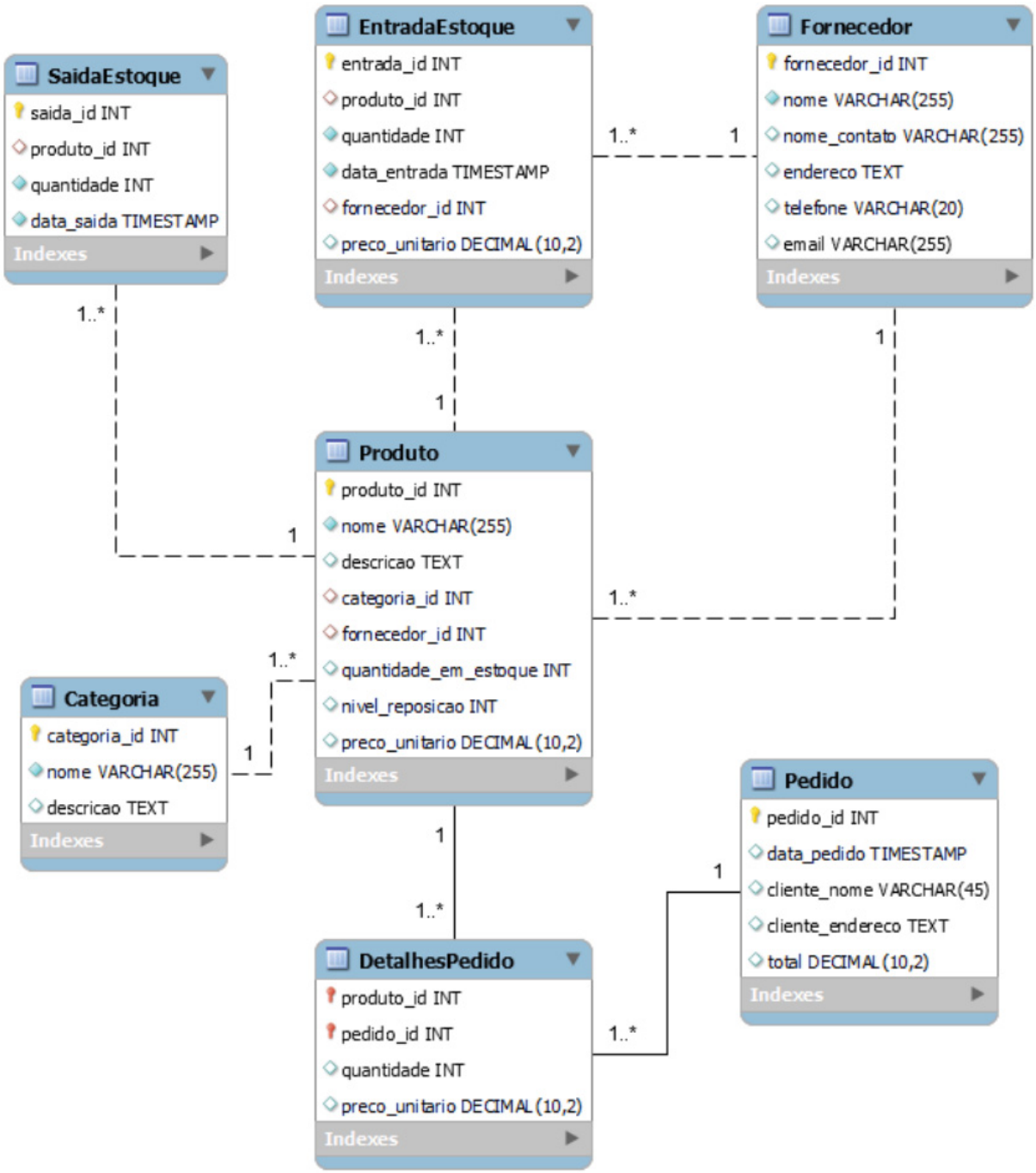
Fonte: O autor (2025)

Figura 6.2 – MODELO LÓGICO - DIAGRAMA DE ENTIDADE RELACIONAMENTO



Fonte: O autor (2025)

Figura 6.3 – MODELO LÓGICO



Fonte: O autor (2025)

Criando as tabelas

```
1 CREATE TABLE Categoria (
2     `categoria_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
3     `nome` VARCHAR(255) NOT NULL,
4     `descricao` TEXT NULL DEFAULT NULL,
5 PRIMARY KEY (`categoria_id`));
```

```
6
7  CREATE TABLE Produto (
8      `produto_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
9      `nome` VARCHAR(255) NOT NULL,
10     `descricao` TEXT NULL DEFAULT NULL,
11     `categoria_id` INT NULL DEFAULT NULL,
12     `fornecedor_id` INT NULL DEFAULT NULL,
13     `quantidade_em_estoque` INT NULL DEFAULT 0,
14     `nivel_reposicao` INT NULL DEFAULT 0,
15     `preco_unitario` DECIMAL(10,2) NULL DEFAULT NULL,
16 PRIMARY KEY (`produto_id`);
```

Inserindo nas tabelas

```
1  INSERT INTO Categoria (categoria_id, nome, descricao) VALUES
2  (1, 'Eletrnicos', 'Aparelhos eletrnicos e gadgets'),
3  (2, 'Roupas', 'Vesturio e acessrios'),
4  (3, 'Alimentos', 'Produtos alimentcios'),
5  (4, 'Mveis', 'Mveis para casa e escritrio'),
6  (5, 'Livros', 'Livros e materiais de leitura');
7
8  INSERT INTO Produto (produto_id, nome, descricao, categoria_id,
       fornecedor_id, quantidade_em_estoque, nivel_reposicao,
       preco_unitario) VALUES
9  (1, 'Smartphone', 'Aparelho celular com tela touch', 1, 1, 50, 10,
       1200.00),
10 (2, 'Camiseta', 'Camiseta de algodo', 2, 2, 200, 50, 20.00),
11 (3, 'Arroz', 'Pacote de arroz de 1kg', 3, 3, 500, 100, 5.00),
12 (4, 'Mesa de Escritrio', 'Mesa de madeira para escritrio', 4, 4, 30,
       5, 300.00),
13 (5, 'Livro de Fico', 'Livro de fico cientfica', 5, 5, 100, 20, 35.00)
       ;
```

7 DISCIPLINA: AAP - ASPECTOS ÁGEIS DE PROGRAMAÇÃO

Na disciplina de Aspectos Ágeis de Programação, o projeto teve como objetivo aplicar princípios de *clean code* (Martin, 2008) a um trecho de código já funcional, de uma implementação de *Bubble Sort* (Cormen *et al.*, 2022). A proposta foi manter o comportamento do algoritmo, mas melhorar sua legibilidade, coesão e clareza, realizando ao menos três alterações e justificando cada uma delas. Esse tipo de atividade mostra que qualidade em desenvolvimento ágil não está apenas em entregar funcionalidades, mas em entregar código que outras pessoas consigam ler, dar manutenção e evoluir sem gerar dívidas técnicas.

