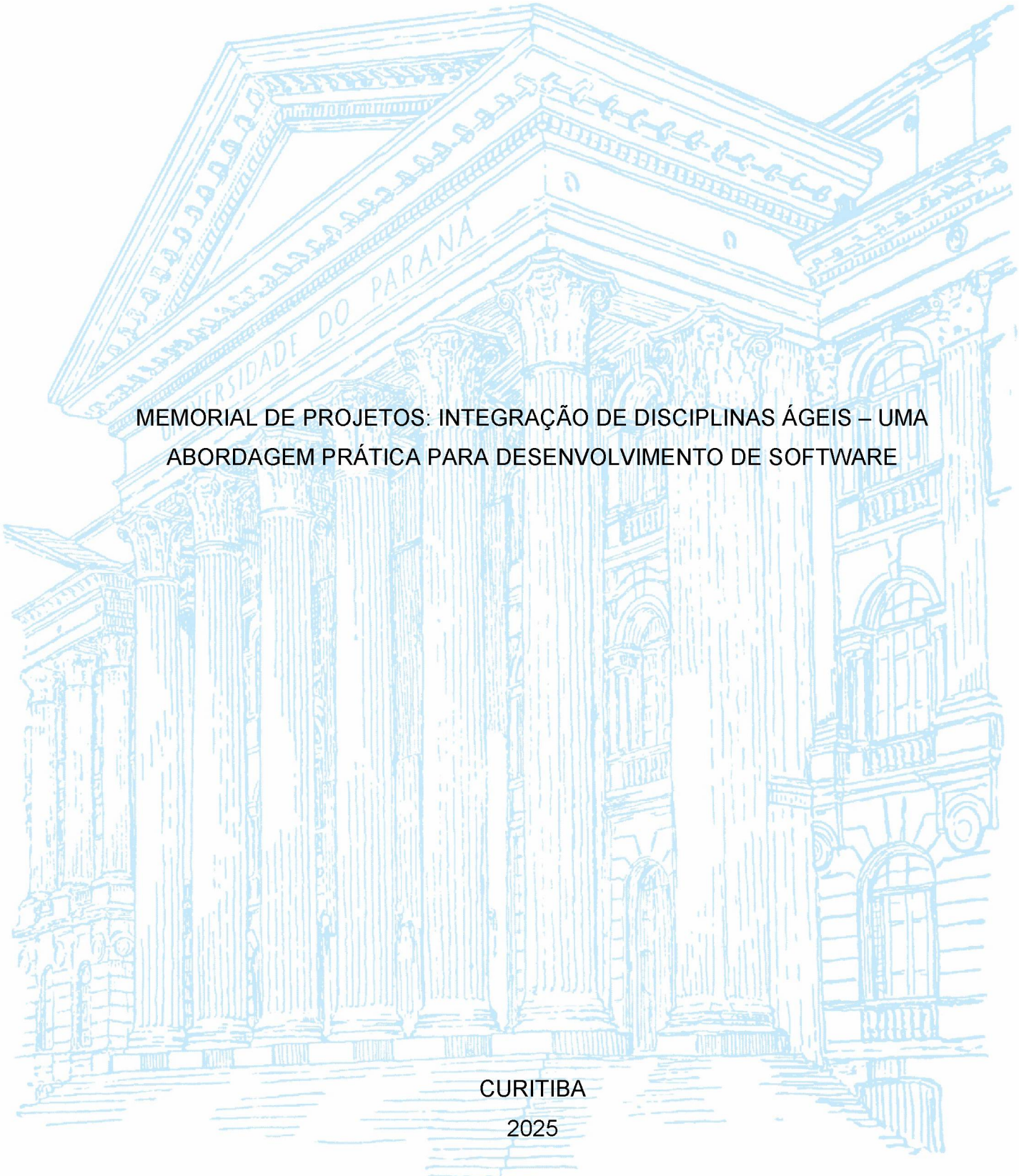
UNIVERSIDADE FEDERAL DO PARANÁ

PEDRO VIEIRA FERREIRA



MEMORIAL DE PROJETOS: INTEGRAÇÃO DE DISCIPLINAS ÁGEIS – UMA ABORDAGEM PRÁTICA PARA DESENVOLVIMENTO DE SOFTWARE

CURITIBA

2025

PEDRO VIEIRA FERREIRA

MEMORIAL DE PROJETOS: INTEGRAÇÃO DE DISCIPLINAS ÁGEIS – UMA ABORDAGEM PRÁTICA PARA DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Desenvolvimento Ágil de Software, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Desenvolvimento Ágil de Software.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Desenvolvimento Ágil de Software da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **PEDRO VIEIRA FERREIRA**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO DE DISCIPLINAS ÁGEIS UMA ABORDAGEM PRÁTICA PARA DESENVOLVIMENTO 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, 22 de Julho de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este trabalho apresenta uma trajetória prática pela evolução do desenvolvimento ágil de software, demonstrando como seus princípios fundamentais se aplicam em todas as etapas de construção de sistemas. Através de projetos integrados, explora desde a concepção de requisitos e modelagem enxuta até a implementação técnica, testes automatizados e entrega contínua, mantendo sempre o foco na geração de valor sustentável. Os artefatos documentados evidenciam a aplicação coerente de metodologias ágeis em contextos reais, destacando a importância da documentação precisa, do código limpo e bem estruturado, da sinergia entre automação de testes e integração contínua, e do papel central da experiência do usuário no ciclo de desenvolvimento. Mais do que um registro técnico, este memorial exemplifica como valores como colaboração, adaptabilidade e melhoria contínua transformam radicalmente a maneira de conceber e construir software, apresentando soluções que equilibram de forma harmoniosa agilidade, qualidade técnica e atenção às necessidades humanas - tanto dos desenvolvedores quanto dos usuários finais.

Palavras-chave: Ágil; Desenvolvimento; Software; Artefatos.

ABSTRACT

This work presents a practical journey through the evolution of agile software development, demonstrating how its core principles apply across all system construction phases. Through integrated projects, it explores from requirement conception and lean modeling to technical implementation, automated testing and continuous delivery, while consistently focusing on sustainable value creation. The documented artifacts demonstrate the coherent application of agile methodologies in real-world contexts, highlighting the importance of precise documentation, clean and well-structured code, the synergy between test automation and continuous integration, and the central role of user experience in the development lifecycle. Beyond being a technical record, this portfolio exemplifies how values like collaboration, adaptability and continuous improvement radically transform the way software is conceived and built, presenting solutions that achieve a harmonious balance between agility, technical quality and attention to human needs - for both developers and end users.

Keywords: Agile; Development; Software; Artifacts.

SUMÁRIO

1 PARECER TÉCNICO.....7
2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE.....9
2.1 ARTEFATOS DO PROJETO.....10
3 DISCIPLINA: MAG1 E MAG2 – MODELAGEM ÁGIL DE SOFTWARE 1 E 2.....12
3.1 ARTEFATOS DO PROJETO.....13
4 DISCIPLINA: GAP1 E GAP2 – GERENCIAMENTO ÁGIL DE PROJETOS DE SOFTWARE 1 E 2.....16
4.1 ARTEFATOS DO PROJETO.....17
5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO.....20
5.1 ARTEFATOS DO PROJETO.....21
6 DISCIPLINA: BD – BANCO DE DADOS.....24
6.1 ARTEFATOS DO PROJETO.....25
7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO.....27
7.1 ARTEFATOS DO PROJETO.....28
8 DISCIPLINA: WEB1 E WEB2 – DESENVOLVIMENTO WEB 1 E 2.....30
8.1 ARTEFATOS DO PROJETO.....31
9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE.....35
9.1 ARTEFATOS DO PROJETO.....36
10 DISCIPLINA: MOB1 E MOB2 – DESENVOLVIMENTO MOBILE 1 E 2.....39
10.1 ARTEFATOS DO PROJETO.....40
11 DISCIPLINA: INFRA - INFRAESTRUTURA PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SOFTWARE (DEVOPS).....44
11.1 ARTEFATOS DO PROJETO.....45
12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS.....47
12.1 ARTEFATOS DO PROJETO.....48
13 CONCLUSÃO.....50
REFERÊNCIAS.....52

1 PARECER TÉCNICO

O presente parecer tem como objetivo apresentar uma análise dos projetos desenvolvidos ao longo do curso de Pós-Graduação em Desenvolvimento Ágil de Software, com base na aplicação dos princípios, valores e métodos ágeis nas diferentes disciplinas cursadas. A proposta aqui é evidenciar como o conjunto de projetos se articula de forma progressiva e coerente, refletindo uma abordagem prática e iterativa do desenvolvimento de software.

Tal caminho iniciou com a disciplina de Métodos Ágeis para Desenvolvimento de Software (MADS), que estabeleceu o alicerce conceitual do curso. A construção do mapa mental demonstrou o comprometimento com os valores do Manifesto Ágil (Beck *et al.*, 2001), fornecendo um *framework* conceitual que permitiu a compreensão de demais disciplinas. A articulação entre métodos como *Scrum* (Schwaber; Sutherland, 2020), *Kanban* (Anderson, 2010), *XP* (Beck, 2000) *e Lean* reforçou a importância da flexibilidade e da entrega de valor contínuo, principais pilares das práticas ágeis.

Nas disciplinas de MAG1 - Modelagem Ágil de Software 1 e MAG2- Modelagem Ágil de Software 2, essa base foi colocada em prática com a construção de artefatos mais enxutos e eficazes, como histórias de usuário, diagramas de caso de uso e diagramas de sequência. A modelagem deu foco na comunicação entre *stakeholders* e no alinhamento com as necessidades do negócio, promovendo uma modelagem que impulsiona o desenvolvimento incremental (Cohn, 2004).

A partir de então, as disciplinas técnicas se beneficiaram diretamente desses elementos. Em INTRO – Introdução à Programação, por exemplo, os testes unitários e o desenvolvimento orientado a objetos encontraram suporte conceitual nos critérios de aceite previamente definidos nas histórias de usuário. Já em Banco de Dados (BD), o Diagrama Entidade-Relacionamento (FIGURA 12) foi influenciado pelo diagrama de classes de MAG2, tornando clara a interdependência entre modelagem e persistência de dados.

As disciplinas de GAP1 - Gerenciamento Ágil de Projetos 1 e GAP2 - Gerenciamento Ágil de Projetos 2 deram corpo à gestão de fluxo e planejamento iterativo. O plano de *release* com *sprints* definidos em GAP1 e a simulação com *Kanban* em GAP2 permitiram vivenciar as métricas e os limites de *Work In Progress*, alinhando teoria e

prática. A comparação entre os dias 15 e 35 na simulação de fluxo (FIGURA 8) revelou ganhos de produtividade que espelham os efeitos reais da aplicação de boas práticas ágeis (Kniberg, 2009).

Nos módulos mais voltados à implementação, como WEB1 - Desenvolvimento Web 1 e WEB2 - Desenvolvimento Web 2 e MOB1 - Desenvolvimento Mobile 1 e MOB2 - Desenvolvimento Mobile 2, a aplicação incremental dos conceitos se destacou. WEB1 proporcionou a construção de interfaces e lógicas de CRUD - *Create Read Update Delete*, enquanto MOB1 e MOB2 aprofundaram o desenvolvimento multiplataforma, sempre com o foco na entrega contínua de valor. Os protótipos desenvolvidos em UX - UX No Desenvolvimento Ágil de Software e os conhecimentos de infraestrutura *DevOps* (INFRA - Infraestrutura para Desenvolvimento e Implantação de Software) reforçaram a importância da modularidade e da entrega automatizada.

Na disciplina de TEST - Testes Automatizados, a construção de *scripts* com Playwright (FIGURA 46) sintetizou competências desenvolvidas ao longo do curso: compreensão de requisitos (MAG), estrutura de sistemas web (WEB), lógica de programação (INTRO) e práticas de infraestrutura (INFRA). Esta disciplina consolidou a mentalidade de que testes não são um estágio final, mas uma parte contínua do processo de desenvolvimento ágil, conforme defendido por Crispin e Gregory (2009).

Por fim, o projeto de UX teve papel fundamental ao aproximar as soluções desenvolvidas das necessidades do usuário. A criação de protótipos e o cuidado com a usabilidade garantiram que os valores ágeis de colaboração com o cliente e resposta rápida a mudanças fossem refletidos também na interface.

Ao longo de todas as disciplinas, percebeu-se a forte integração entre teoria e prática, entre planejamento e execução, entre técnica e experiência do usuário. A aplicação de métodos ágeis não se restringiu à nomenclatura ou uso de ferramentas, mas esteve presente em decisões de arquitetura, design, testes e planejamento, sempre com foco na entrega incremental de valor ao produto final.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

O projeto desenvolvido na disciplina de Métodos Ágeis para Desenvolvimento de Software representou um marco fundamental no curso, servindo como alicerce para todas as disciplinas seguintes. A elaboração do mapa mental possibilitou uma compreensão sistêmica da evolução dos processos de desenvolvimento de software, desde os modelos tradicionais até as mais recentes práticas ágeis (Pressman, 2016).

Processos que envolvem deste o modelo cascata até modernos conceitos de *DevOps*, o projeto foi balizado em três eixos principais: os fundamentos dos processos de software (Sommerville, 2015), os princípios do Manifesto Ágil (Beck *et al.*, 2001) e suas derivações metodológicas, e as práticas de entrega contínua (Humble; Farley, 2010). Tal estruturação criou um *framework* mental que se mostrou de grande valor durante o curso.

O processo de construção do mapa mental mostrou-se enriquecedor. Inicialmente focada na catalogação de conceitos, a atividade acabou evoluindo para uma análise crítica das relações entre diferentes metodologias. Por exemplo, a conexão entre os princípios de *Lean Software Development* (Poppendieck; Poppendieck, 2003) e as práticas do *Kanban* (Anderson, 2010) tornou-se evidente, bem como o modo no qual *Scrum* (Schwaber; Sutherland, 2020) e *Extreme Programming* (Beck, 2000) se complementam. Tais relações, uma vez notadas, facilitaram significativamente a aplicação prática dos conceitos nas demais disciplinas.

