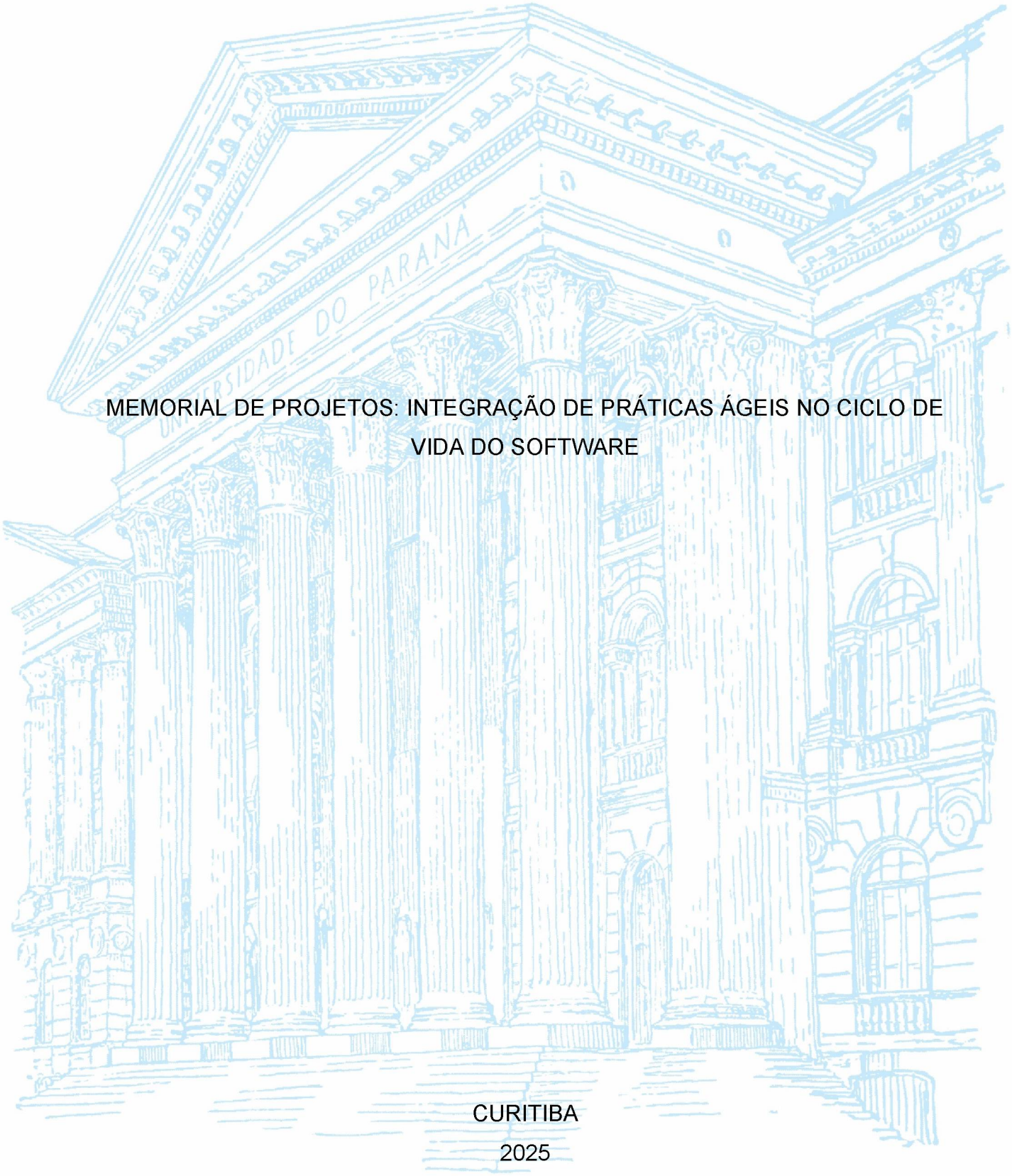
UNIVERSIDADE FEDERAL DO PARANÁ

DANIEL MACHADO PINTOS



MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO CICLO DE VIDA DO SOFTWARE

CURITIBA

2025

DANIEL MACHADO PINTOS

MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO CICLO DE VIDA DO 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. Jaime Wojciechowski