A relevância para o desenvolvimento ágil está justamente nisso: código limpo torna baratas as mudanças que acontecem a cada *sprint*. Quando nomes são objetivos, quando blocos longos são quebrados em funções menores e quando comentários redundantes são removidos, adicionar uma validação, trocar uma estrutura de repetição ou adaptar uma regra passa a ser uma alteração pequena, compatível com ciclos curtos.

Esse projeto também se conecta às demais disciplinas porque prepara o terreno técnico para todas elas. Testes automatizados ficam mais fáceis de escrever sobre código enxuto e com responsabilidades claras; *DevOps* consegue colocar esse código em *pipelines* com menos ajustes; e as disciplinas de *web*, *mobile* e modelagem conseguem mapear com mais fidelidade as histórias de usuário para funções do sistema. Em síntese, *clean code* é o que garante que o que foi planejado em ágil consiga ser entregue em ágil.

7.1 ARTEFATOS DO PROJETO

BubbleSortCleanCode.java

```java
// Implementao otimizada em Java do Bubble sort
// Cdigo extrado de https://www.geeksforgeeks.org/bubble-sort/

import java.io.*;

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

   static void bubbleSort(int array[]) {
      int arraySize = array.length;
      boolean swapped;
      for (int i = 0; i < arraySize - 1; i++) {
         swapped = false;
         for (int j = 0; j < arraySize - i - 1; j++) {
            if (array[j] > array[j + 1]) {
               swapArrayPositions(array, j);
               swapped = true;
            }
         }
         if (!swapped)
            break;
      }
   }

   static void swapArrayPositions(int array[], int arrayPosition) {
      int temp = array[arrayPosition];
```

```
32          array[arrayPosition] = array[arrayPosition + 1];
33          array[arrayPosition + 1] = temp;
34      }
35
36      static void printArray(int[] array) {
37          for (int value : array) {
38              System.out.print(value + " ");
39          }
40          System.out.println();
41      }
42
43  }
44
45  // This code is contributed
46  // by Nikita Tiwari.
47
48  // MUDANAS PARA CLEAN CODE
49  // 1 - MELHORIA NAS NOMENCLATURAS
50  // 2 - CRIAO DA  FUNO  PARA TROCAR AS POSIES DO ARRAY
51  // 3 - DIMINUIO DE PARAMETROS NAS FUNES
52  // 4 - REMOO DE COMENTRIOS DESNECESSRIOS
53  // 5 - MELHORA DA IDENTAO
54  // 6 - SIMPLIFICAO DO FOR AO IMPRIMIR ARRAY
55  // 7 - REPOSICIONA FUNCOES PARA SEGUIR A SEQUENCIA LOGICA DO CODIGO
```

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

Na disciplina de Desenvolvimento Web 1, o projeto teve como objetivo construir uma aplicação *Angular* (Angular, 2025) simples, porém completa, com dois *CRUDs* (Create, Read, Update, Delete)[1] (Aluno e Curso) no mesmo projeto, rotas configuradas, interface feita com *Bootstrap* (Bootstrap, 2025) ou *Material Design* (Google, 2025) e persistência em *Local Storage*. Trata-se de um cenário típico de *front-end* ágil: entregar rapidamente telas funcionais, com criação, listagem, edição e remoção, mesmo sem depender ainda de uma *API* externa.

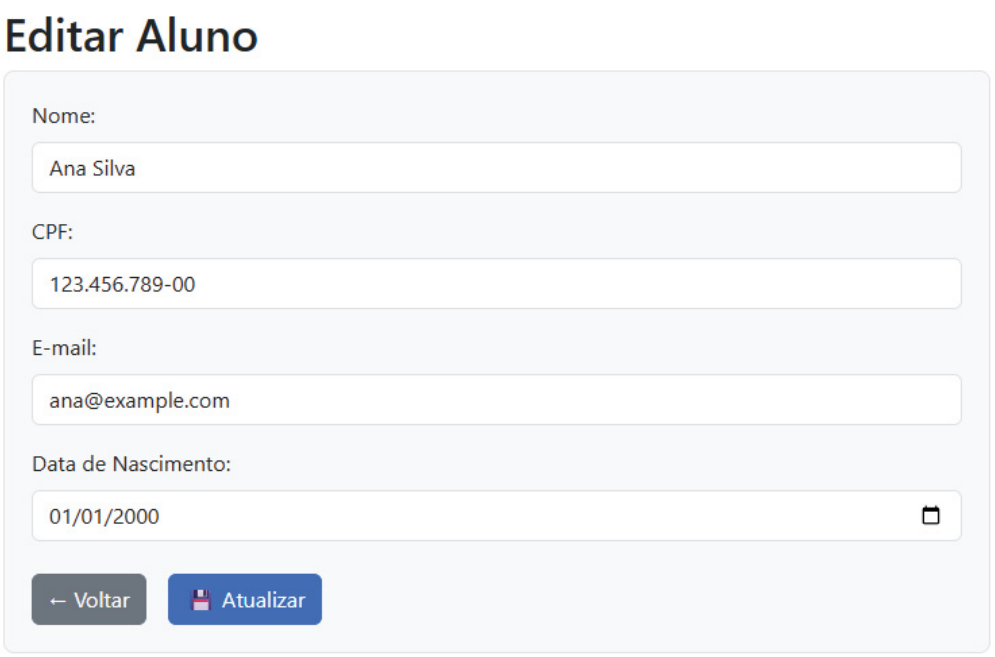
Na disciplina de Desenvolvimento Web 2, o mesmo domínio foi evoluído, com a inclusão do CRUD de Matrícula e com a integração de um *backend* em *Spring Boot* (Spring, 2025) e banco de dados *PostgreSQL* (PostgreSQL, 2025). A ênfase passa da simples construção de telas para a separação de camadas, consumo de serviços e persistência real, mostrando que uma aplicação iniciada de forma local pode, em iterações, tornar-se uma arquitetura cliente-servidor mais robusta.