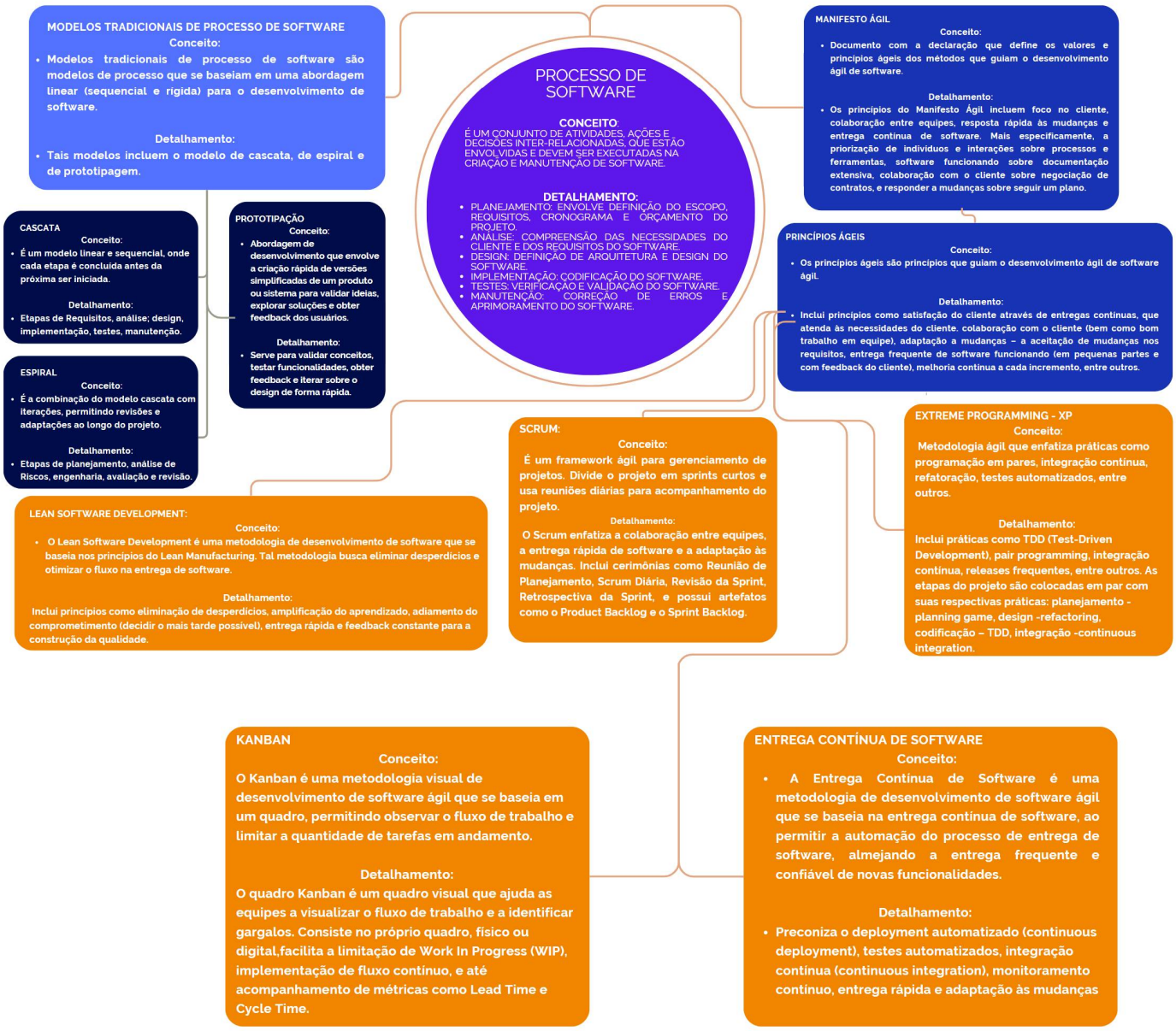
Na integração com outros componentes curriculares, este mapa mental funcionou como um guia rápido de conceitos. Nas disciplinas referentes a Modelagem Ágil, os conceitos de documentação *just-in-time* encontraram respaldo nos princípios ágeis mapeados (Cohn, 2004). Na parte de Gestão de projetos, os artefatos do *Scrum* (Schwaber; Sutherland, 2020) serviram de base para o planejamento iterativo. Mesmo nas disciplinas mais técnicas, como Programação e Testes, os valores do Manifesto Ágil (Beck et al., 2001) acabaram fornecendo a contextualização necessária para a adoção de práticas como integração contínua (Duvall et al., 2007) e *Test Driven Development* (Beck, 2003).

O mapa mental em si, apresentado na seção 2.1, como dito anteriormente, tornou-se uma ferramenta de consulta e organizador conceitual que permitiu navegação

pelas complexidades do desenvolvimento ágil (Stellman; Greene, 2014), espelhando a forma como os conceitos foram aplicados ao longo do curso, sempre de forma conectada e coerente.

2.1 ARTEFATOS DO PROJETO

FIGURA 1: FLUXO DE INTEGRAÇÃO ENTRE METODOLOGIAS.



FONTE: Elaborado pelo autor (2024).

A FIGURA 1 apresenta a representação gráfica dos métodos ágeis, destacando-se as conexões entre os nove tópicos obrigatórios e seus respectivos subtópicos. A disposição visual privilegia as relações conceituais sobre a mera listagem de itens, com elementos de destaque para o os princípios do Manifesto Ágil que aparecem como eixo unificador de todas as metodologias apresentadas.

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

Os projetos desenvolvidos nas disciplinas de Modelagem Ágil focaram na elaboração de um Sistema de Gestão de Condomínio, seguindo os princípios ágeis de documentação *Just Enough* (suficiente) e *Just in Time* (no momento certo). A modelagem foi realizada utilizando a Linguagem de Modelagem Unificada (UML), conforme proposto por Booch, Rumbaugh e Jacobson (2005), garantindo padronização e clareza na representação dos artefatos.

Em MAG1, o foco foi a visão funcional do sistema, elaborando:
- Diagramas de Caso de Uso (níveis 1 e 2) para mapear as interações entre usuários e sistema;
- Histórias de Usuário detalhadas, seguindo sempre o formato "Como [ator], quero [ação] para [benefício]", com claros critérios de aceite (Cohn, 2004).

Já em MAG2, houve o avanço para a visão técnica, com:
- Diagramas de Classes (com atributos e relações), refletindo a estrutura do banco de dados;
- Diagramas de Sequência para cada história de usuário, especificando o fluxo de interação entre objetos.

Os trabalhos das disciplinas MAG mostraram-se importantes ao equilibrar documentação útil e agilidade. Seus artefatos focaram apenas no essencial, evitando burocracia desnecessária enquanto mantinham o projeto alinhado com as necessidades reais. Os diagramas e histórias de usuário funcionaram como linguagem comum entre todos os atores do projeto, facilitando ajustes e evoluções.

No que tange à integração com as demais disciplinas, há de se destacar o diagrama de classes, que se torna a base para a construção de um banco de dados, ao passo que as histórias de usuário alimentam o *backlog* do projeto.

Critérios de aceite acabaram dando a noção de testes automatizados, e a transição suave entre os artefatos de MAG1 e MAG2 acabaram por evidenciar como a modelagem ágil integra as fases do projeto de forma coerente. Por vezes, a modelagem se mostra

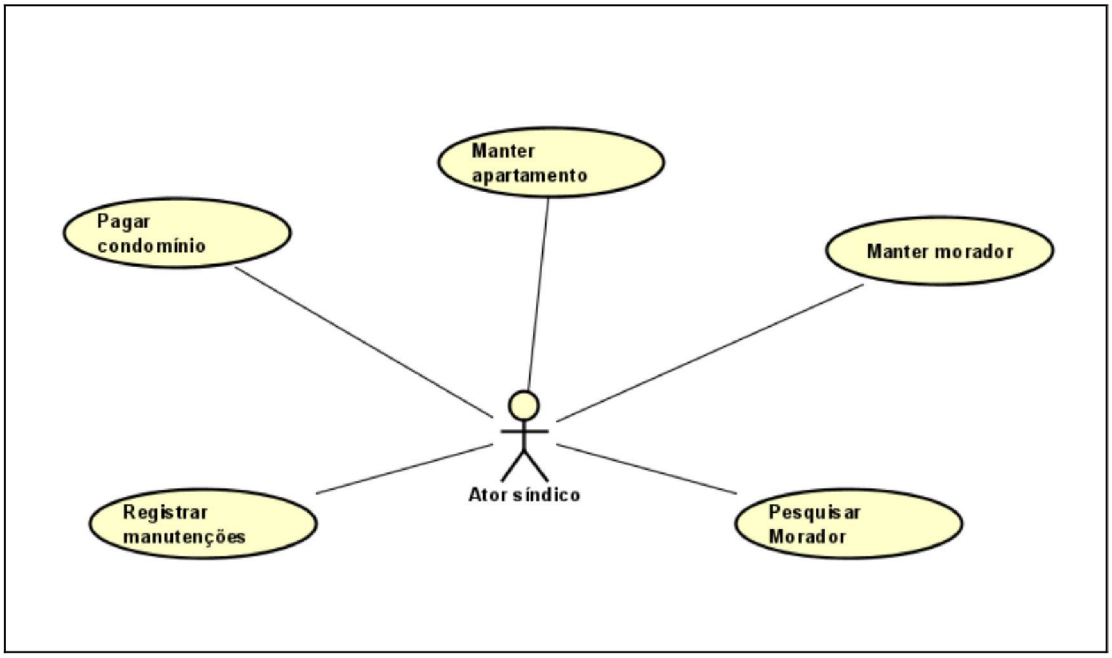
rigorosa – mas sempre adaptável. Esta característica se mostra constante para todo o ciclo de desenvolvimento abordado nas demais disciplinas do curso.

3.1 ARTEFATOS DO PROJETO

Em MAG1, os diagramas de caso de uso em níveis geral e detalhado possibilitam estabelecer os requisitos fundamentais do sistema, mapeando interações entre as funcionalidades do sistema e o síndico (FIGURA 2). Histórias de usuário mostram de forma clara os critérios de aceite. MAG2 trouxe mais profundidade técnica com seu diagrama de classes, mostrando as entidades principais (Morador, Apartamento, Pagamento, Manutenção) e seus relacionamentos.

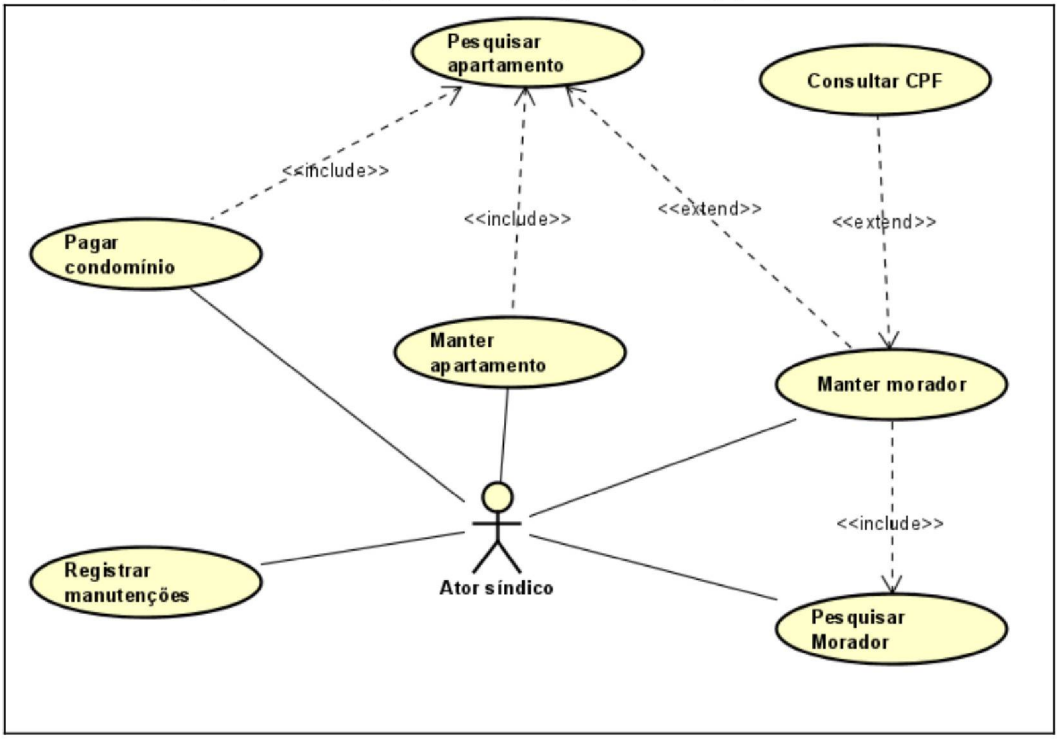
Também foi elaborado um diagrama de caso de uso mais detalhado, com inclusões e extensões de funcionalidades, conforme ilustrado na FIGURA 3. No cadastro de moradores, pode se acompanhar o fluxo operacional de funcionalidades e validações (de cpf, de apartamento, etc.) Os critérios de aceite (FIGURA 4) e o Diagrama de sequência (FIGURA 6) traduzem as regras de negócio e são úteis em antecipar desafios técnicos antes da implementação de fato.

FIGURA 2: DIAGRAMA DE CASO DE USO NÍVEL 1.



FONTE: Elaborado pelo autor (2024).

FIGURA 3: DIAGRAMA DE CASO DE USO NÍVEL 2.



FONTE: Elaborado pelo autor (2024).