CIDADE

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 **DANIEL MACHADO PINTOS**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO CICLO DE VIDA DO 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 Agosto de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este Memorial de Projetos concretiza o aprendizado e a aplicação dos conceitos de Desenvolvimento Ágil de Software, por meio da criação e apresentação artefatos práticos desenvolvidos ao longo das diversas disciplinas do curso de Pós-Graduação em Desenvolvimento Ágil de Software. Os objetivos do memorial são evidenciar a capacidade do aluno de integrar conhecimentos de diferentes áreas, demonstrar a aplicabilidade prática das metodologias ágeis e refletir sobre os desafios inerentes à sua adoção em projetos de software. Os artefatos apresentados abrangem desde a modelagem (MADS, MAG1, MAG2) e o gerenciamento de projetos (GAP1, GAP2), até a implementação técnica de *backends* e *frontends* (INTRO, BD, AAP, WEB1, WEB2, MOB1, MOB2), além de aspectos de experiência do usuário (UX), infraestrutura (INFRA) e testes automatizados (TEST). A integração desses conteúdos se manifesta na abordagem iterativa e incremental dos projetos, nos ciclos de *feedback* rápidos e na priorização da entrega contínua de valor, características essenciais do desenvolvimento ágil. O memorial destaca como cada disciplina contribuiu para a formação de uma visão holística e prática, capacitando o aluno a atuar em ambientes de desenvolvimento ágil e tecnologias atuais utilizadas no mercado de trabalho.

Palavras-chave: desenvolvimento ágil, metodologias ágeis, programação

ABSTRACT

This Project Portfolio embodies the learning and application of Agile Software Development concepts through the creation and presentation of practical artifacts produced throughout the Post-Graduate course in Agile Software Development. Its goals are to highlight the student's ability to integrate knowledge from different areas, demonstrate the practical applicability of agile methodologies, and reflect on the challenges inherent in their adoption in software projects. The artifacts presented range from modeling (MADS, MAG1, MAG2) and project management (GAP1, GAP2) to the technical implementation of back-ends and front-ends (INTRO, BD, AAP, WEB1, WEB2, MOB1, MOB2), as well as user-experience aspects (UX), infrastructure (INFRA), and automated testing (TEST). The integration of these topics emerges through the iterative and incremental approach of the projects, rapid feedback cycles, and the prioritization of continuous value delivery—hallmark characteristics of agile development. The portfolio demonstrates how each subject contributed to forming a holistic and practical perspective, equipping the student to work effectively in agile environments using technologies currently employed in the industry.

Keywords: agile development, agile methodologies, programming

SUMÁRIO

1 PARECER TÉCNICO

O Memorial de Projetos apresentado consiste na jornada de aprendizado e aplicação prática dos princípios do Desenvolvimento Ágil de Software, culminando na criação de artefatos que demonstram a integração e a aplicabilidade de modo prático das metodologias ágeis ao longo do curso de Pós-Graduação. Este parecer técnico explora a forma como os projetos desenvolvidos em cada disciplina se conectam e contribuem para uma visão holística do ciclo de vida ágil.

A base conceitual foi estabelecida na disciplina de MADS – Métodos Ágeis para Desenvolvimento de Software (Capítulo 2), onde foi construído um mapa mental abrangente sobre os fundamentos da Engenharia de Software e as metodologias ágeis. Este artefato inicial serviu como um guia para a compreensão dos pilares do desenvolvimento ágil – como ciclos de *feedback* curtos, melhoria contínua e entrega de valor – que foram explorados e aprofundados nas disciplinas subsequentes. A especificação, projeto, implementação, validação e evolução, atividades essenciais do processo de software, foram introduzidas e interligadas, formando a espinha dorsal para os projetos futuros.

Em seguida, as disciplinas de MAG1 e MAG2 – Modelagem Ágil de Software 1 e 2 (Capítulo 3) focaram na modelagem de software com *UML* e histórias de usuário. Os artefatos criados nessa fase, como diagramas e *backlogs* executáveis, evidenciaram a importância de uma modelagem *"just-in-time"* que gere valor real, eliminando desperdícios de documentação excessiva. A adoção do Domain-Driven Design (Evans, 2003) garantiu que cada artefato de modelagem estivesse diretamente alinhado aos problemas do negócio, promovendo uma comunicação clara e colaboração contínua entre *stakeholders* e equipe de desenvolvimento, aspectos cruciais para a agilidade.