A integração dessas duas disciplinas com o restante do curso é direta: os *CRUDs* podem ser derivados das histórias geradas em *MAG* e planejadas em *GAP*; o *backend* e o banco se conectam ao que foi visto em Banco de Dados. Em termos de filosofia ágil, fica nítido que entregar em pequenos passos não significa entregar algo incompleto, mas algo utilizável naquele estágio da arquitetura.

[1] CRUD: conjunto de operações fundamentais de persistência: criar, ler, atualizar e remover registros.

8.1 ARTEFATOS DO PROJETO

Figura 8.1 – TELA EDITAR ALUNO



Fonte: O autor (2025)

Figura 8.2 – TELA EDITAR MATRÍCULA

Fonte: O autor (2025)

src\app\aluno\editar-aluno\editar-aluno.component.ts

```ts
import { CommonModule } from "@angular/common";
import { Component, ViewChild } from "@angular/core";
import { FormsModule, NgForm } from "@angular/forms";
import { ActivatedRoute, Router, RouterModule } from "@angular/router
  ";
import { AlunoModule } from "../aluno.module";
import { Aluno } from "../../shared/models/aluno.model";
import { AlunoService } from "../services/aluno.service";

@Component({
  selector: "app-editar-aluno",
  standalone: true,
  imports: [RouterModule, CommonModule, FormsModule, AlunoModule],
  templateUrl: "./editar-aluno.component.html",
  styleUrl: "./editar-aluno.component.css",
})
export class EditarAlunoComponent {
  @ViewChild("formAluno") formAluno!: NgForm;

  aluno: Aluno = new Aluno();

  constructor(
    private alunoService: AlunoService,
    private route: ActivatedRoute,
    private router: Router
  ) {}

  ngOnInit(): void {
    let id = +this.route.snapshot.params["id"];

    const res = this.alunoService.buscarPorID(id);
    if (res !== undefined) {
      this.aluno = res;
```

```
33      } else {
34        throw new Error("Aluno nao encontrado: id = " + id);
35      }
36    }
37
38    atualizar(): void {
39      if (this.formAluno.form.valid) {
40        this.alunoService.atualizar(this.aluno);
41        this.router.navigate(["/alunos"]);
42      }
43    }
44  }
```

src\app\aluno\editar-aluno\editar-aluno.component.html

```
1  <h1>Editar Aluno</h1>
2
3  <div class="well">
4     <form #formAluno="ngForm">
5        <div class="form-group">
6           <label for="nome">Nome:</label>
7           <input [(ngModel)]="aluno.nome" #nome="ngModel" type="text"
               class="form-control" id="nome" name="nome"
8              minlength="2" required>
9           <div *ngIf="nome.errors && (nome.dirty || nome.touched)"
               class="alert alert-danger">
10             <div [hidden]="!nome.errors['required']">Digite o nome do
                   aluno.</div>
11             <div [hidden]="!nome.errors['minlength']">O nome deve
                  conter ao menos 2 caracteres.</div>
12          </div>
13       </div>
14       <div class="form-group mt-3">
15          <label for="idade">CPF:</label>
```

```
16          <input [(ngModel)]="aluno.cpf" #cpf="ngModel" type="text"
               class="form-control" id="cpf" name="cpf" required>
17          <div *ngIf="cpf.errors && (cpf.dirty || cpf.touched)" class=
               "alert alert-danger">
18             <div [hidden]="!cpf.errors['required']">Digite o cpf do
                  aluno.</div>
19          </div>
20       </div>
21       <div class="form-group mt-3">
22          <label for="idade">E-mail:</label>
23          <input [(ngModel)]="aluno.email" #email="ngModel" type="text
               " class="form-control" id="email" name="email"
24             required>
25          <div *ngIf="email.errors && (email.dirty || email.touched)"
               class="alert alert-danger">
26             <div [hidden]="!email.errors['required']">Digite o e-mail
                   do aluno.</div>
27          </div>
28       </div>
29       <div class="form-group mt-3">
30          <label for="idade">Data de Nascimento:</label>
31          <input [(ngModel)]="aluno.dataNasc" #dataNasc="ngModel" type
               ="text" class="form-control" id="dataNasc"
32             name="dataNasc" required>
33          <div *ngIf="dataNasc.errors && (dataNasc.dirty || dataNasc.
               touched)" class="alert alert-danger">
34             <div [hidden]="!dataNasc.errors['required']">Digite a
                  data de nascimento do aluno.</div>
35          </div>
36       </div>
37       <div class="form-group">
38          <div class="form-group row gy-2 gx-3 align-items-center mt-3
               ">
39             <div class="col-auto">
```

```
40                <a class="btn btn-secondary" [routerLink]="['/alunos']
                     ">
41                   <i class="fa fa-arrow-left" aria-hidden="true"></i>
                        Voltar
42                </a>
43             </div>
44             <div class="col-auto">
45                <button type="button" class="btn btn-primary" (click)=
                     "atualizar()" [disabled]="!formAluno.form.valid">
46                   <i class="fa fa-save" aria-hidden="true"></i>
                        Atualizar
47                </button>
48             </div>
49          </div>
50       </div>
51    </form>
52 </div>
```