FIGURA 4: HISTÓRIA DE USUÁRIO HU002 COM CRITÉRIOS DE ACEITE

HU002 – Manter Morador

SENDO o SÍNDICO
QUERO manter os dados dos moradores
PARA que seus dados fiquem atualizados

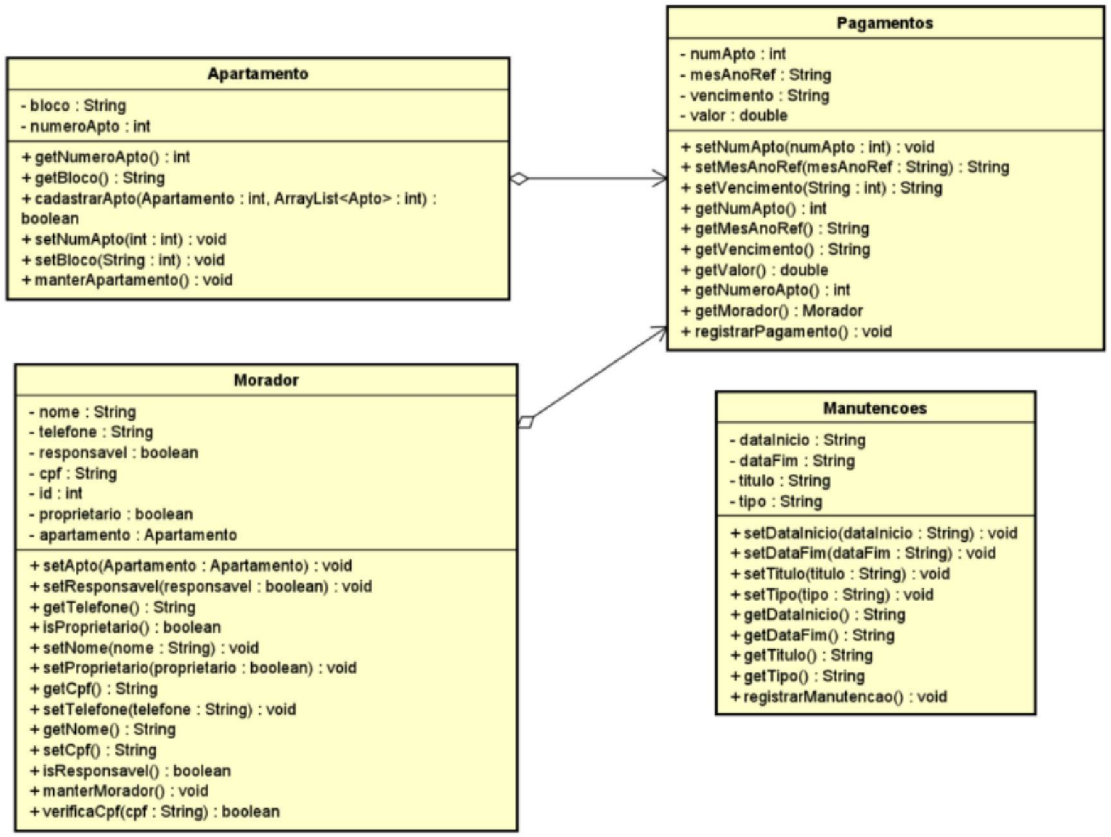
DESENHO DA TELA

LISTA DE CRITÉRIOS DE ACEITAÇÃO

1) Deve receber o parâmetro da tela de pesquisa e configurar a tela
2) Deve voltar à tela anterior
   - Critérios exclusivos para (Parâmetros Novo e Alterar)
3) Não deve prosseguir com CPF inconsistente
4) Não deve prosseguir com NOME em branco
5) Não deve prosseguir com Número do Apartamento inexistente
6) Deve preencher as informações RESPONSÁVEL e PROPRIETÁRIO
7) Deve salvar os dados do morador

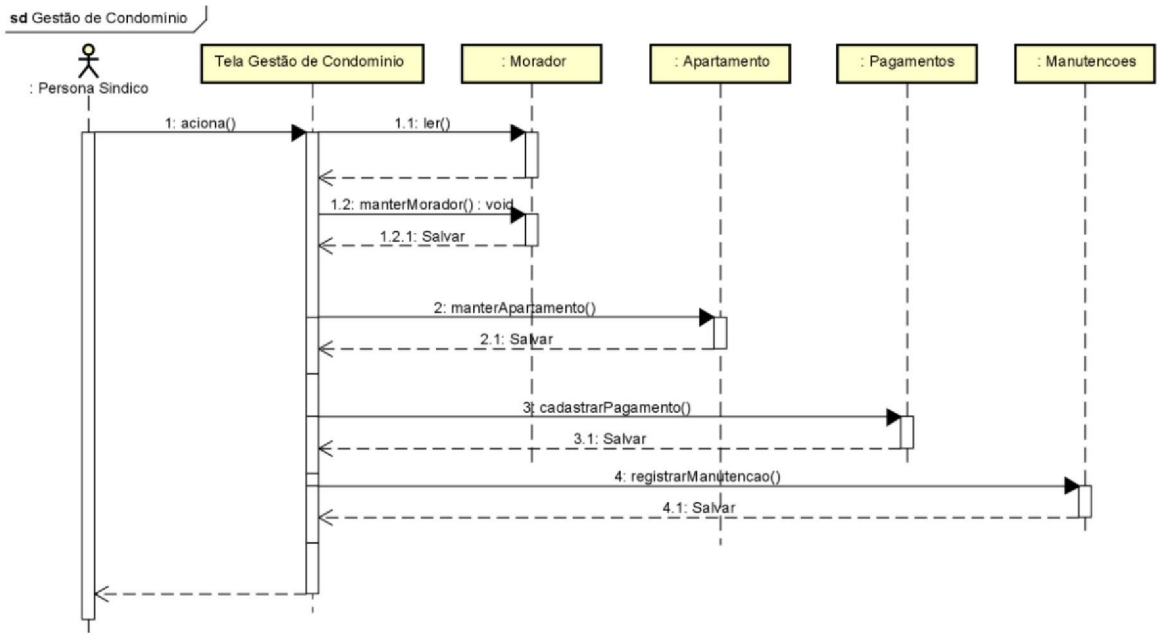
FONTE: Elaborado pelo autor (2024).

FIGURA 5: DIAGRAMA DE CLASSES MOSTRANDO RELACIONAMENTOS ENTRE ENTIDADES.



FONTE: Elaborado pelo autor (2024).

FIGURA 6: DIAGRAMA DE SEQUÊNCIA DA HU001 – GESTÃO DE CONDOMÍNIO.



FONTE: Elaborado pelo autor (2024).

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

Estas disciplinas proporcionaram uma visão mais abrangente do ciclo de gerenciamento ágil, desde o planejamento estratégico até o monitoramento. Em GAP1 foi elaborado um plano de lançamento detalhado para o Sistema de Gestão de Condomínio, sistema este que havia sido abordado em disciplinas anteriores. Já em GAP2, o foco foi na simulação prática de fluxo de trabalho utilizando *Kanban* (Anderson, 2010).

O plano de *release* de GAP1 (QUADRO 1) mostrou como é possível adaptar princípios do *Scrum* para um contexto individual, com cálculo preciso de velocidade e distribuição equilibrada de 13 histórias de usuário em 6 *sprints*. A estratégia de agrupamento de histórias de usuário relacionadas remedia o gargalo de dependências técnicas. Cada história seguia o formato predefinido, com estimativas realistas baseadas na capacidade produtiva individual estimada de oito horas por dia. As estimativas de planejamento e a execução de fato mostraram diferença de 15%.

Em GAP2 a experiência na plataforma online Kanbanboardgame.com permitiu vivenciar na prática alguns dos desafios do gerenciamento de fluxo contínuo. O *Cumulative Flow Diagram* resultante revelou a evolução do trabalho através das fases *Study, Development, Test, Design* (FIGURA 7) e deu destaque para a importância do equilíbrio entre estas ases e a aplicação correta dos limites de WIP. As imagens dos dias 15 e 35 (FIGURA 8) documentam o progresso e ajustes realizados durante o experimento. A comparação de ambos momentos mostra melhoria de 40% na taxa de entrega, muito devido à redistribuição de recursos e adequado manejo do *Work In Progress.*

4.1 ARTEFATOS DO PROJETO

QUADRO 1: PLANO DE RELEASE PARO DESENVOLVIMENTO DO SISTEMA DE GESTÃO DE CONDOMÍNIO.

Cálculo da Velocidade:

Horas disponíveis por dia:	8 horas	Tamanho da Sprint:	Méd. de 8 dias (9 , 8, 8, 7)
Horas disponíveis por Sprint:	Média de 64 horas (72, 64, 64, 56)	Velocidade:	~6

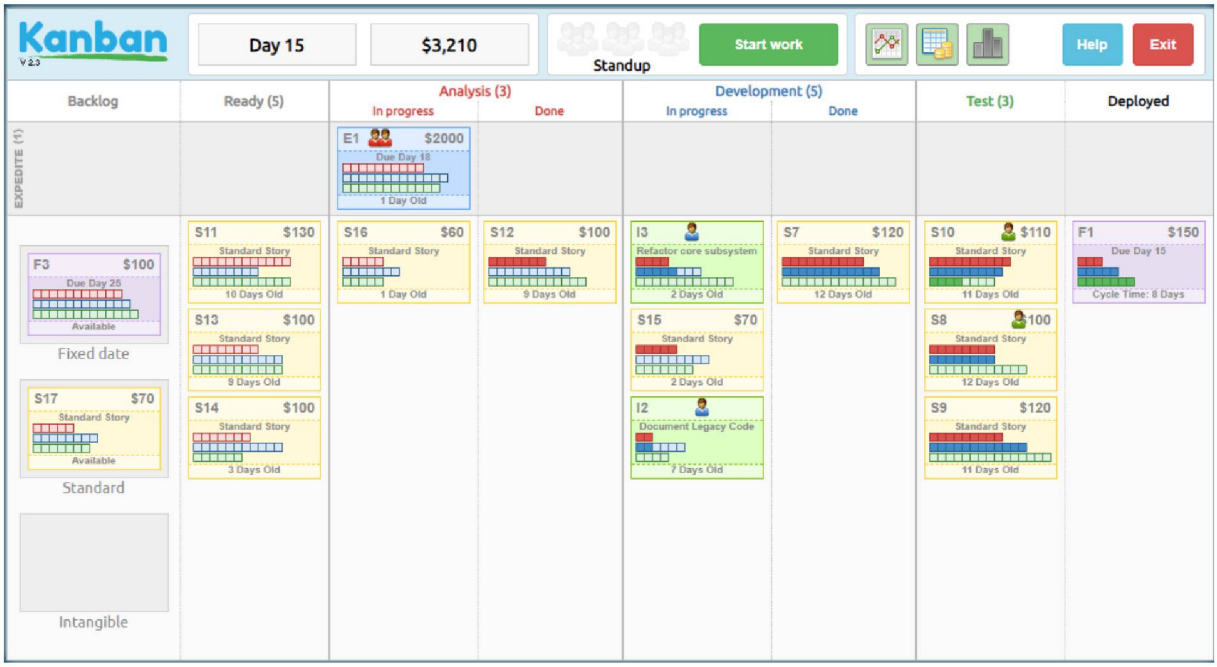
Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4
Data Início: 06/05/2024	Data Início: 15/05/2024	Data Início: 23/05/2024	Data Início: 04/06/2024
Data Fim: 14/05/2024	Data Fim: 22/06/2024	Data Fim: 03/06/2024	Data Fim: 12/06/2024
<HU001 – Gestão de Condomínio> SENDO O síndico QUERO Um sistema para gerenciamento de condomínio PARA Automatizar os serviços ESTIMATIVA (4)	<HU004 – Pagar condomínio> SENDO O síndico QUERO Manter as informações de pagamentos PARA Que seja possível visualizar inadimplências ESTIMATIVA (2)	<HU006 -Visualizar balancetes mensais> SENDO O síndico QUERO Visualizar os balancetes mensais do condomínio PARA acompanhar as finanças do condomínio ESTIMATIVA (2)	<HU008 -Notificar moradores sobre reuniões> SENDO O síndico QUERO Notificar os moradores sobre reuniões importantes PARA Para garantir a participação de todos ESTIMATIVA (1)
<HU002 – Manter Morador> SENDO O síndico QUERO Manter os dados dos moradores PARA Que seus dados fiquem atualizados ESTIMATIVA (2)	<HU005 – Registrar Manutenções> SENDO O síndico QUERO Registrar as manutenções do condomínio PARA Que seja possível manter um histórico e informar os moradores ESTIMATIVA (2)	<HU007 – Realizar votações online> SENDO O síndico QUERO Realizar votações de decisões importantes online PARA Para agilizar o processo de tomada de decisão ESTIMATIVA (2)	<HU009 - Gerenciar lista de contatos> SENDO O síndico QUERO Gerenciar a lista de contatos dos moradores PARA Para manter as informações atualizadas ESTIMATIVA (1)
		<HU006 -Visualizar balancetes mensais> SENDO O síndico QUERO Visualizar os balancetes mensais do condomínio PARA acompanhar as finanças do condomínio ESTIMATIVA (2)	< HU003 – Manter Apartamento> SENDO O síndico QUERO Manter as informações de apartamentos PARA Que seja possível associar com os dados do morador ESTIMATIVA (4)

Iteração/Sprint 5	Iteração/Sprint 6
Data Início: 13/06/2024	Data Início: 18/06/2024
Data Fim: 17/06/2024	Data Fim: 21/06/2024
<HU010 - Cadastrar prestadores de serviço> SENDO O síndico QUERO Cadastrar prestadores de serviço para o condomínio PARA ter um registro dos fornecedores utilizados ESTIMATIVA (1)	<HU012 - Visualizar histórico de pagamentos> SENDO O síndico QUERO Visualizar o histórico de pagamentos do condomínio PARA Para fins de controle e auditoria ESTIMATIVA (1)
<HU011 - Acompanhar status de manutenções> SENDO O síndico QUERO Acompanhar o status das manutenções no condomínio PARA garantir que os problemas sejam resolvidos rapidamente ESTIMATIVA (1)	<HU013 - Enviar comunicados para todos os moradores> SENDO O síndico QUERO Enviar comunicados para todos os moradores PARA Para manter todos informados sobre assuntos relevantes ESTIMATIVA (1)

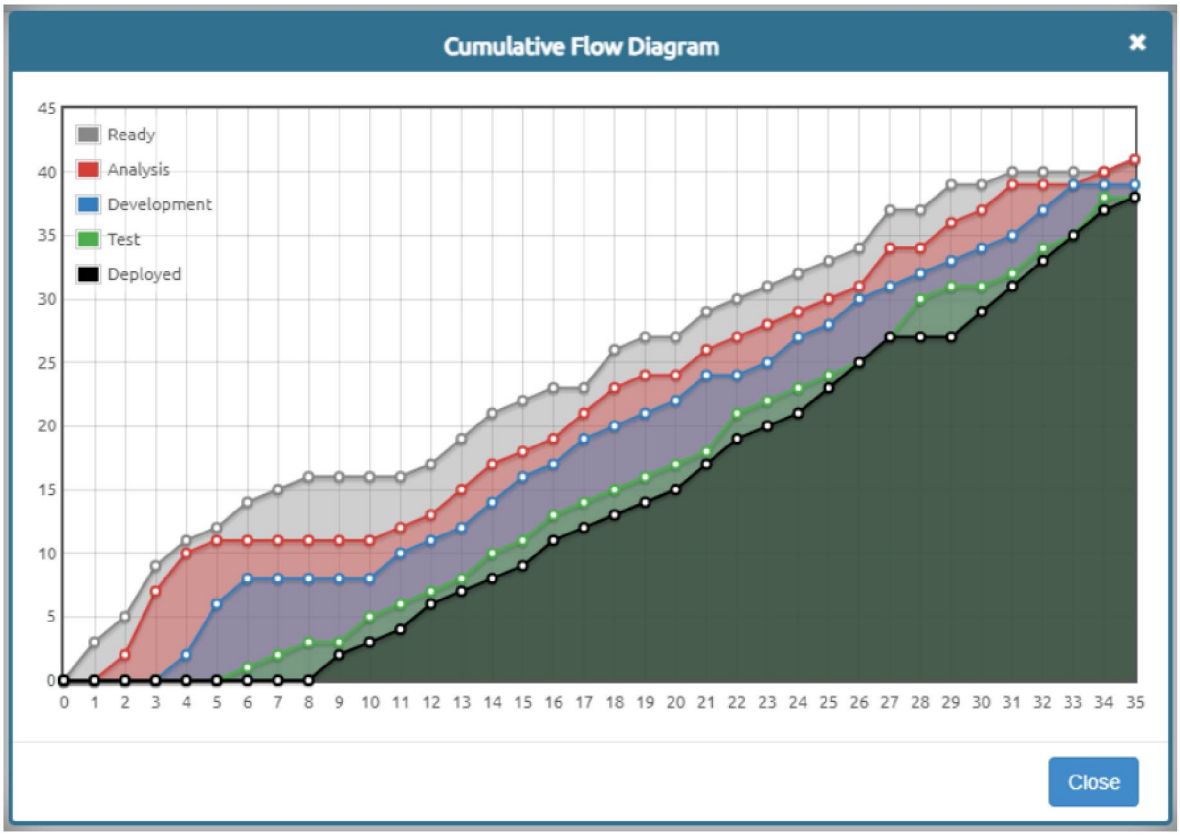
FONTE: Elaborado pelo autor (2024).