O gerenciamento de projetos foi abordado nas disciplinas de GAP1 e GAP2 – Gerenciamento Ágil de Projetos de Software 1 e 2 (Capítulo 4). Os projetos desenvolvidos materializaram um plano de *release* e a implementação de um quadro Kanban, com métricas e relatórios. O plano de *release* (GAP1) demonstrou a aplicação do *story-pointing* (Cohn, 2004) e do planejamento incremental (Schwaber & Sutherland, 2020), conferindo flexibilidade ao escopo e previsibilidade às entregas. O quadro Kanban (GAP2), por sua vez, exemplificou a gestão visual do fluxo de trabalho, os limites de WIP (Work In Progress) e a utilização de métricas como *lead time* e *cycle time* (Dinardo, 2020), reforçando a transparência e a gestão de riscos essenciais para a agilidade (Anderson,

2010). A integração do planejamento de *release* com a gestão visual do Kanban permitiu antecipar mudanças e manter o ritmo de entregas, aderindo aos princípios do Manifesto Ágil de entrega contínua de valor (Humble & Farley, 2011).

A vertente de implementação técnica teve início com INTRO – Introdução à Programação (Capítulo 5), onde foi desenvolvido um sistema bancário simplificado em Java. Este projeto introduziu conceitos fundamentais como o padrão DAO (Data Access Object) com JDBC para persistência em MySQL, alinhando-se a boas práticas de camadas limpas (Alur, Cruspi & Malks, 2003). A disciplina destacou a importância do TDD (Test-Driven Development) (Beck, 2003) com a utilização de testes JUnit, garantindo a detecção precoce de defeitos e a produção de um design modular. A configuração de uma pipeline para compilação e execução de testes a cada *push* reforçou a Integração Contínua (CI), um pilar da agilidade que promove *feedback* rápido e entrega frequente (Merkel, 2014). A aplicação de princípios de Clean Code (Deitel, 2010; Martin, 2008) assegurou a legibilidade e manutenção do código, preparando o terreno para equipes ágeis sustentáveis.

A disciplina de BD – Banco de Dados (Capítulo 6) solidificou a compreensão da modelagem e implementação de bancos de dados. O projeto de um sistema de Controle de Vacinação envolveu a criação de um Modelo Entidade-Relacionamento (MER), esquema lógico relacional e DDL testável. A prática de normalização até 3FN (Codd & Amber, 1970; Ambler, 2023) garantiu a integridade dos dados e a redução de redundâncias, um aspecto vital para a evolução de esquemas em ambientes ágeis. A ênfase no Test Driven Database Development e na proteção de dados críticos em *deploys* contínuos (CHEN, 1976) demonstrou a importância de um banco de dados robusto e adaptável a mudanças frequentes, capacitando as equipes a iterar sem comprometer a qualidade dos dados e a suportar análises em tempo real com transações ACID.

Em AAP – Aspectos Ágeis de Programação (Capítulo 7), a disciplina focou em boas práticas de Clean Code, TDD, programação em par e refatoração incremental. A refatoração do algoritmo BubbleSort em Java, seguindo o ciclo vermelho-verde-refatora de Kent Beck (2003), ilustrou a evolução do código por meio de testes JUnit. A aplicação de nomes significativos, funções pequenas e classes com responsabilidade única (Martin, 2008), bem como a extração de métodos e eliminação de código duplicado (Fowler, 2018), demonstrou a qualidade interna do código. A integração com pipelines de integração contínua para métricas de cobertura reforçou a importância da automação e da melhoria contínua da qualidade do código.

As disciplinas de WEB1 e WEB2 – Desenvolvimento Web 1 e 2 (Capítulo 8) abordaram o desenvolvimento de aplicações *front-end* e *back-end*. O projeto de um protótipo Angular para gerenciar Alunos, Cursos e Matrículas demonstrou a iteração incremental orientada ao valor. A primeira parte (WEB1) focou na interface com Angular e Bootstrap, utilizando Local Storage. A segunda parte (WEB2) ampliou o escopo para incluir Matrículas e migrou o *back-end* para Spring Boot, Spring Data JPA e Postgres, utilizando testes de integração e scripts SQL versionados (Beck, 2001). A prática de Thin Vertical Slices (Hunt & Thomas, 2020), TDD, reúso de componentes e refatoração contínua (Fowler, 2018) foram essenciais para a construção de um sistema escalável. A conexão com a disciplina de Banco de Dados (BD) no esquema de Aluno-Curso-Matrícula demonstrou a integração prática dos conhecimentos adquiridos.