src\app\matricula\editar-matricula\editar-matricula.component.ts

```
1  import { Component, ViewChild } from "@angular/core";
2  import { CommonModule } from "@angular/common";
3  import { FormsModule, NgForm } from "@angular/forms";
4  import { ActivatedRoute, Router, RouterModule } from "@angular/router
      ";
5  import { Matricula } from "../../shared/models/matricula.model";
6  import { MatriculaService } from "../services/matricula.service";
7  import { Aluno } from "../../shared/models/aluno.model";
8  import { Curso } from "../../shared/models/curso.model";
9  import { AlunoService } from "../../aluno/services/aluno.service";
10 import { CursoService } from "../../curso/services/curso.service";
11
12 @Component({
13   selector: "app-editar-matricula",
14   standalone: true,
```

```
15    imports: [CommonModule, FormsModule, RouterModule],
16    templateUrl: "./editar-matricula.component.html",
17    styleUrl: "./editar-matricula.component.css",
18  })
19  export class EditarMatriculaComponent {
20    @ViewChild("formMatricula") formMatricula!: NgForm;
21
22    matricula: Matricula = new Matricula();
23    alunos: Aluno[] = [];
24    cursos: Curso[] = [];
25
26    constructor(
27      private matriculaService: MatriculaService,
28      private alunoService: AlunoService,
29      private cursoService: CursoService,
30      private route: ActivatedRoute,
31      private router: Router
32    ) {}
33
34    ngOnInit(): void {
35      const id = +this.route.snapshot.params["id"];
36      const res = this.matriculaService.buscarPorID(id);
37
38      this.alunos = this.alunoService.listarTodos();
39      this.cursos = this.cursoService.listarTodos();
40
41      if (res !== undefined) {
42        this.matricula = res;
43      } else {
44        throw new Error("Matricula nao encontrada: id = " + id);
45      }
46    }
47
48    atualizar(): void {
```

```
49      if (this.formMatricula.form.valid) {
50        // Atualiza as referencias de aluno e curso
51        this.matricula.aluno = this.alunos.find(a => a.id == this.
              matricula.aluno.id)!;
52        this.matricula.curso = this.cursos.find(c => c.id == this.
              matricula.curso.id)!;
53
54        this.matriculaService.atualizar(this.matricula);
55        this.router.navigate(["/matriculas"]);
56      }
57    }
58  }
```

src\app\matricula\editar-matricula\editar-matricula.component.html

```
1  <h1>Editar Matricula</h1>
2
3  <div class="well">
4    <form #formMatricula="ngForm">
5      <div class="form-group">
6        <label>Aluno:</label>
7        <select
8          class="form-control"
9          required
10         [(ngModel)]="matricula.aluno.id"
11         name="aluno"
12       >
13         <option [ngValue]="undefined">-- Selecione um aluno --</option>
14         <option *ngFor="let aluno of alunos" [value]="aluno.id">
15           {{ aluno.nome }}
16         </option>
17       </select>
18     </div>
19
20     <div class="form-group mt-3">
```

```
21      <label>Curso:</label>
22      <select
23        class="form-control"
24        required
25        [(ngModel)]="matricula.curso.id"
26        name="curso"
27      >
28        <option [ngValue]="undefined">-- Selecione um curso --</option>
29        <option *ngFor="let curso of cursos" [value]="curso.id">
30          {{ curso.nome }}
31        </option>
32      </select>
33    </div>
34
35    <div class="form-group mt-3">
36      <label>Data da Matricula:</label>
37      <input
38        [(ngModel)]="matricula.data"
39        name="data"
40        type="date"
41        class="form-control"
42        required
43      />
44    </div>
45
46    <div class="form-group mt-3">
47      <label>Nota:</label>
48      <input
49        [(ngModel)]="matricula.nota"
50        name="nota"
51        type="number"
52        class="form-control"
53        required
54        min="0"
```

```
55        max="10"
56        step="0.1"
57      />
58    </div>
59
60    <div class="form-group row gy-2 gx-3 align-items-center mt-3">
61      <div class="col-auto">
62        <a class="btn btn-secondary" [routerLink]="['/matriculas']">
63          <i class="fa fa-arrow-left" aria-hidden="true"></i> Voltar
64        </a>
65      </div>
66      <div class="col-auto">
67        <button
68          type="button"
69          class="btn btn-primary"
70          (click)="atualizar()"
71          [disabled]="!formMatricula.form.valid"
72        >
73          <i class="fa fa-save" aria-hidden="true"></i> Atualizar
74        </button>
75      </div>
76    </div>
77  </form>
78</div>
```

9 DISCIPLINA: UX - UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

Na disciplina de UX no Desenvolvimento Ágil de Software, o projeto teve como objetivo mostrar que a experiência do usuário pode ser concebida e validada no mesmo ritmo das entregas técnicas. Primeiro foi necessário explicar o produto proposto, descrevendo sua função principal e a justificativa do problema que ele resolve, alinhando o exercício à prática ágil de entender o valor antes de implementar (Gothelf e Seiden, 2021). Em seguida, foram produzidas cinco telas completas, já com navegação, cores e componentes definidos, simulando um protótipo de baixa ou média fidelidade que pudesse ser apresentado mesmo sem backend pronto.

Outro ponto relevante foi a justificativa das escolhas de cor, tipografia, espaçamento e *layout*. Registrar essas decisões torna o projeto orientado ao usuário e facilita o alinhamento com quem vai desenvolver, reduzindo retrabalho em *sprints* posteriores. Por fim, a etapa de mostrar as telas para um possível usuário e coletar *feedback* traduz diretamente o ciclo ágil de entregar, observar e ajustar: quanto mais cedo o retorno chega, mais barato é corrigir fluxo, rótulos e hierarquia de informação.

Esse trabalho se integra às demais disciplinas porque fornece insumos visuais e de usabilidade para as aplicações desenvolvidas em *WEB 1* e *WEB 2*, dá material claro para priorização em *GAP*, a disciplina mostra que *UX* não vem depois do código: faz parte do próprio processo ágil de construção do software.

9.1 ARTEFATOS DO PROJETO

O HamonizAI

O HarmonizAI é um sistema desenvolvido para ajudar os usuários a encontrarem harmonizações ideais entre vinhos e pratos. A proposta é proporcionar uma experiência gastronômica, oferecendo sugestões de harmonização personalizadas baseadas em diferentes critérios, como tipo de vinho, tipo de comida e ocasião. Este projeto visa facilitar o processo de escolha de harmonizações, tornando-o acessível tanto para iniciantes quanto para apreciadores experientes de vinhos.

Uma das principais características do HarmonizAI é a utilização de Inteligência Artificial para oferecer recomendações precisas e personalizadas. A IA analisa uma vasta base de dados de vinhos e pratos, considerando sabores, aromas, texturas e outras características sensoriais, para sugerir as melhores combinações possíveis.

Será possível encontrar uma harmonização partindo de uma descrição, que pode conter um prato, uma ocasião ou um vinho. Se o usuário tiver um prato específico em mente, poderá inseri-lo no aplicativo, e a IA sugerirá vinhos que complementam os sabores e texturas daquele prato. Da mesma forma, se o usuário tiver um vinho e quiser saber quais pratos harmonizam melhor com ele, o sistema fornecerá sugestões de pratos que realçam as características do vinho escolhido.