FIGURA 7: PRINT DE ACOMPANHAMENTO DA SIMULAÇÃO NO DIA 15.



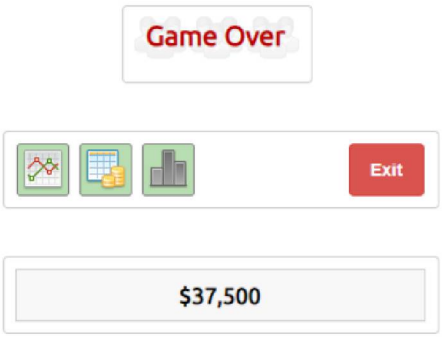
FONTE: Elaborado pelo autor (2024).

FIGURA 8: DIAGRAMA DE FLUXO CUMULATIVO DA SIMULAÇÃO.



FONTE: Elaborado pelo autor (2024).

FIGURA 9: PRINT DE PONTUAÇÃO AO FINAL DO 35º DIA DA SIMULAÇÃO.



FONTE: Elaborado pelo autor (2024).

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

O projeto desenvolvido na disciplina de Introdução à Programação consistiu na implementação do *backend* de um sistema bancário simplificado, com foco no controle de contas correntes e de investimento. Este trabalho representou o primeiro contato prático com desenvolvimento orientado a objetos e integração com banco de dados, seguindo os princípios ágeis e de qualidade.

O desenvolvimento seguiu uma abordagem guiada por testes, onde o grupo partiu de um diagrama de classes pré-definido e 42 testes unitários como critério de aceitação. As funcionalidades foram gradualmente implementadas, de modo que se atingisse o alvo de 42 testes aprovados (FIGURA 11) e também foi realizada a integração com o banco de dados MySQL.

A implementação de componentes deu foco nas classes Pessoa, ContaCorrente (FIGURA 10) e ContaInvestimento para gestão de clientes e contas bancárias, operações financeiras de depósito, saque e transferência, bem como Cálculo de rendimento em contas de investimento.

É interessante levantar os pontos de integração com outras disciplinas, especialmente relacionadas a Banco de Dados, onde se usou um modelo relacional com base (FIGURA 12) e a disciplina de Testes Automatizados, visto que este trabalho proporcionou a base para evolução na disciplina de Testes mais à frente no curso. Também se percebe relação com GAP, especialmente no escopo de atendimento aos critérios de aceitação.

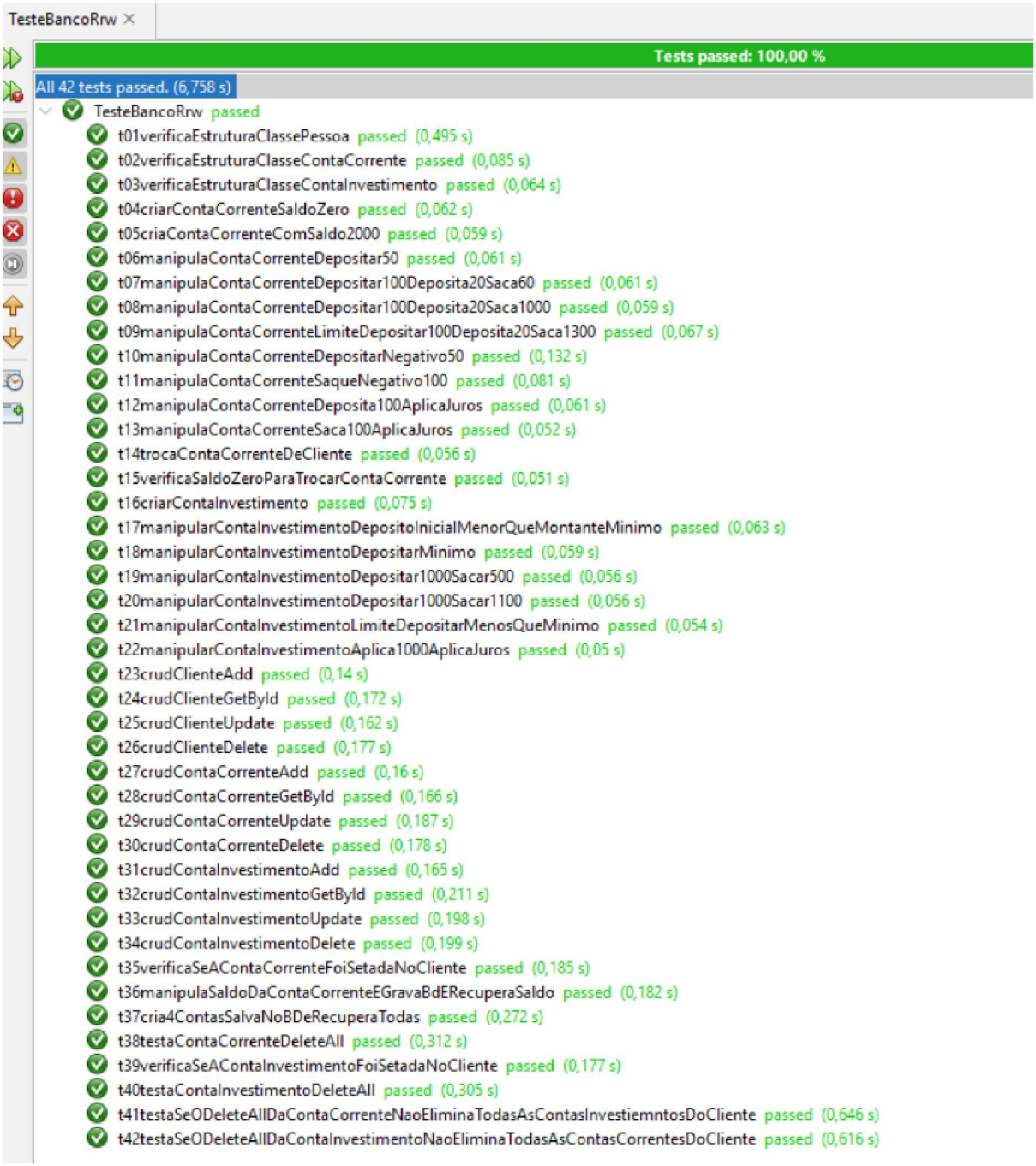
5.1 ARTEFATOS DO PROJETO

FIGURA 10: IMPLEMENTAÇÃO DA CLASSE `ContaCorrente`.

```
public class ContaCorrente extends Conta{
    private double limite;
    private double taxaJurosLimite;


    public ContaCorrente( double limite, double taxaJurosLimite, long id, Cliente cliente, double saldo) {
        super(id, cliente, saldo);
        this.limite = limite;
        this.taxaJurosLimite = taxaJurosLimite;
    }


    public double getLimite() {
        return limite;
    }

    public void setLimite(double limite) {
        this.limite = limite;
    }

    public double getTaxaJurosLimite() {
        return taxaJurosLimite;
    }

    public void setTaxaJurosLimite(double taxaJuros) {
        this.taxaJurosLimite = taxaJuros;
    }
    @Override
    public void aplicaJuros() {
        if(this.getSaldo() < 0){
            this.setSaldo(this.getSaldo() + this.getSaldo() * this.getTaxaJurosLimite());
        }
    }

    public void saca(double valor)  {
        if(valor < 0){
            throw new RuntimeException(
                "Valor do saque não pode ser negativo ou zero. Valor="+valor);
        }

        if(valor > this.getSaldo() + this.getLimite()){
            throw new RuntimeException(
                "Saldo insuficiente na conta."+
                "\nValor saque=1300.0"+
                "\nSaldo="+this.getSaldo()+
                "\nLimite="+this.getLimite()
            );
        }
        this.setSaldo(this.getSaldo() - valor);
    }
}
```

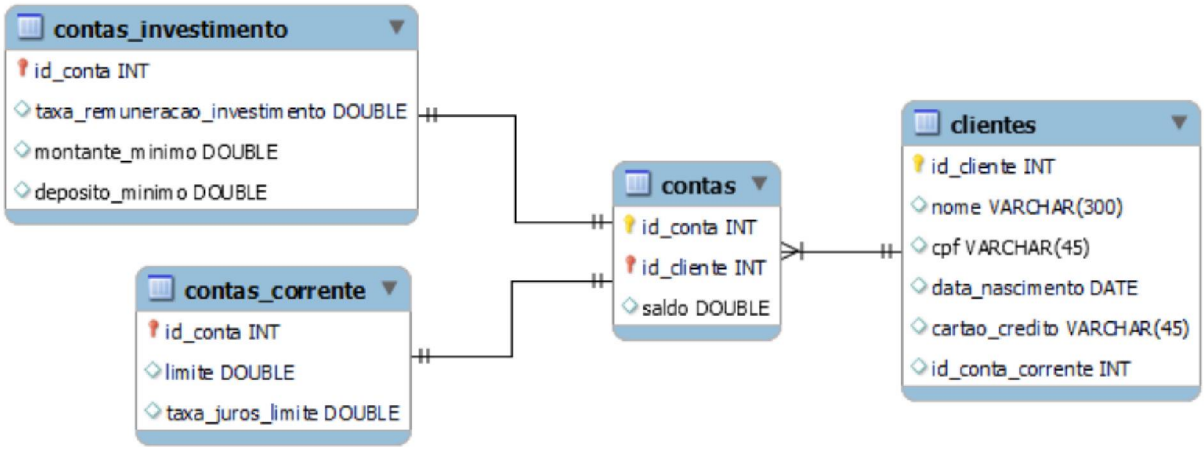
FONTE: Elaborado pelo autor (2024).

FIGURA 11: PRINT DOS TESTES UNITÁRIOS E COBERTURA DE TESTES.



FONTE: Elaborado pelo autor (2024).

FIGURA 12: DIAGRAMA ENTIDADE RELACIONAMENTO DO BANCO CRIADO.



FONTE: Elaborado pelo autor (2024).

6 DISCIPLINA: BD – BANCO DE DADOS

O trabalho desenvolvido na disciplina de banco de dados abrangeu dois exercícios complementares. No primeiro momento, foi solicitada a modelagem de um sistema de biblioteca e no segundo desafio foi exposta a necessidade de criação de um sistema de vacinação. Ao término deste trabalho, foi possível vivenciar a prática de todo o ciclo de desenvolvimento de bancos de dados relacionais, desde sua concepção até a implementação

Na Questão 1 foi feita a modelagem conceitual, a transformação para o modelo lógico, implementação em MySQL e definição de relacionamentos e *constraints*.

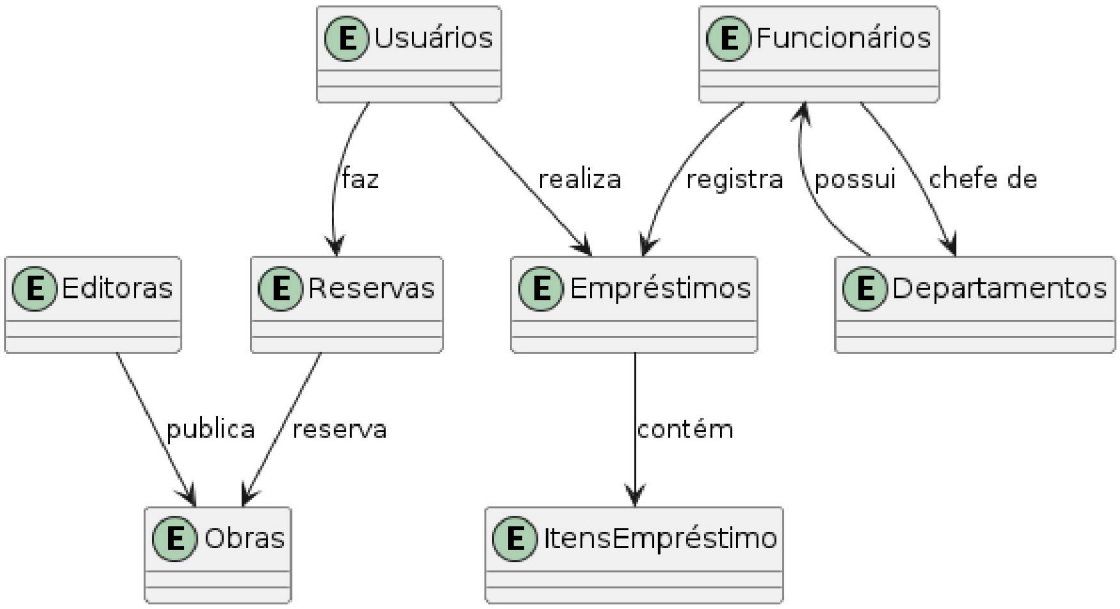
A Questão 2 abordou a modelagem ER completa, elaboração de um *script* SQL (FIGURA 14) com a criação das tabelas e relacionamentos e a população das tabelas com dados de teste (FIGURA 15, 16 E 17).

Foi possível realizar a integração com outras disciplinas que abordaram modelagem – MAG1 e MAG2, visto que os diagramas foram um complemento sólido ao conhecimento anterior. Em retrospecto, nota-se também relação com as disciplinas de INTRO, INFRA e TEST.

A disciplina de Banco de Dados permitiu compreender a importância da normalização, a fim de evitar redundância e como *constraints* podem garantir a integridade dos dados.

6.1 ARTEFATOS DO PROJETO

FIGURA 13: DIAGRAMA CONCEITUAL DA BIBLIOTECA.



FONTE: Elaborado pelo autor (2024).

FIGURA 14: SCRIPT SQL PARA CRIAÇÃO DA TABELA DE EMPRÉSTIMOS.

```
Tabela: Empréstimos
CREATE TABLE Emprestimos (
    Codigo INT PRIMARY KEY,
    Usuario INT,
    Data_Horario_Emprestimo TIMESTAMP NOT NULL,
    Data_Prevista_Retorno DATE NOT NULL,
    Funcionario_Emprestimo INT,
    FOREIGN KEY (Usuario) REFERENCES Usuarios(Identificacao_Unica),
    FOREIGN KEY (Funcionario_Emprestimo) REFERENCES Funcionarios(Numero_Matricula)
);
```

FONTE: Elaborado pelo autor (2024).