A disciplina de UX – UX no Desenvolvimento Ágil de Software (Capítulo 9) trouxe o usuário para o centro do processo de desenvolvimento. O projeto de um site de busca de treinadores, focado na experiência do usuário, na interatividade e no *feedback*, ilustrou a importância do Design Centrado no Usuário (UCD) (CURCIO, 2021). A criação de um MVP (Minimum Viable Product) (Zanette, 2024) e a justificativa do valor de mercado do produto reforçaram a necessidade de entender as demandas e desejos do *stakeholder*. A íntima relação com as disciplinas WEB evidencia como o UX é crucial para o refinamento do produto final e para a redução das taxas de abandono (Camara, 2023).

O desenvolvimento mobile foi abordado em MOB1 e MOB2 – Desenvolvimento Mobile 1 e 2 (Capítulo 10), com a criação de aplicações Android em Kotlin. Os projetos, que variavam de mini-apps com *story slices* (Kniberg, 2010) a aplicações com persistência local (SQLite), demonstraram a importância do MVVM (Model-View-ViewModel) para separar a UI da lógica (Glauber, 2022), facilitando refatoração e testes. A introdução de FileStorage e migrações de *schema* versionadas, fundamentadas em BD, suportou o *deploy* contínuo. A utilização de componentes reutilizáveis e Clean Kotlin (Martin, 2008) garantiu um código conciso. O projeto final, com a integração de web *services* (WEB1) e Corrotinas, exemplificou a capacidade de trabalhar em projetos com o Lean Startup Loop (Ries, 2011) – construir → medir → aprender – habilidades cruciais para equipes ágeis mobile.

A disciplina de INFRA – Infraestrutura para desenvolvimento e implantação de Software (DevOps) (Capítulo 11) forneceu a base para a sustentação de sistemas em produção. A construção de uma esteira CI/CD (Integração Contínua).

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

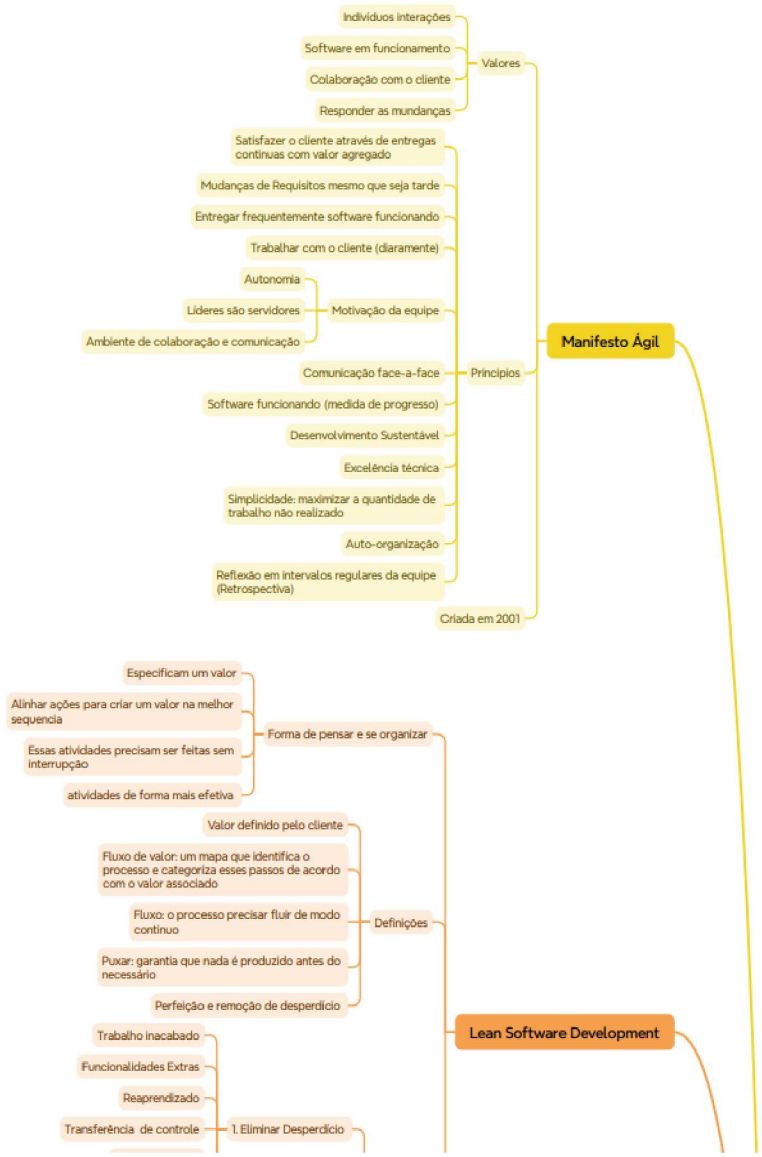
Essa disciplina teve como produto principal a criação de um mapa mental integrado que explica os principais conceitos sobre os fundamentos da Engenharia de Software, os modelos de processo tradicionais e as metodologias ágeis. Partindo da definição de engenharia de software, segundo a IEEE (2017), “aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, à operação e à manutenção de software”. O trabalho reforça a adoção de práticas ágeis e evidencia atributos ao promover ciclos curtos de *feedbacks*, melhoria contínua e foco na entrega de valor. No cerne do trabalho, podemos encontrar quatro atividades essenciais do processo – especificação, projeto e implementação, validação e evolução. Esses pilares que serão abordados nas demais disciplinas.