Justificativa

O desenvolvimento do HarmonizAI auxiliará muitas pessoas que enfrentam dificuldades ao tentar combinar vinhos e pratos. Mesmo apreciadores de vinhos com mais experiência se deparam com dúvidas sobre qual bebida escolher para acompanhar determinado prato, e vice-versa. Ao incorporar IA, o HarmonizAI oferece uma solução prática e intuitiva, capaz de auxiliar os usuários de maneira eficiente e personalizada.

Entre os benefícios oferecidos aos usuários estão: aprender sobre vinhos e harmonizações, aumentar o prazer em refeições com combinações adequadas e receber recomendações personalizadas com base em preferências e restrições alimentares. O uso da IA permite que o aplicativo se adapte às necessidades individuais de cada usuário, proporcionando uma experiência única e enriquecedora.

Cores

As cores escolhidas remetem diretamente ao universo do vinho, criando uma conexão imediata com o propósito do aplicativo. As combinações de cores foram pensadas para garantir contraste suficiente, atendendo a critérios de acessibilidade para pessoas com deficiências visuais.

- #3E1F25 (Vinho Escuro): Utilizado como fundo dos cards, traz um tom sofisticado e elegante. Essa cor remete diretamente ao tom de vinho tinto, associando o aplicativo ao universo da enologia.
- #5F0E1D (Vinho Médio): Utilizado como a cor primária nos botões. Essa cor vibrante sugere ação e convida o usuário a interagir, proporcionando um toque elegante que remete à sofisticação do universo dos vinhos.
- #777E31 (Verde Oliva): Aplicado como detalhe no logo. O verde oliva traz uma sensação de equilíbrio e frescor, remetendo às folhas de videira.
- #DFDFDF (Cinza Claro): Utilizada como cor de fundo, proporciona um contraste suave com as cores mais escuras, facilitando a leitura e destacando os elementos principais sem cansar a visão do usuário.
- #282828 (Cinza Escuro): Utilizada como fundo dos menus superior e inferior. Esta cor proporciona um forte contraste com os elementos interativos do menu, facilitando a navegação e destacando os ícones e textos presentes nesses espaços.

Layout

O projeto foi desenvolvido com layout minimalista, evitando poluição visual e focando na usabilidade. Os elementos são organizados de forma lógica, com hierarquia visual bem definida, com uso de tamanhos diferentes de fontes e espaçamentos, permitindo que o usuário identifique rapidamente as informações mais importantes. Espaços em branco, respiros, são utilizados para separar seções e elementos, tornando a interface mais leve.

Fonte

A fonte 'Roboto' foi escolhida pela sua legibilidade e aparência moderna. É uma fonte sem serifa que combina bem com o estilo clean do aplicativo. Variações de peso (regular, medium, bold) para destacar títulos, subtítulos e textos importantes, mantém a consistência visual.

Ícones e Elementos Gráficos

Os ícones utilizados, como a lupa para busca e os ícones da barra inferior, são facilmente reconhecíveis e contribuem para a intuitividade do aplicativo. As imagens são de alta resolução, atraindo a atenção do usuário e enriquecendo a experiência visual.

Feedback do Usuário

Para coletar o feedback, apresentamos as telas do HarmonizAI a um usuário em potencial, Maria Eduarda, que é apreciadora de vinhos e gosta de cozinhar em casa.

Maria comentou que a interface parece amigável e intuitiva. Ela gostou da presença do campo de busca logo na tela inicial, o que facilita o início da interação. Maria gostou da possibilidade de buscar harmonizações a partir de um prato ou vinho. Ela mencionou que frequentemente tem dúvidas sobre qual vinho combinar com suas receitas e que o aplicativo seria muito útil.

Ela sugeriu incluir uma seção de "Harmonizações Populares"na tela inicial, para descobrir novas combinações sem necessariamente buscar por um prato ou vinho específico.

Figura 9.1 – HAMONIZAI - LOGIN

Fonte: O autor (2025)

Figura 9.2 – HAMONIZAI - DASHBOARD

Fonte: O autor (2025)

Figura 9.3 – HAMONIZAI - BUSCAR

Fonte: O autor (2025)

Figura 9.4 – HAMONIZAI - RESULTADOS

Fonte: O autor (2025)

Figura 9.5 – HAMONIZAI - DETALHES

Fonte: O autor (2025)

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

Na disciplina de Desenvolvimento Mobile 1, o projeto teve como objetivo criar um aplicativo Android simples para controle financeiro pessoal, com fluxos para cadastro e consulta de lançamentos. Foi implementado o cadastro de operações de débito e crédito, com armazenamento em memória, reforçando a ideia de entregar primeiro a funcionalidade essencial, mesmo sem persistência definitiva. Esse formato dialoga com o desenvolvimento ágil porque mostra que é possível disponibilizar valor rápido ao usuário e, em iterações seguintes, evoluir para recursos mais completos, como filtros, categorias ou banco de dados.

Na disciplina de Desenvolvimento Mobile 2, o mesmo padrão de aplicação foi levado para um cenário mais avançado. A estrutura de *dashboard* foi mantida, mas cada opção passou a acionar uma operação remota, ilustrando uma progressão típica da agilidade: primeiro resolver o fluxo interno, depois conectar com serviços externos.

Esses dois trabalhos se integram ao restante do curso porque podem consumir as histórias produzidas em *MAG* e planejadas em *GAP* e podem aproveitar as telas concebidas em *UX*. Em termos de metodologia ágil, as duas disciplinas enfatizam o mesmo ponto: aplicações móveis também devem ser construídas em pequenos incrementos, começando por uma versão funcional mínima e avançando para integrações externas, sempre com código versionado e com entregas que possam ser demonstradas.