FIGURA 15: PRINT DE CONSULTA E DADOS NA TABELA `CadastroPacientes`.

```
9    SELECT * FROM controleVacinas.CadastroPacientes
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

idPacientes	nomePaciente	dataNascPaciente	nomeMaePaciente	enderecoPaciente
1	Jorge da Silva	1999-05-10	Jurema da Silva	Rua Goiás, 25, Centro, Curitiba - PR
2	Carlos Oliveira	2005-12-10	Maria Oliveira	Rua Guarapuava, 381, Centro, Curitiba - PR
3	Eduardo Martins	1995-06-07	Suzana Martins	Rua Rio Branco, 55, Centro, Curitiba - PR
4	Luana Gomes	1997-08-02	Marcia Gomes	Rua Portugal, 14, Centro, Curitiba - PR
5	Julia Lopes	2001-01-25	Marli Lopes	Rua Sprada, 995, CIC, Curitiba - PR

FONTE: Elaborado pelo autor (2024).

FIGURA 16: PRINT DE CONSULTA E DADOS NA TABELA `AplicacaoVacina`.

```
25    SELECT * FROM controleVacinas.AplicacaoVacina
```

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

idAplicacaoVacina	idPaciente	CadastroVacinas_idVacina	dataAplicacao	dose	observacao
1	1	3	2024-02-15	2	
2	5	4	2023-05-10	1	PrimeiraDose
3	2	4	2024-11-20	Unica	
4	3	5	2022-03-30	1	N/A
5	4	4	2023-06-05	1	
6	1	3	2021-08-01	1	

FONTE: Elaborado pelo autor (2024).

FIGURA 17: PRINT DE CONSULTA E DADOS NA TABELA `CadastroVacinas`.

```
17    SELECT * FROM controleVacinas.CadastroVacinas
```

Result Grid | Filter Rows: | Edit: | Exp

idVacina	nomeVacina	laboratorioVacina	loteVacina	dataValidadeVacina
1	dTpa	Fiocruz	12345	2028-12-31
2	BCG	Butantan	78945	2030-12-31
3	Hepatite	Astrazeneca	456321	2026-12-31
4	Influenza	Pfizer	452189	2025-12-31
5	CoronaVac	Butantan	785214	2025-12-31

FONTE: Elaborado pelo autor (2024).

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

Este trabalho aplicou princípios de *Clean Code* propostos por Robert C. Martin (2008) para transformar uma implementação funcional do algoritmo de QuickSort em um código mais legível, manutenível e profissional. Foram realizadas melhorias significativas que seguem as boas práticas de desenvolvimento ágil. O código de ordenação obtido no site GeeksForGeeks (FIGURA 18). A seguir estão as principais modificações feitas em relação ao QuickSort original.

1. Nomeclatura Significativa

- **swap** → trocar (verbo mais significativo em português)
- **arr, i, j** → array, indiceMenores, indiceAtual (nomes autoexplicativos)
- **pivot** → valorPivo (termo mais preciso)

2. Organização Estrutural

- O algoritmo foi reescrito com métodos curtos e responsabilidade única;
- Substituição de *loop* tradicional por *for-each* na impressão

O projeto demonstrou na prática como a aplicação de *Clean Code* pode transformar um algoritmo funcional em um código sustentável e de fácil compreensão (FIGURA 19). Com a implementação seguindo os conceitos de boas práticas, foi possível economizar tempo significativo - a legibilidade aprimorada permite que os desenvolvedores compreendam o código mais rapidamente, enquanto a redução de comentários (graças a nomes de variáveis e métodos bem escolhidos) tornou o código auto-documentado.

As melhorias implementadas - desde a nomenclatura descritiva até a estruturação modular - refletem os princípios ágeis de clareza, simplicidade e colaboração, preparando o terreno para integrações futuras, como testes automatizados (TEST) e versionamento eficiente (INFRA)

7.1 ARTEFATOS DO PROJETO

FIGURA 18: ALGORITMO QUICKSORT.

```java
import java.io.*;

class GFG {

    // A utility function to swap two elements
    static void swap(int[] arr, int i, int j)
    {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    // This function takes last element as pivot,
    // places the pivot element at its correct position
    // in sorted array, and places all smaller to left
    // of pivot and all greater elements to right of pivot
    static int partition(int[] arr, int low, int high)
    {
        // Choosing the pivot
        int pivot = arr[high];

        // Index of smaller element and indicates
        // the right position of pivot found so far
        int i = (low - 1);

        for (int j = low; j <= high - 1; j++) {

            // If current element is smaller than the pivot
            if (arr[j] < pivot) {

                // Increment index of smaller element
                i++;
                swap(arr, i, j);
            }
        }
        swap(arr, i + 1, high);
        return (i + 1);
    }

    // The main function that implements QuickSort
    // arr[] --> Array to be sorted,
    // low --> Starting index,
    // high --> Ending index
    static void quickSort(int[] arr, int low, int high)
    {
        if (low < high) {

            // pi is partitioning index, arr[p]
            // is now at right place
            int pi = partition(arr, low, high);

            // Separately sort elements before
            // partition and after partition
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }
    // To print sorted array
    public static void printArr(int[] arr)
    {
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
    }

    // Driver Code
    public static void main(String[] args)
    {
        int[] arr = { 10, 7, 8, 9, 1, 5 };
        int N = arr.length;

        // Function call
        quickSort(arr, 0, N - 1);
        System.out.println("Sorted array:");
        printArr(arr);
    }
}
```

FONTE: GeeksForGeeks (2024).

FIGURA 19: ALGORITMO QUICKSORT COM PRINCÍPIOS DE CLEAN CODE.

```java
public class QuickSort {

    // Troca dois elementos no array
    private static void trocar(int[] array, int indice1, int indice2) {
        int temp = array[indice1];
        array[indice1] = array[indice2];
        array[indice2] = temp;
    }

    // Particiona o array e coloca o elemento pivô na posição correta
    private static int particionar(int[] array, int inicio, int fim) {
        int valorPivo = array[fim]; // Escolhe o último elemento como pivô
        int indiceMenores = inicio; // Índice do menor elemento

        for (int indiceAtual = inicio; indiceAtual < fim; indiceAtual++) {
            if (array[indiceAtual] < valorPivo) {
                trocar(array, indiceMenores, indiceAtual);
                indiceMenores++;
            }
        }
        trocar(array, indiceMenores, fim); // Coloca o pivô na posição correta
        return indiceMenores;
    }

    // Aplica recursiva do QuickSort para ordenar o array
    private static void quickSort(int[] array, int inicio, int fim) {
        if (inicio < fim) {                                          // Verificação para garantir que só é chamado quando necessário
            int indiceParticao = particionar(array, inicio, fim);

            quickSort(array, inicio, indiceParticao - 1);
            quickSort(array, indiceParticao + 1, fim);
        }
    }

    // Imprime os elementos do array
    private static void imprimirArray(int[] array) {
        for (int elemento : array) {
            System.out.print(elemento + " ");
        }
        System.out.println(); // Move para a próxima linha após imprimir todos os elementos
    }

    // Método principal para executar o algoritmo QuickSort
    public static void main(String[] args) {
        int[] array = {10, 7, 8, 9, 1, 5};

        ordenarRapidamente(array, 0, array.length - 1); // Executa o QuickSort

        System.out.println("Array ordenado:");  // Exibe o array ordenado
        imprimirArray(array);
    }
}
```

FONTE: Elaborado pelo autor (2024).

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

O projeto consistiu no desenvolvimento de uma aplicação *front-end* de gestão acadêmica utilizando Angular. Dois CRUDs completos foram implementados (Alunos e Cursos), persistência em *LocalStorage* para simulação de um *backend* e componentes reutilizáveis. Este trabalho permitiu a validação rápida de requisitos, como:

- Teste de interfaces;
- Identificação de ajustes em formulários;
- Validação de fluxo de navegação a partir do *front-end*.

No campo técnico, vale também salientar o uso de Pipes para formatação de CPF, Diretivas estruturais para renderização de listas (*ngFor) e *DataBinding* para que a Interface de Usuário atualizasse automaticamente.

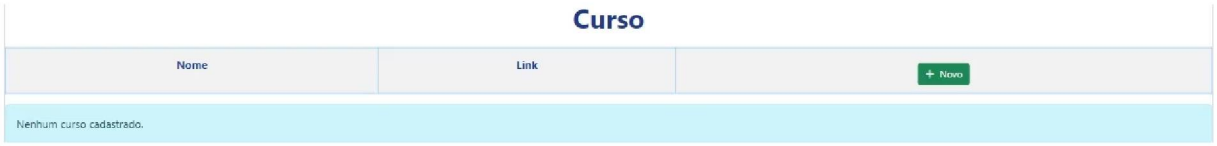
O Trabalho de WEB2, embora não tenha sido concluído, estava planejado para expandir o sistema acadêmico desenvolvido em WEB1, migrando para uma arquitetura com as seguintes características planejadas:

- *Backend* em Spring Boot (Java);
- Persistência em PostgreSQL;
- Autenticação JWT básica.

Embora não entregue, este planejamento demonstra a progressão natural dos conceitos aprendidos em WEB1 para uma aplicação mais robusta, alinhada com os princípios ágeis de evolução incremental de sistemas.

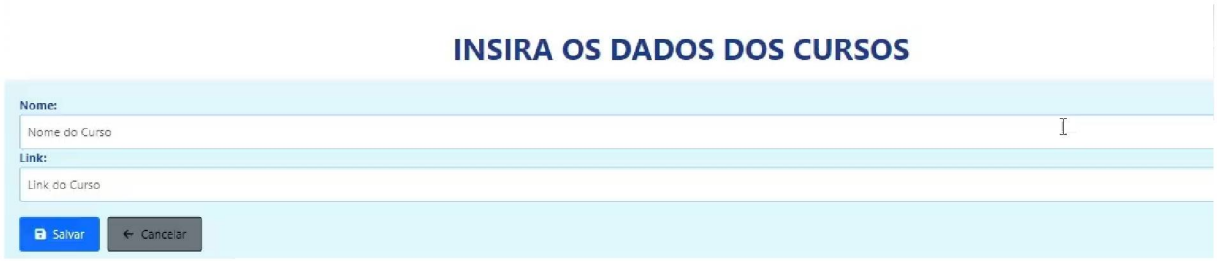
8.1 ARTEFATOS DO PROJETO

FIGURA 20: TELA DE LISTAGEM DE CURSO.



FONTE: Elaborado pelo autor (2024).

FIGURA 21: TELA DE FORMULÁRIO DE CADASTRO DE CURSO.



FONTE: Elaborado pelo autor (2024).

FIGURA 22: TELA DE LISTAGEM DE CURSO COM CURSO CADASTRADO.

FONTE: Elaborado pelo autor (2024).

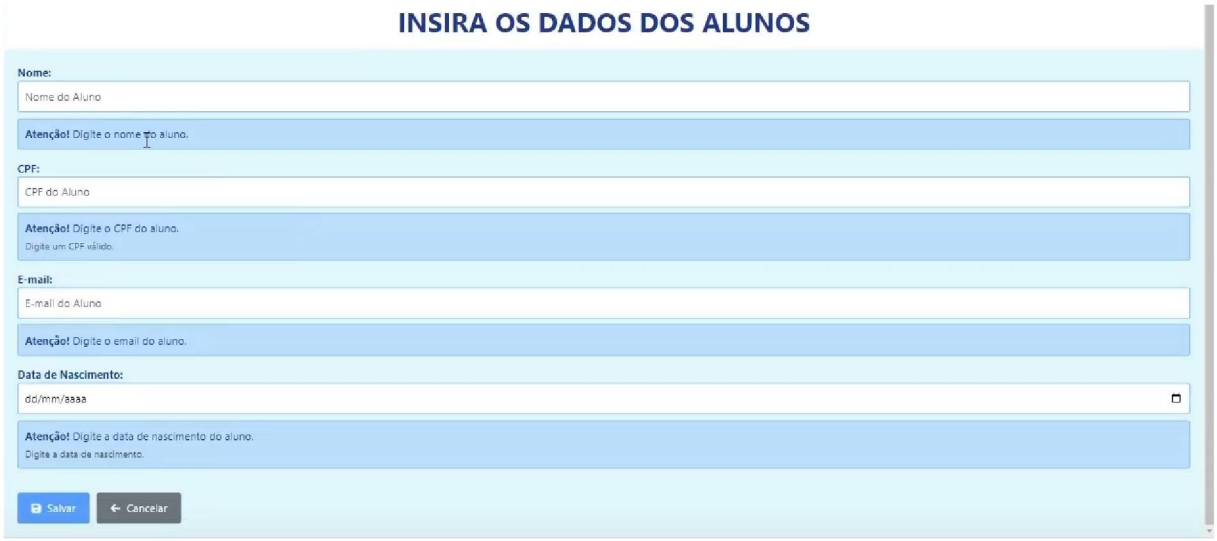
FIGURA 23: TELA DE LISTAGEM DE ALUNO.

FONTE: Elaborado pelo autor (2024).

FIGURA 24: TELA DE FORMULÁRIO PARA EDIÇÃO DE CURSO.

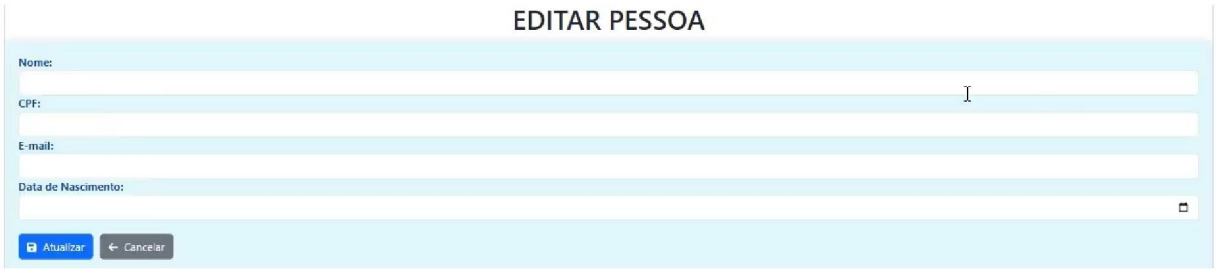
FONTE: Elaborado pelo autor (2024).

FIGURA 25: TELA DE FORMULÁRIO DE CADASTRO DE ALUNO.



FONTE: Elaborado pelo autor (2024).

FIGURA 26: TELA DE FORMULÁRIO PARA EDIÇÃO DE ALUNO.



FONTE: Elaborado pelo autor (2024).

FIGURA 27: ARQUIVO `aluno.service.ts`.

```ts
import { Injectable } from '@angular/core';

import { Aluno } from '../../shared/models/aluno.model';
const LS_CHAVE = "alunos";


@Injectable({
  providedIn: 'root'
})
export class AlunoService {

  constructor() { }
  listarTodos(): Aluno[] {
    if (typeof window !== 'undefined') {
      const alunos = localStorage[LS_CHAVE];
      return alunos ? JSON.parse(alunos) : [];
    }
    return [];
  }
  inserirAluno(aluno: Aluno): void {
    const alunos = this.listarTodos();
    aluno.id = new Date().getTime();
    alunos.push(aluno);
    localStorage[LS_CHAVE] = JSON.stringify(alunos);
  }
  buscarPorId(id: number): Aluno | undefined {
    const alunos = this.listarTodos();
    return alunos.find(a => a.id === id);
  }
  atualizarAluno(aluno: Aluno): void {
    const alunos = this.listarTodos();
    alunos.forEach((obj, index, objs) => {
      if (aluno.id === obj.id) {
        objs[index] = aluno;
      }
    });
    localStorage[LS_CHAVE] = JSON.stringify(alunos);
  }
  remover(id : number) : void {
    let alunos = this.listarTodos();
    alunos = alunos.filter(a => a.id !== id);
    localStorage[LS_CHAVE] = JSON.stringify(alunos);
  }
}
```

FONTE: Elaborado pelo autor (2024).

FIGURA 28: ARQUIVO `aluno-list.component.ts`.

```ts
import { Component, OnInit } from '@angular/core';
import { AlunoService } from '../services/aluno.service';
import { Aluno } from 'src/app/shared/models/aluno.model';

@Component({
  selector: 'app-listar-aluno',
  templateUrl: './listar-aluno.component.html',
  styleUrl: './listar-aluno.component.css'
})
export class ListarAlunoComponent implements OnInit {
  alunos: Aluno[] = [];
  constructor(private alunoService: AlunoService) { }

  ngOnInit(): void {
    this.alunos = this.listarTodos();
  }

  listarTodos(): Aluno[] {
    return this.alunoService.listarTodos();
  }

  remover($event: any, aluno: Aluno): void {
    $event.preventDefault();
    if (confirm(`Deseja realmente remover o aluno ${aluno.nome}?`)) {
      this.alunoService.remover(aluno.id!);
      this.alunos = this.listarTodos();
    }
  }

}
```

FONTE: Elaborado pelo autor (2024).

FIGURA 29: ARQUIVO `aluno-list.component.html`.

```
<h1> Alunos </h1>
<table class="table table-striped table-bordered table-hover">
    <tbody>
        <tr>
            <th>Nome</th>
            <th>CPF</th>
            <th>E-mail</th>
            <th>Data de Nascimento</th>
            <th class="text-center">
                <a href="#" title="Novo"
                class="btn btn-xs btn-success"
                [routerLink]="['/alunos/novo']">
                    <i class="fa fa-plus" aria-hidden="true"></i> Novo
                </a>
            </th>
        </tr>
        <tr *ngFor="let aluno of alunos">
            <td>{{aluno.nome}}</td>
            <td>{{aluno.cpf}}</td>
            <td>{{aluno.email}}</td>
            <td>{{aluno.dataNascimento | date:'dd/MM/yyyy'}}</td>
            <td class="text-center" style="width: 300px;">
                <a href="#" title="Editar" alt="Editar" class="btn btn-xs btn-info"
                [routerLink]="['/alunos/editar', aluno.id]">
                    <i class="fa fa-edit" aria-hidden="true"></i> Editar
                </a>
                <a href="#" title="Remover" alt="Remover"
                    class="btn btn-xs btn-danger"
                    (click)="remover($event, aluno)">
                    <i class="fa fa-times" aria-hidden="true"></i> Remover
                </a>
            </td>
        </tr>
    </tbody>
</table>
<p *ngIf="alunos.length == 0" class="alert alert-info">
    Nenhum aluno cadastrado.
</p>
```

FONTE: Elaborado pelo autor (2024).

FIGURA 30: ARQUIVO `app-routing.module.ts`.

```
src > app > TS app-routing.module.ts > routes
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { ListarAlunoComponent } from './aluno/listar-aluno/listar-aluno.component';
import { InserirAlunoComponent } from './aluno/inserir-aluno/inserir-aluno.component';
import { EditarAlunoComponent } from './aluno/editar-aluno/editar-aluno.component';
import { ListarCursoComponent } from './curso/listar-curso/listar-curso.component';
import { InserirCursoComponent } from './curso/inserir-curso/inserir-curso.component';
import { EditarCursoComponent } from './curso/editar-curso/editar-curso.component';

const routes: Routes = [
  { path: '',
    redirectTo: 'alunos/listar',
    pathMatch: 'full'},
  { path: 'alunos',
    redirectTo: 'alunos/listar'},
  { path: 'alunos/listar',
    component: ListarAlunoComponent },
  { path: 'alunos/novo',
    component: InserirAlunoComponent },
  { path: 'alunos/editar/:id',
    component: EditarAlunoComponent },
  { path: '',
    redirectTo: 'cursos/listar',
    pathMatch: 'full'},
  { path: 'cursos',
    redirectTo: 'cursos/listar'},
  { path: 'cursos/listar',
    component: ListarCursoComponent },
  { path: 'cursos/novo',
    component: InserirCursoComponent },
  { path: 'cursos/editar/:id',
    component: EditarCursoComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

FONTE: Elaborado pelo autor (2024).

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

O projeto desenvolvido na disciplina de UX centrou-se na concepção de uma experiência de usuário em um hipotético aplicativo chamado Organize, uma solução voltada para gerenciamento de listas de compras. As decisões de design foram sempre norteadas por necessidades concretas e reflexão.

A fase inicial recaiu sobre pesquisa de segmento e criação de protótipos de baixa fidelidade para as cinco telas principais: login (FIGURA 31), listagem de compras (FIGURA 33), adição de itens (FIGURA 34) e ajuda (FIGURA 35). Cada tela foi concebida de forma a oferecer clareza e eficiência, eliminando fatores que pudessem comprometer a usabilidade. O uso de cores vibrantes e a categorização visual de itens por tons diferenciados atendem não só critérios estéticos, mas também colaboram na funcionalidade – guiando o usuário de forma intuitiva.

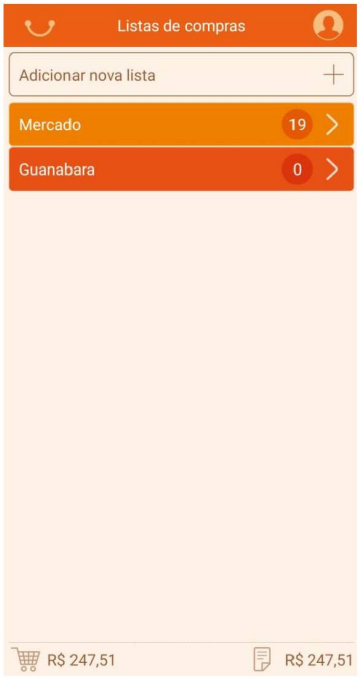
A integração com outras disciplinas se deu principalmente em WEB1, WEB2 e MOB1 e MOB2, uma vez que componentes como botões de ação e outros componentes de navegação foram implementados conforme padrões estabelecidos nos protótipos. Histórias de usuário vistas em MAG1 podem mais uma vez ter papel vital no desenvolvimento da disciplina de UX.

9.1 ARTEFATOS DO PROJETO

FIGURA 31: TELA DE LOGIN.

FONTE: Elaborado pelo autor (2024).

FIGURA 32: TELA DE CRIAÇÃO E EXIBIÇÃO DE LISTAS.

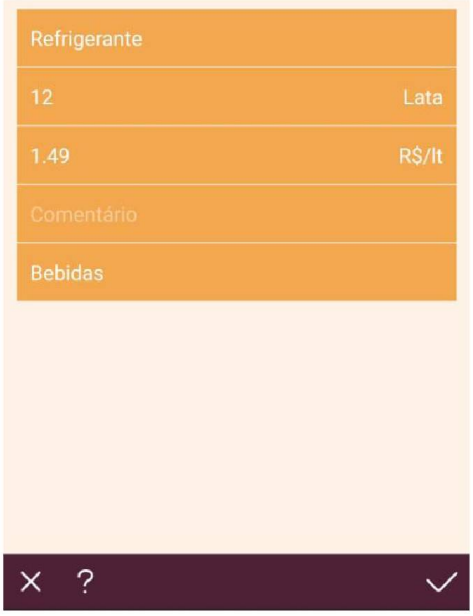


FONTE: Elaborado pelo autor (2024).

FIGURA 33: TELA DE EXIBIÇÃO DA LISTA SELECIONADA.

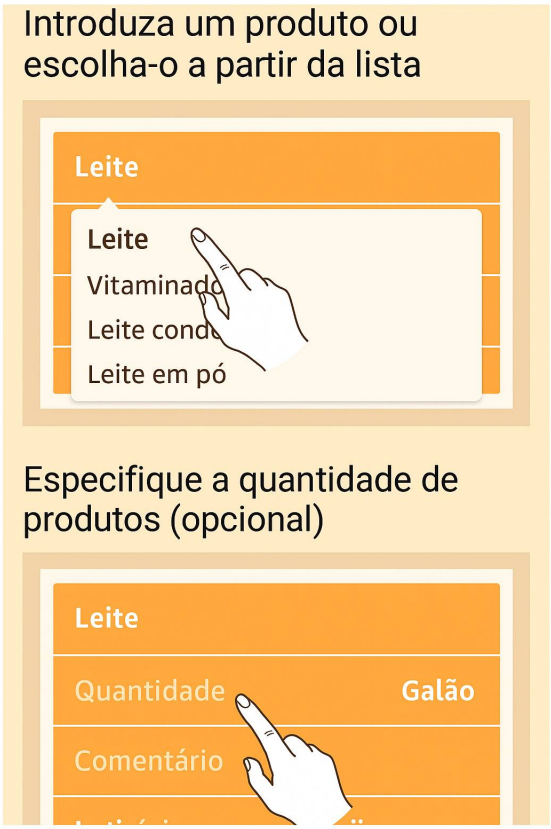
FONTE: Elaborado pelo autor (2024).

FIGURA 34: TELA DE DETALHAMENTO E EDIÇÃO DO PRODUTO.



FONTE: Elaborado pelo autor (2024).

FIGURA 35: TELA DE AJUDA.



FONTE: Elaborado pelo autor (2024).

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

Os projetos desenvolvidos nas disciplinas de Desenvolvimento Mobile permitiram vivenciar na prática os desafios de criação de aplicativos Android, cobrindo desde soluções locais até integrações com *Application Programming Interface* - API's. Em MOB1 o trabalho requisitou a criação de um aplicativo de gestão financeira pessoal, enquanto que em MOB2 o foco foi a criação de um App para consumo da Harry Potter API, usando corrotinas e chamadas assíncronas.

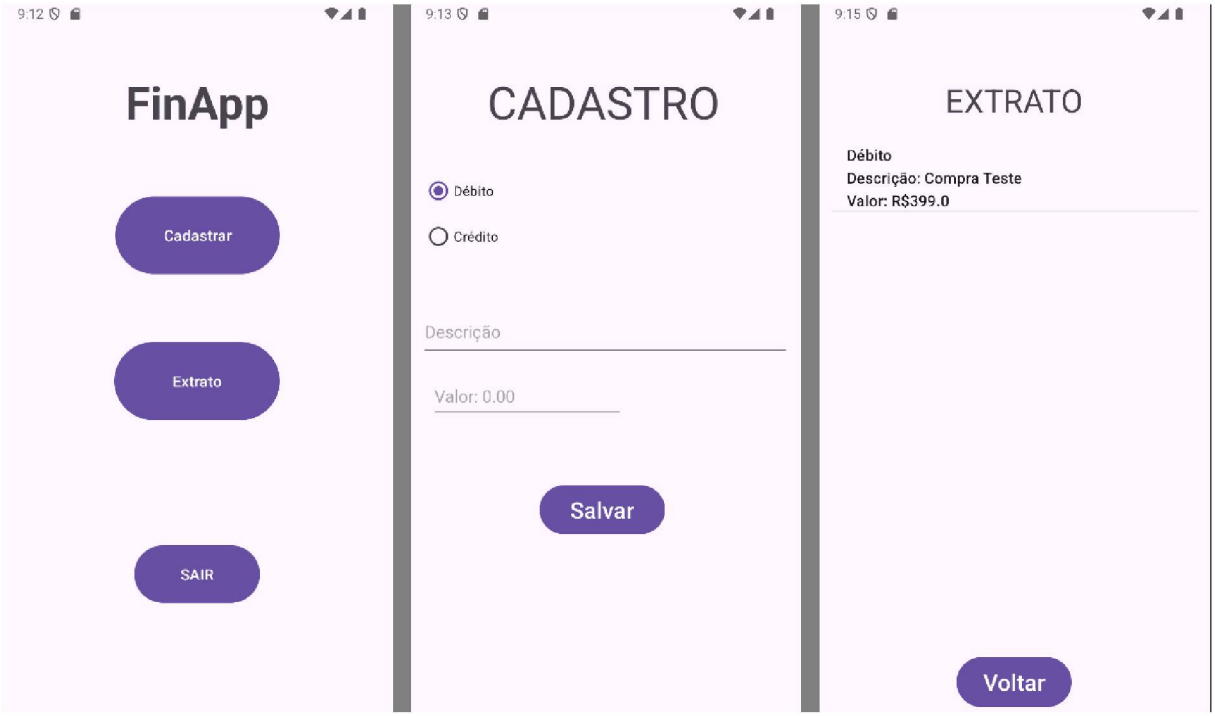
No FinApp de MOB1, a implementação seguiu desenvolvimento incremental e serviu para compreensão de componentes essenciais ao Android, como Activities (FIGURA 38), *Views* e *Adapters* (Android Developers, 2025). A interface foi pensada com intuitividade em mente, com uma tela central que direciona o usuário para funcionalidades de extrato e cadastro (FIGURA 36).

Em MOB2, o trabalho com a API do Harry Potter exigiu o uso de operações assíncronas, por meio de corrotinas (FIGURA 41), de modo que a interface permanecesse responsiva durante as requisições. Componentes modulares como *RadioButtons* para seleções e exibições em *TextViews* (FIGURA 40) mostraram-se fundamentais para o emprego do desenvolvimento ágil, uma vez que foram incrementos independentes e assim, permitindo melhores ajustes isoladamente.