10.1 ARTEFATOS DO PROJETO

MainActivity.kt

```
package com.example.finapp

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.Button

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        findViewById<Button>(R.id.btnCadastro).setOnClickListener {
            startActivity(Intent(this, CadastroActivity::class.java))
        }

        findViewById<Button>(R.id.btnExtrato).setOnClickListener {
            startActivity(Intent(this, ExtratoActivity::class.java))
        }

        findViewById<Button>(R.id.btnSair).setOnClickListener {
            finishAffinity()
        }
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:id="@+id/layoutDashboard"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="24dp"
    android:background="#FAFAFA">

    <Button
        android:id="@+id/btnCadastro"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Cadastrar Operao"
        android:layout_marginBottom="16dp"
        android:backgroundTint="#4CAF50"
        android:textColor="#FFFFFF" />

    <Button
        android:id="@+id/btnExtrato"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ver Extrato"
        android:layout_marginBottom="16dp"
        android:backgroundTint="#2196F3"
        android:textColor="#FFFFFF" />

    <Button
        android:id="@+id/btnSair"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sair"
        android:backgroundTint="#F44336"
        android:textColor="#FFFFFF" />
</LinearLayout>
```

MainActivity.kt

```
package com.example.finapp
```

```
2
3  import android.content.Intent
4  import android.os.Bundle
5  import android.widget.Button
6  import androidx.appcompat.app.AppCompatActivity
7
8  class MainActivity : AppCompatActivity() {
9      override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         findViewById<Button>(R.id.btnPersonagem).setOnClickListener {
14             startActivity(Intent(this, PersonagemActivity::class.java))
15         }
16
17         findViewById<Button>(R.id.btnProfessores).setOnClickListener {
18             startActivity(Intent(this, ProfessoresActivity::class.java))
19         }
20
21         findViewById<Button>(R.id.btnEstudantes).setOnClickListener {
22             startActivity(Intent(this, EstudantesCasaActivity::class.
                   java))
23         }
24
25         findViewById<Button>(R.id.btnSair).setOnClickListener {
26             finishAffinity()
27         }
28     }
29 }
```

activity_main.xml

```
1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent"
```

```
4     android:orientation="vertical"
5     android:gravity="center"
6     android:padding="24dp">
7
8     <Button
9         android:id="@+id/btnPersonagem"
10        android:layout_width="match_parent"
11        android:layout_height="wrap_content"
12        android:text="Buscar Personagem por ID"
13        android:layout_marginBottom="16dp" />
14
15    <Button
16        android:id="@+id/btnProfessores"
17        android:layout_width="match_parent"
18        android:layout_height="wrap_content"
19        android:text="Listar Professores"
20        android:layout_marginBottom="16dp" />
21
22    <Button
23        android:id="@+id/btnEstudantes"
24        android:layout_width="match_parent"
25        android:layout_height="wrap_content"
26        android:text="Estudantes por Casa"
27        android:layout_marginBottom="16dp" />
28
29    <Button
30        android:id="@+id/btnSair"
31        android:layout_width="match_parent"
32        android:layout_height="wrap_content"
33        android:text="Sair" />
34 </LinearLayout>
```

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

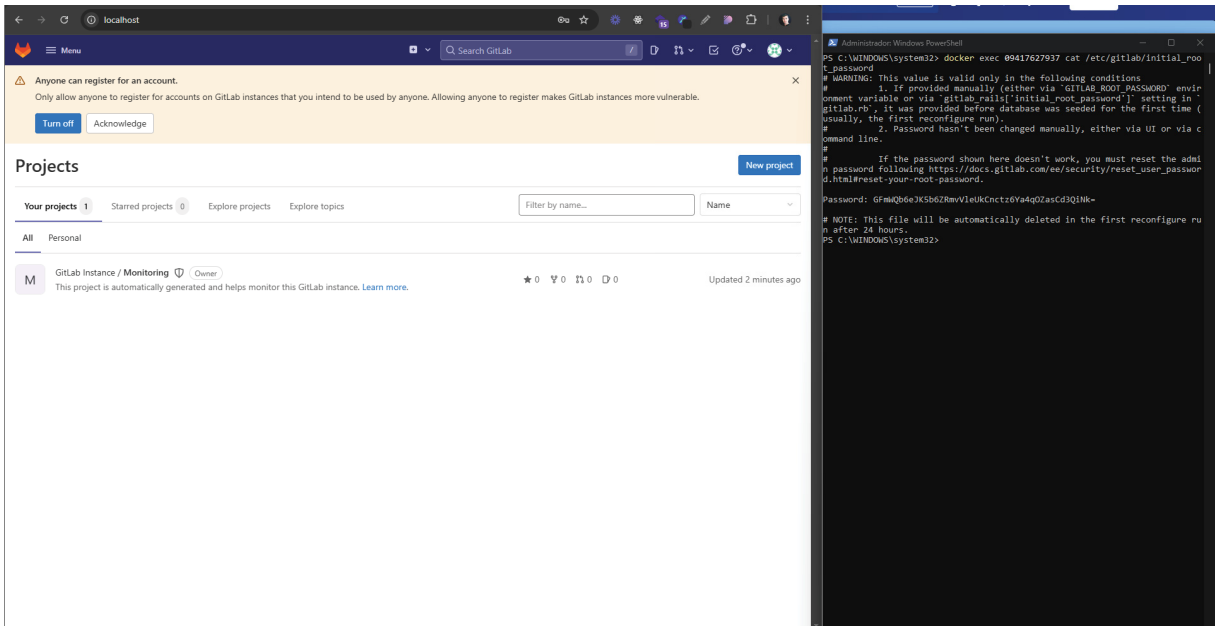
Na disciplina de Infraestrutura para desenvolvimento e implantação de software, o projeto teve como objetivo montar um ambiente de *DevOps* local usando *Docker* (Docker, 2025), a partir da imagem `dfwandarti/gitlab_jenkins:3`, nomeando o contêiner com a matrícula e publicando as portas 22, 80, 443 e 9091. Também foi necessário acessar o *GitLab* (GitLab, 2025) interno do contêiner, localizar a senha inicial em `/etc/gitlab/initial_root_password` e autenticar via navegador, concluindo com *commit* e *push* em um projeto existente. Todo o processo deveria ser documentado por meio de capturas de tela, garantindo evidência da execução correta dos passos.

A importância desse projeto para o desenvolvimento ágil de software está em mostrar que não há entrega contínua sem ambiente previsível. Subir o ambiente por contêiner, versionar em servidor central e registrar a execução são práticas que evitam variações entre máquinas, reduzem tempo de configuração e permitem que cada incremento de código possa ser testado e integrado rapidamente.

A integração desse projeto com as demais disciplinas é direta: o código produzido em *WEB 1 e 2*, *MOBILE 1 e 2*, nas disciplinas de programação e até nos exercícios de testes pode ser enviado para esse repositório e ter *pipelines* configurados. Dessa forma, a disciplina de Infraestrutura funciona como o elo técnico que sustenta a cadência de entrega proposta por todo o curso.

11.1 ARTEFATOS DO PROJETO

Figura 11.1 – DOCKER SENHA



Fonte: O autor (2025)

Figura 11.2 – DOCKER RODANDO

Fonte: O autor (2025)

Figura 11.3 – DOCKER COMMIT

Fonte: O autor (2025)

12 DISCIPLINA: TEST - TESTES AUTOMATIZADOS

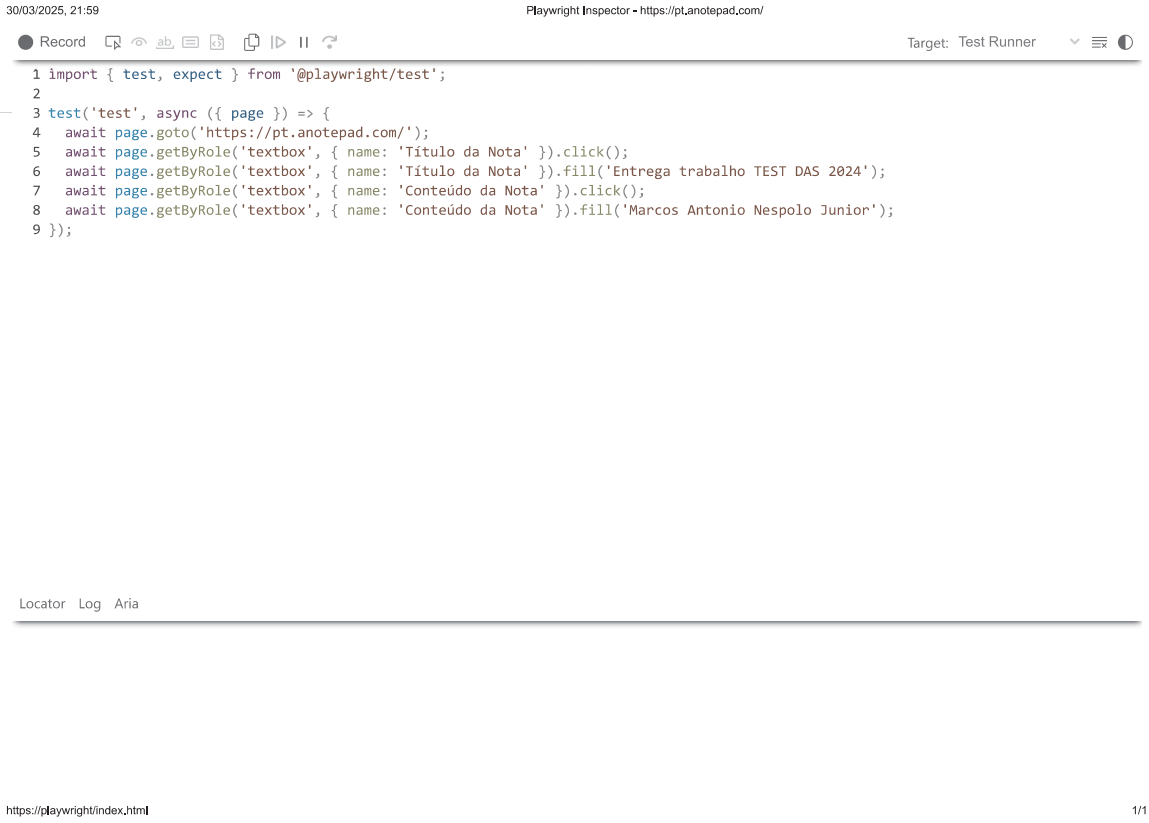
Na disciplina de Testes Automatizados, o projeto teve como objetivo automatizar um fluxo simples em uma aplicação *web* real (aNotepad, 2025), criando uma nota com o título "Entrega trabalho TEST DAS 2024" e registrando no corpo nome e matrícula. O exercício envolveu exatamente as etapas de um teste funcional: abrir a página, identificar os componentes da interface, preencher os campos corretos e confirmar que o resultado foi gravado. Mesmo sendo um cenário pequeno, ele representa o que ferramentas de automação fazem em sistemas maiores.

A importância desse projeto para o desenvolvimento ágil de software está em mostrar que uma entrega só pode entrar no ciclo de *release* contínuo se puder ser verificada automaticamente. Em processos ágeis, telas mudam, rotas mudam e componentes são refatorados; se não houver testes automatizados cobrindo os fluxos principais, cada mudança passa a depender de validação manual, o que desacelera o time e aumenta o risco de regressão.

Esse projeto se integra às demais disciplinas porque pode ser usado para validar o que foi construído em *WEB 1 e 2* e *MOBILE 1 e 2* e para alimentar o fluxo de *DevOps*, que depende de testes executáveis para publicar novas versões com segurança. Em síntese, a disciplina mostra que testes não ficam no final do processo: fazem parte do próprio mecanismo que permite ao desenvolvimento ágil entregar rápido sem perder qualidade.

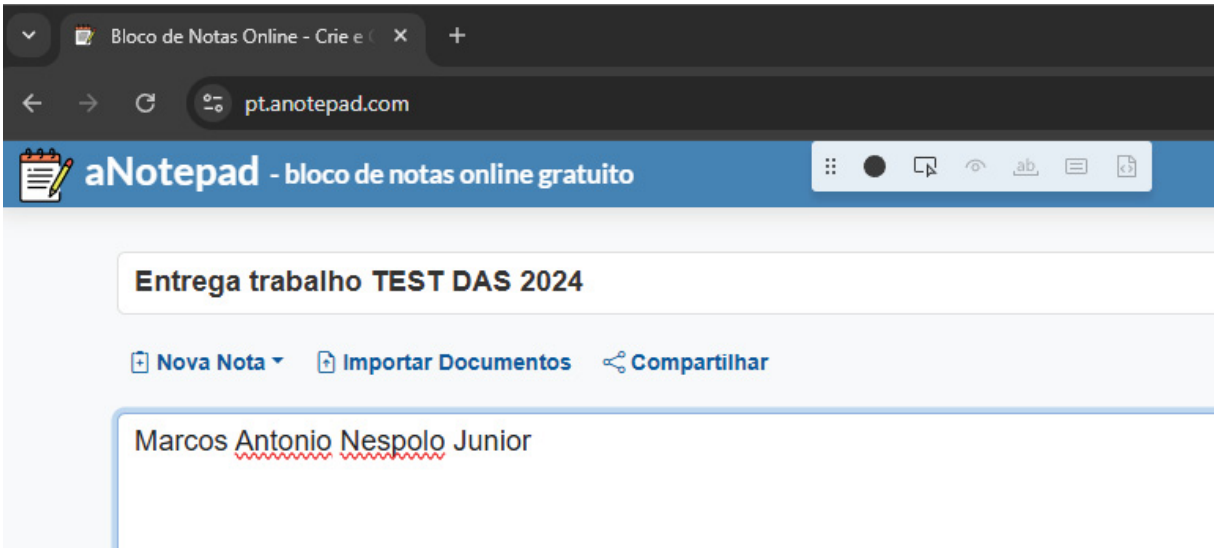
12.1 ARTEFATOS DO PROJETO

Figura 12.1 – TESTES AUTOMATIZADOS - CÓDIGO



Fonte: O autor (2025)