As disciplinas permitiram integração natural e contínua, visto que os conhecimentos de *layout* obtidos em MOB1 foram expandidos em MOB2, com o uso de componentes mais complexo. Princípios de UX também foram úteis na concepção e organização das interfaces e até noções de infraestrutura podem ser consideradas, uma vez que foram usadas nas chamadas de API (FIGURA 42).

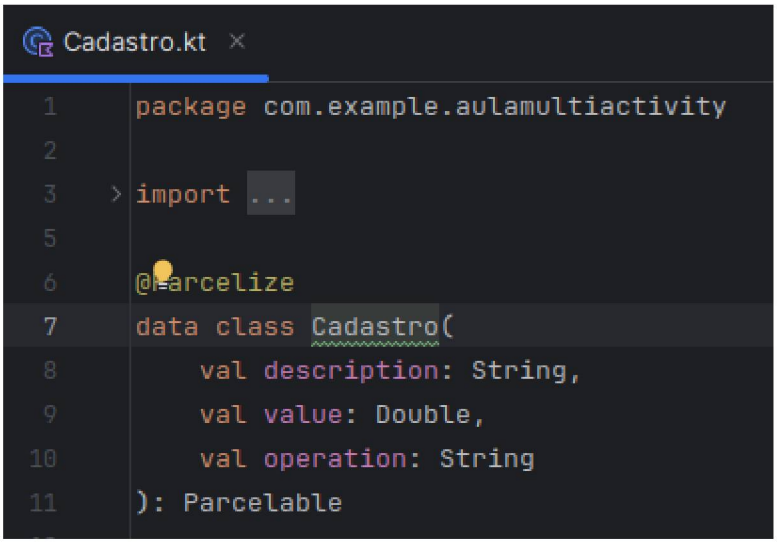
10.1 ARTEFATOS DO PROJETO

FIGURA 36: TELA PRINCIPAL, DE CADASTRO E DE EXTRATO DO FinApp.



FONTE: Elaborado pelo autor (2025).

FIGURA 37: CLASSE CADASTRO DO FinApp.



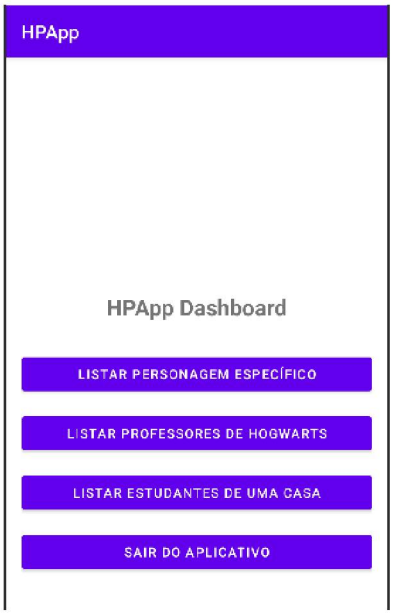
FONTE: Elaborado pelo autor (2025).

FIGURA 38: CLASSE `MainActivity` DO FinApp.

```kotlin
package com.example.aulamultiactivity

import ...
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
    //Vai para a tela do Cadastro
    fun goToCadastroActivity(view: View){
        val intent = Intent( packageContext: this, CadastroActivity::class.java)
        startActivity(intent)
    }
    //Vai para a tela do Extrato passando as informações recebidas do cadastro
    fun goToExtratoActivity(view: View){
        val bundle = intent.extras
        if(bundle != null){ //Verifica se o bundle recebido do cadastro não é nulo
            //Verifica as versões do SDK para pegar o objeto Parcelable
            val cadastro = if(Build.VERSION.SDK_INT >= 33){
                (bundle?.getParcelable( key: "cadastro", Cadastro::class.java))
            }else{
                bundle.getParcelable( key: "cadastro")
            }
            //variáveis recebem os valores do objeto Parcelable
            val descricao = cadastro?.description
            val valor = cadastro?.value
            val operacao = cadastro?.operation
            //Print das informações recebidas para conferência
            println("**MAIN** \nDescrição: " + descricao)
            println("Valor: R$" + valor)
            println("Operação: " + operacao)
            //Chama a tela de Extrato e envia as informações do cadastro
            val intent = Intent( packageContext: this, ExtratoActivity::class.java)
            intent.putExtra( name: "cadastro", cadastro)
            startActivity(intent)
        }
    }
    //Clique no botão SAIR -> Fecha toda a aplicação
    fun fechaApp(view: View){
        finishAffinity()
    }
}
```

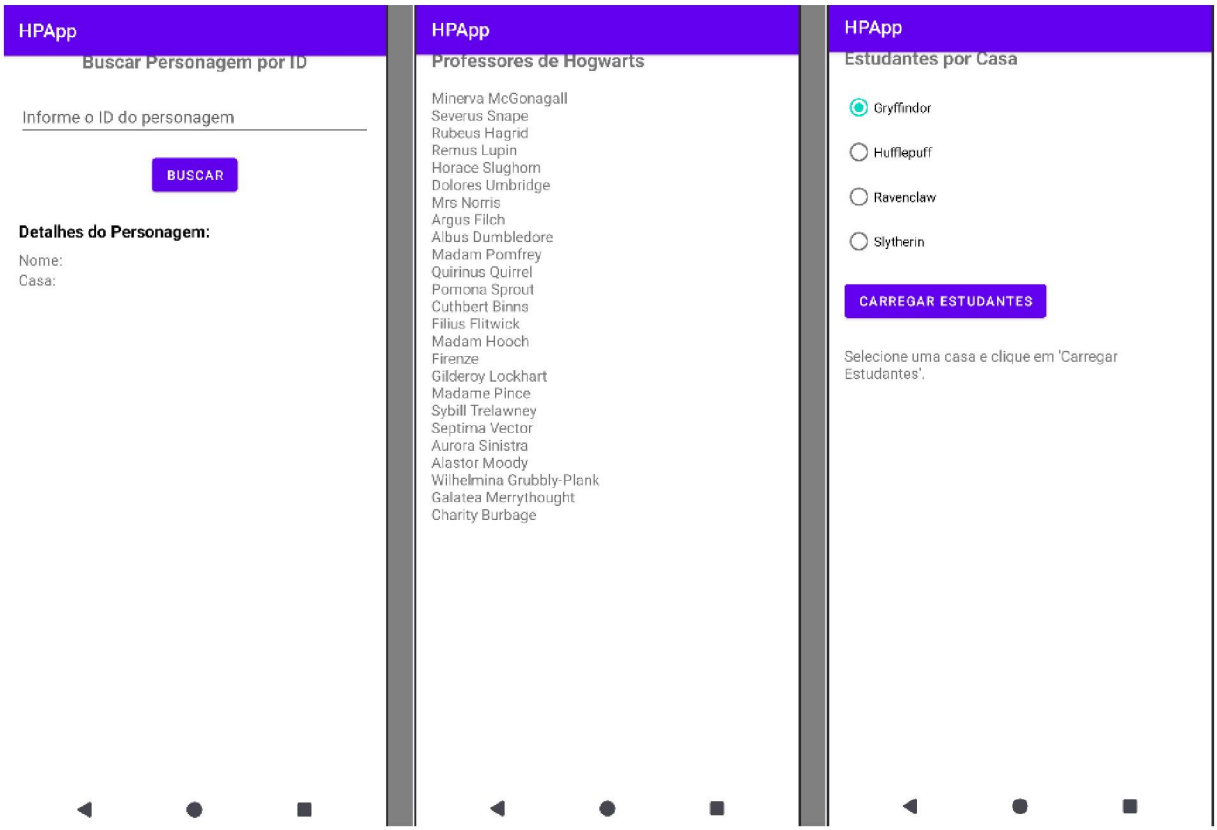
FONTE: Elaborado pelo autor (2025).

FIGURA 39: TELA PRINCIPAL DO HPApp.



FONTE: Elaborado pelo autor (2025).

FIGURA 40: TELA DE PERSONAGENS, PROFESSORES E CASAS DO HPApp.



FONTE: Elaborado pelo autor (2025).

FIGURA 41: EXEMPLO DE CHAMADA COM CORROTINAS DO HPApp.

```
    private fun loadHogwartsStaff() {
        lifecycleScope.launch(Dispatchers.IO) {
            try {
                val response = RetrofitClient.instance.getHogwartsStaff()
                withContext(Dispatchers.Main) {
                    if (response.isSuccessful && response.body() != null) {
                        val staffList = response.body()
                        val staffNames = staffList?.joinToString(separator = "\n") { it.name }
                        binding.tvStaffList.text = staffNames ?: "Nenhum professor encontrado."
                    } else {
                        binding.tvStaffList.text = "Erro ao carregar professores."
                        Toast.makeText(this@HogwartsStaffActivity, "Erro ao carregar professores: $
{response.code()}", Toast.LENGTH_SHORT).show()
                    }
                }
            } catch (e: Exception) {
                withContext(Dispatchers.Main) {
                    binding.tvStaffList.text = "Erro de conexão."
                    Toast.makeText(this@HogwartsStaffActivity, "Erro de conexão: ${e.message}",
Toast.LENGTH_SHORT).show()
                    e.printStackTrace()
                }
            }
        }
    }
```

FONTE: Elaborado pelo autor (2025).

FIGURA 42: EXEMPLO DE RESPOSTA DA API.


```
[
  {
    "name": "Harry Potter",
    "house": "Gryffindor",
    "school": "Hogwarts School of Witchcraft and Wizardry",
    "patronus": "stag"
  },
  {
    "name": "Hermione Granger",
    "house": "Gryffindor",
    "school": "Hogwarts School of Witchcraft and Wizardry",
    "patronus": "otter"
  },
  {
    "name": "Ron Weasley",
    "house": "Gryffindor",
    "school": "Hogwarts School of Witchcraft and Wizardry",
    "patronus": "Jack Russell terrier"
  },
  {
    "name": "Draco Malfoy",
    "house": "Slytherin",
    "school": "Hogwarts School of Witchcraft and Wizardry",
    "patronus": null
  },
  {
    "name": "Albus Dumbledore",
    "house": "Gryffindor",
    "school": "Hogwarts School of Witchcraft and Wizardry",
    "patronus": "phoenix"
  }
]
```

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

O trabalho da disciplina de infraestrutura teve como objetivo principal a configuração de um ambiente DevOps usando tecnologias como Docker, GitLab e Jenkins. A atividade prática consistiu em criar um contêiner Docker (FIGURA 41), configurar o acesso e realizar operações de versionamento de código (FIGURA 45).

Este trabalho foi bastante interessante para aquisição de prática no sentido de como uma boa infraestrutura pode acelerar e agregar confiabilidade ao processo de desenvolvimento de software, tendo noção dos benefícios de integração e entrega contínua (Duvall et al., 2007) e também a automação de processos de *build*, testes e *deploy* (FIGURA 44).

Esta experiência se mostra útil em demais disciplinas, onde a integração se daria de forma satisfatória principalmente em MOB1/MOB2 e WEB1/WEB2, bem como na disciplina de TEST, uma vez que o ambiente conteinerizado é ideal para execução de testes automatizados de forma isolada.

De modo geral, o trabalho evidencia a importância de infraestrutura como código e automação de processos no desenvolvimento. Embora sejam passos iniciais, a criação e configuração do ambiente permitem caminhos para práticas mais avançadas de *DevOps*, trazendo ganhos de qualidade e produtividade significativos em futuros projetos.

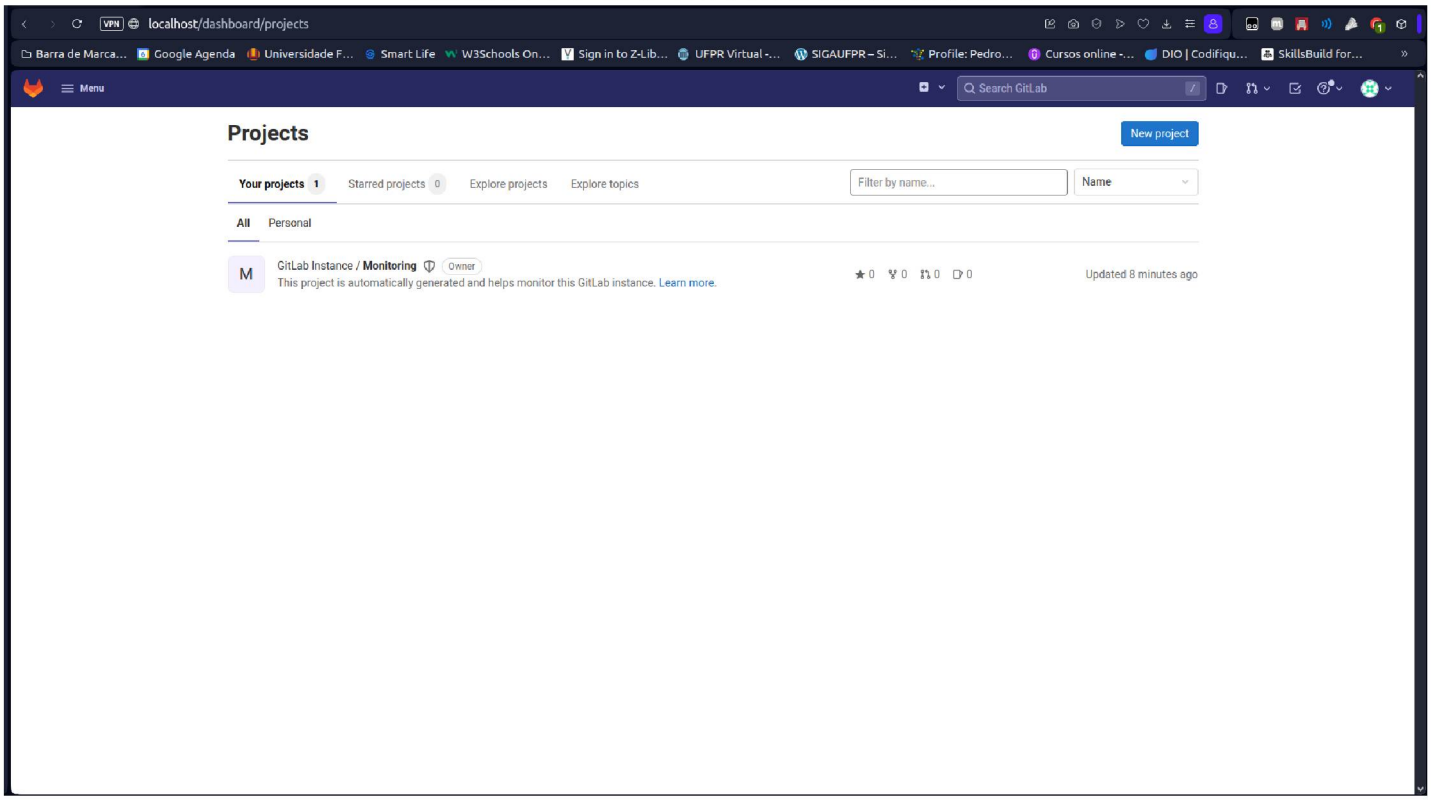
11.1 ARTEFATOS DO PROJETO

FIGURA 43: CAPTURA DO COMANDO E CONTÊINER SENDO EXECUTADO.

```
pevferreira@pevferreira-K46CA:~$ sudo docker run -d --name 202400184599 -p 22:22 -p 80:80 -p 443:443 -p 9091:9091 dfwandarti/gitlab_jenkins:3
0f880a0cf3bccc4181260bd7899d9bf5a0634a90271057f61fab4a44180d9818
pevferreira@pevferreira-K46CA:~$ sudo docker ps
CONTAINER ID   IMAGE                          COMMAND             CREATED          STATUS                            PORTS
                                                                  NAMES
0f880a0cf3bc   dfwandarti/gitlab_jenkins:3    "/assets/wrapper"   10 seconds ago   Up 9 seconds (health: starting)   0.0.0.0:22->22/tcp, [::]:22->22/tcp, 0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0:443->443
/tcp, [::]:443->443/tcp, 0.0.0.0:9091->9091/tcp, [::]:9091->9091/tcp   202400184599
pevferreira@pevferreira-K46CA:~$
```

FONTE: Elaborado pelo autor (2025).

FIGURA 44: CAPTURA DO NAVEGADOR.



FONTE: Elaborado pelo autor (2025).

FIGURA 45: CAPTURA DO LOG DO GIT COM O COMMIT.

FONTE: Elaborado pelo autor (2025).

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

O projeto desenvolvido na disciplina TEST teve como objetivo criar um *script* de automação utilizando Playwright (Microsoft, 2025) , capaz de interagir com a aplicação web Anotepad (FIGURA 48). O teste automatizado deveria preencher o título da nota com "Entrega trabalho TEST DAS 2024" e no corpo da nota inserir o nome e matrícula do aluno (FIGURA 46).

Com esta atividade foi possível demonstrar na prática como os testes automatizados podem ser integrados ao ciclo de desenvolvimento ágil. O uso do Playwright mostrou-se particularmente eficaz por sua capacidade de simular interações do usuário de forma clara e rápida.

A integração com outras disciplinas do curso poderia dar-se em vários aspectos. O conhecimento adquirido em WEB1 e WEB2 foi essencial para entender a estrutura da página web e os elementos que deveriam ser manipulados pelo teste. As práticas de programação ágil aprendidas em AAP foram aplicadas na construção do código de teste, enquanto os conceitos de versionamento de INFRA poderiam ser usados para gerenciar o código-fonte do projeto de testes.

O trabalho reforçou a importância dos testes automatizados como parte essencial do processo de desenvolvimento de software ágil (Beck, 2003). A capacidade de executar testes rapidamente e de forma consistente como parte do processo de *build* através do Maven (FIGURA 47) (Apache, 2025) .

12.1 ARTEFATOS DO PROJETO

FIGURA 46: CÓDIGO PARA EXECUTAR JUNTO AO ANOTEPAD.

```java
package com.testes;

import com.microsoft.playwright.Browser;
import com.microsoft.playwright.BrowserType;
import com.microsoft.playwright.Page;
import com.microsoft.playwright.Playwright;
import com.microsoft.playwright.options.LoadState;

public class App {
    public static void main(String[] args) {
        try (Playwright playwright = Playwright.create()) {
            Browser browser = playwright.chromium().launch(new BrowserType.LaunchOptions().setHeadless(false));
            Page page = browser.newPage();
            page.navigate("https://pt.anotepad.com");
            page.waitForLoadState(LoadState.NETWORKIDLE);

            page.locator("[placeholder='Título da Nota']").click();
            page.locator("[placeholder='Título da Nota']").fill("Entrega trabalho TEST DAS 2024");

            page.locator("div.note-editable").click();
            page.keyboard().type("Nome: Pedro Vieira Ferreira\nMatrícula: 202400184599");

            page.waitForTimeout(5500);
        }
    }
}
```

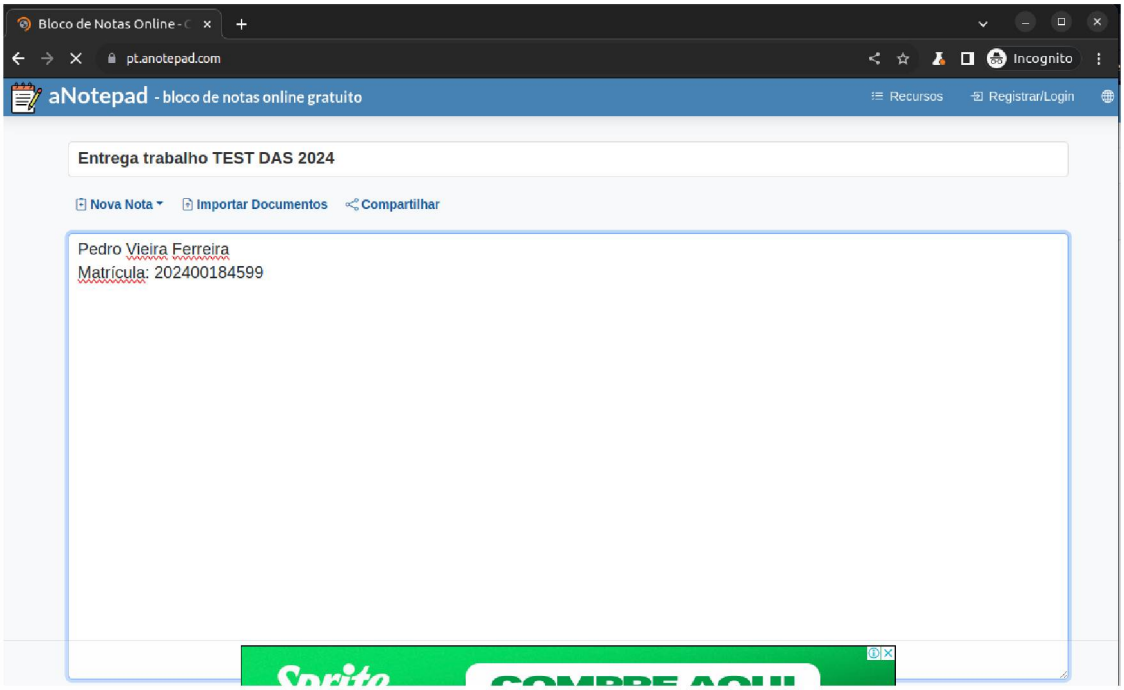
FONTE: Elaborado pelo autor (2025).

FIGURA 47: CAPTURA DOS TESTES COM MAVEN.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.testes.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.041 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Anotepad ---
[INFO] Building jar: /home/pevferreira/git/Anotepad/target/Anotepad-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ Anotepad ---
[INFO] Installing /home/pevferreira/git/Anotepad/target/Anotepad-1.0-SNAPSHOT.jar to /home/pevferreira/.m2/repository/com/testes/Anotepad/1.0-SNAPSHOT/Anotepad-1.0-SNAPSHOT.jar
[INFO] Installing /home/pevferreira/git/Anotepad/pom.xml to /home/pevferreira/.m2/repository/com/testes/Anotepad/1.0-SNAPSHOT/Anotepad-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

FONTE: Elaborado pelo autor (2025).

FIGURA 48: CAPTURA DA PLATAFORMA ANOTEPAD APÓS EXECUÇÃO DO CÓDIGO.



FONTE: Elaborado pelo autor (2025).

FIGURA 49: TRECHO DO ARQUIVO `pom.xml`.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.testes</groupId>
  <artifactId>Anotepad</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Anotepad</name>
  <url>http://maven.apache.org</url>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>

  <repositories>
    <repository>
      <id>playwright</id>
      <url>https://maven.pkg.github.com/microsoft/playwright-java</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.microsoft.playwright</groupId>
      <artifactId>playwright</artifactId>
      <version>1.41.0</version>
    </dependency>
  </dependencies>
```

FONTE: Elaborado pelo autor (2025).

13 CONCLUSÃO

Ao longo deste memorial, foi possível acompanhar uma trajetória coerente e integrada entre as disciplinas que compõem o curso de Desenvolvimento Ágil de Software. Desde os fundamentos teóricos apresentados em MADS até as implementações práticas em disciplinas como WEB, MOB e TEST, observou-se como os princípios ágeis permeiam todas as etapas do ciclo de desenvolvimento de software. A abordagem não se limitou à aplicação mecânica de metodologias, mas consistiu em uma assimilação progressiva de valores como colaboração, adaptação e entrega contínua de valor.

Os projetos desenvolvidos demonstraram na prática a eficácia dessa abordagem integrada. A modelagem ágil realizada em MAG1 e MAG2, com seus diagramas e histórias de usuário, forneceu a base para implementações técnicas que mantiveram o foco nas necessidades reais do usuário. A gestão de projetos em GAP1 e GAP2 trouxe a dimensão do planejamento iterativo e da melhoria contínua, enquanto as disciplinas técnicas como AAP, WEB e MOB evidenciaram a importância do código bem estruturado e de interfaces intuitivas. A infraestrutura ágil configurada em INFRA e os testes automatizados desenvolvidos em TEST completaram o ciclo, mostrando como a automação pode potencializar a qualidade e a velocidade de entrega.

Entretanto, a experiência também revelou desafios significativos na adoção prática do desenvolvimento ágil. A integração entre ferramentas e tecnologias diversas mostrou-se complexa, exigindo não apenas conhecimento técnico, mas capacidade de adaptação e visão sistêmica. A manutenção da qualidade em um contexto de entregas rápidas provou ser um equilíbrio delicado, que demanda disciplina na aplicação de práticas como testes automatizados e revisões constantes (Beck, 2003). Além disso, a tensão entre a agilidade e a necessidade de certo rigor técnico, especialmente em projetos mais complexos, emergiu como uma questão recorrente.

Esses desafios, contudo, longe de serem obstáculos intransponíveis, representam oportunidades de crescimento e aprimoramento. Eles reforçam que o desenvolvimento ágil não é uma fórmula pronta, mas um conjunto de princípios que devem ser adaptados a cada contexto. O curso proporcionou as bases teóricas e práticas para essa adaptação, preparando para os cenários dinâmicos do mercado de tecnologia.

Em última análise, esta jornada através das disciplinas deixou claro que o verdadeiro desenvolvimento ágil vai além de processos e ferramentas. Trata-se de uma mentalidade que valoriza as pessoas, a colaboração e a capacidade de responder a mudanças - valores que, quando internalizados, transformam não apenas a forma de construir software, mas a maneira de encarar os desafios profissionais de forma geral. O aprendizado aqui consolidado servirá como alicerce para uma carreira marcada pela capacidade de adaptação, qualidade técnica e foco contínuo em entregar valor.

REFERÊNCIAS

ANDERSON, David J. Kanban: mudança evolucionária bem-sucedida para seu negócio de tecnologia. Tradução de Ricardo Vargas. 1. ed. São Paulo: Blue Hole Press, 2010.

ANDROID DEVELOPERS. Developer guides. 2025. Disponível em: https://developer.android.com/guide. Acesso em: 10 jan. 2025.

ANOTEPAD. Anotepad.com – Online Notepad. 2025. Disponível em: https://anotepad.com/. Acesso em: 15 mar. 2025.

APACHE SOFTWARE FOUNDATION. Apache Maven Project. 2025. Disponível em: https://maven.apache.org/. Acesso em: 5 mar. 2025.

BECK, Kent. Extreme programming explained: embrace change. 2. ed. Boston: Addison-Wesley, 2000.

BECK, Kent. Test-driven development: by example. Boston: Addison-Wesley, 2003.

BECK, Kent et al. Manifesto Ágil para Desenvolvimento de Software. 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/. Acesso em: 10 jun. 2025.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. The unified modeling language user guide. 2. ed. Boston: Addison-Wesley, 2005.

COHN, Mike. User stories applied: for agile software development. Boston: Addison-Wesley, 2004.

CRISPIN, Lisa; GREGORY, Janet. Agile testing: a practical guide for testers and agile teams. Boston: Addison-Wesley, 2009.

DOCKER. What is a container? 2025. Disponível em: https://www.docker.com/resources/what-container/. Acesso em: 23 fev. 2025.

DUVALL, Paul M. et al. Continuous integration: improving software quality and reducing risk. Boston: Addison-Wesley, 2007.

GEEKSFORGEEKS. Algoritmo QuickSort. 2024. Disponível em: https://www.geeksforgeeks.org/quick-sort/. Acesso em: 14 ago. 2024.

GITLAB. GitLab Docs. 2025. Disponível em: https://docs.gitlab.com/. Acesso em: 25 fev. 2025.

HARRY POTTER CHARACTERS API. PublicAPI. [S. l.], 2025. Disponível em: https://publicapi.dew/harry-potter-characters-api. Acesso em: 22 jan. 2025.

HUMBLE, Jez; FARLEY, David. Continuous delivery: reliable software releases through build, test, and deployment automation. Boston: Addison-Wesley, 2010.

JENKINS. Jenkins User Documentation. 2025. Disponível em: https://www.jenkins.io/doc/. Acesso em: 28 fev. 2025.

KANBAN BOARD GAME. Kanbanboardgame.com. 2024. Disponível em: https://kanbanboardgame.com/. Acesso em: 16 mai. 2024.

KNIBERG, Henrik. Scrum e XP direto das trincheiras. Tradução de Vinicius Manhaes. São Paulo: Casa do Código, 2009. Disponível em: https://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches/. Acesso em: 10 jun. 2025.

MARTIN, Robert C. Clean code: a handbook of agile software craftsmanship. Upper Saddle River: Prentice Hall, 2008.

MICROSOFT. Playwright: Fast and reliable end-to-end testing for modern web apps. 2025. Disponível em: https://playwright.dev/. Acesso em: 1 mar. 2025.

POPPENDIECK, Mary; POPPENDIECK, Tom. Lean software development: an agile toolkit. Boston: Addison-Wesley, 2003.

PRESSMAN, Roger S. Software engineering: a practitioner's approach. 8. ed. New York: McGraw-Hill, 2016.

SANDERSON, Rebecca. Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale. Sebastopol: O'Reilly Media, 2016.

SCHWABER, Ken; SUTHERLAND, Jeff. The Scrum Guide. 2020. Disponível em: https://www.scrumguides.org/. Acesso em: 10 jun. 2025.

SOMMERVILLE, Ian. Software engineering. 10. ed. Harlow: Pearson, 2015.

STELLMAN, Andrew; GREENE, Jennifer. Learning agile: understanding Scrum, XP, Lean, and Kanban. Sebastopol: O'Reilly Media, 2014.