Figura 12.2 – TESTES AUTOMATIZADOS - ANOTEPAD



Fonte: O autor (2025)

13 CONCLUSÃO

Este memorial apresentou a trajetória das disciplinas do curso mostrando que todas elas apontam para o mesmo eixo: conceber, construir e evoluir sistemas em ciclos curtos, com foco em entrega de valor. A disciplina de Métodos Ágeis de Desenvolvimento de Software introduziu os conceitos, valores e práticas que orientam o restante do curso; em seguida, Modelagem Ágil de Software 1 e 2 mostrou como transformar requisitos em artefatos leves e atualizáveis; Gerenciamento Ágil de Projetos 1 e 2 organizou essas entregas no tempo, por *sprints* e por fluxo. A partir daí, Introdução à Programação e Banco de Dados deram o suporte técnico para implementar soluções funcionais e persistentes; Aspectos Ágeis de Programação reforçou a importância de *clean code*; Desenvolvimento *Web* 1 e 2 e UX no Desenvolvimento Ágil de Software levaram os artefatos para interfaces reais; Desenvolvimento *Mobile* 1 e 2 ampliou o mesmo raciocínio para o contexto móvel; Infraestrutura para desenvolvimento e implantação de software (*DevOps*) garantiu ambiente reprodutível e versionado; e Testes Automatizados fechou o ciclo com verificação contínua. O conjunto confirma o que o parecer técnico descreve: o ágil não é uma técnica isolada, mas uma forma integrada de trabalhar.

Cada disciplina reforçou que entregas devem ser pequenas, frequentes e verificáveis; que a modelagem deve ser *just-enough*; que testes e *pipelines* precisam acompanhar o código; e que decisões de design, dados e interface devem ser tomadas cedo para evitar desperdício. Ao percorrer análise, modelagem, construção, testes e implantação contínua, fica evidente a dimensão filosófica da agilidade: colaboração, simplicidade e adaptação contínua deixam de ser slogans e passam a ser critérios práticos para aceitar uma entrega.

Conclui-se, assim, que o curso apresentou a agilidade não como um método único, mas como um modo completo de organizar o trabalho de software ao longo de todo o ciclo de vida. Em síntese, o memorial demonstra que a filosofia ágil aplicada ao desenvolvimento de software pode ser praticada de forma responsável e incremental quando pessoas, processos e tecnologia são planejados em conjunto desde o início.

REFERÊNCIAS

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

ANGULAR. **Angular Documentation**. 2025. Disponível em: <https://angular.dev>. Acesso em: 7 nov. 2025.

ANOTEPAD. **aNotepad — Notepad Online**. 2025. Disponível em: <https://pt.anotepad.com>. Acesso em: 7 nov. 2025.

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

BECK, K. *et al.* **Manifesto for Agile Software Development**. 2001. Disponível em: <http://agilemanifesto.org>. Acesso em: 7 nov. 2025.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language User Guide**. 2. ed. Boston: Addison-Wesley, 2005. ISBN 0321267974.

BOOTSTRAP. **Bootstrap Documentation**. 2025. Disponível em: <https://getbootstrap.com/docs/>. Acesso em: 7 nov. 2025.

CORMEN, T. H. *et al.* **Introduction to Algorithms**. 4. ed. Cambridge, MA: MIT Press, 2022. ISBN 978-0262046305.

DOCKER. **Docker Documentation**. 2025. Disponível em: <https://docs.docker.com/>. Acesso em: 7 nov. 2025.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 7. ed. São Paulo: Pearson, 2016.

FOWLER, M. **Continuous Integration**. 2006. Disponível em: <https://martinfowler.com/articles/continuousIntegration.html>. Acesso em: 7 nov. 2025.

GITLAB. **GitLab Documentation**. 2025. Disponível em: <https://docs.gitlab.com/>. Acesso em: 7 nov. 2025.

GOOGLE. **Material Design Guidelines**. 2025. Disponível em: <https://m3.material.io/>. Acesso em: 7 nov. 2025.

GOTHELF, J.; SEIDEN, J. **Lean UX: Designing Great Products with Agile Teams**. 3. ed. Sebastopol, CA: O'Reilly Media, 2021. ISBN 978-1492052417.

HIGHSMITH, J. **Agile Software Development Ecosystems**. [S.l.]: Addison-Wesley, 2002.

JUNIT. **JUnit 5 User Guide**. 2025. Disponível em: <https://junit.org/junit5/docs/current/user-guide/>. Acesso em: 7 nov. 2025.

KANBAN. **Kanban Board Game: plataforma de simulação**. 2025. Disponível em: <https://kanbanboardgame.com>. Acesso em: 7 nov. 2025.

LARMAN, C. **Agile and Iterative Development: A Manager's Guide**. [S.l.]: Addison-Wesley, 2004.

MARTIN, R. C. **Clean Code: A Handbook of Agile Software Craftsmanship**. [S.l.]: Prentice Hall, 2008.

ORACLE. **MySQL 8.0 Reference Manual**. 2025. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/>. Acesso em: 7 nov. 2025.

POPPENDIECK, M.; POPPENDIECK, T. **Lean Software Development: An Agile Toolkit**. [S.l.]: Addison-Wesley, 2003.

POSTGRESQL. **PostgreSQL Documentation**. 2025. Disponível em: <https://www.postgresql.org/docs/current/>. Acesso em: 7 nov. 2025.

SCHWABER, K. **Agile Project Management with Scrum**. Redmond: Microsoft Press, 2004.

SHORE, J.; WARDEN, S. **The Art of Agile Development**. [S.l.]: O'Reilly Media, 2007.

SOMMERVILLE, I. **Software Engineering**. 9. ed. [S.l.]: Addison-Wesley, 2011.

SPRING. **Spring Boot Reference Documentation**. 2025. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Acesso em: 7 nov. 2025.