A disciplina atua na transição para o projeto, transformando requisitos em um sistema. Dessa forma o projeto serviu como base conceitual para compreender rapidamente como cada área de conhecimento contribui para o ciclo de vida ágil. Além dos fundamentos, o projeto sintetiza os princípios e práticas de Scrum, XP, Kanban e Lean, sendo o destaque os artefatos (*backlog*, incremento, *burndown*) e cerimônias de inspeção/adaptação e métricas de fluxo.

O projeto demonstrou a importância de enxergar processo, métodos e ferramentas como um ecossistema único. Ao aderir testes automatizados para validação contínua, Infraestrutura e Devops para sustentar a entrega e evolução por meio de integração e implementação contínua, e gerenciamento ágil de projetos orquestram o fluxo de trabalho garantindo transparência e gestão de riscos. Consolidar conceitos em um artefato único e navegável ele facilita a comunicação entre *stakeholders* acadêmicos. O trabalho não apenas cumpre com o objetivo pedagógico, mas também se torna um instrumento integrador que sustenta as entregas incrementais ao longo do curso.

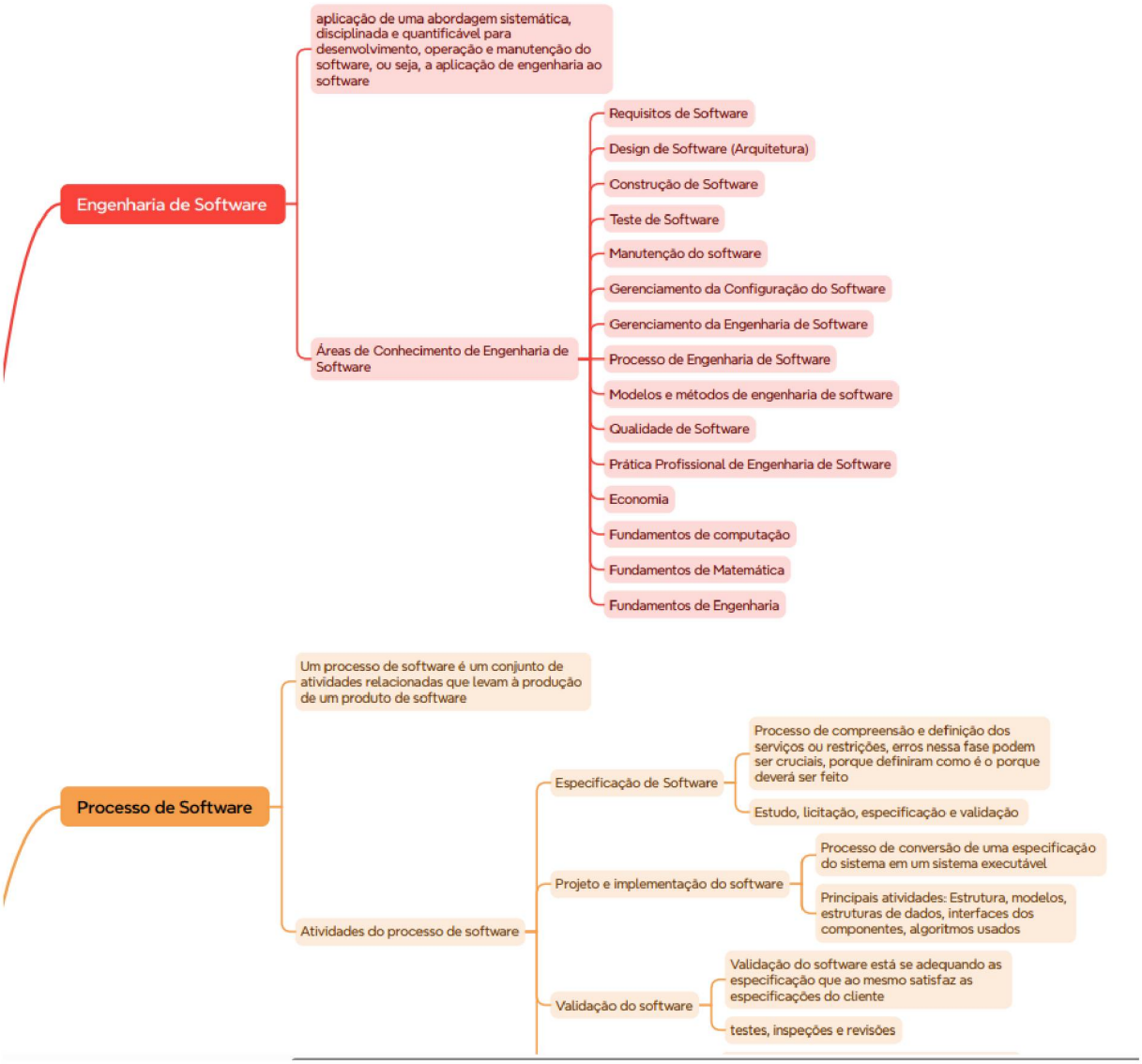
2.1 ARTEFATOS DO PROJETO

Figura 1 – Parte do mapa mental sobre manifesto ágil e Lean Software Development para resumo de informações sobre Métodos Ágeis



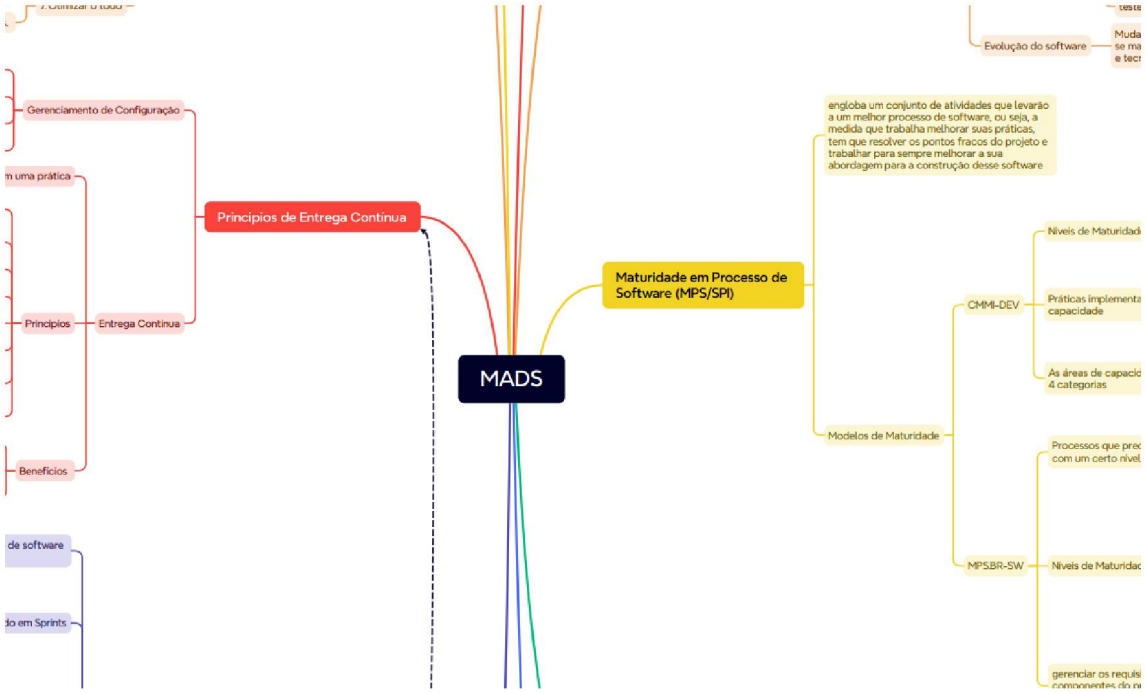
Fonte: O autor (2025)

Figura 2 – Parte do mapa mental sobre Engenharia de Software e Processo de Software para resumo de informações sobre Métodos Ágeis



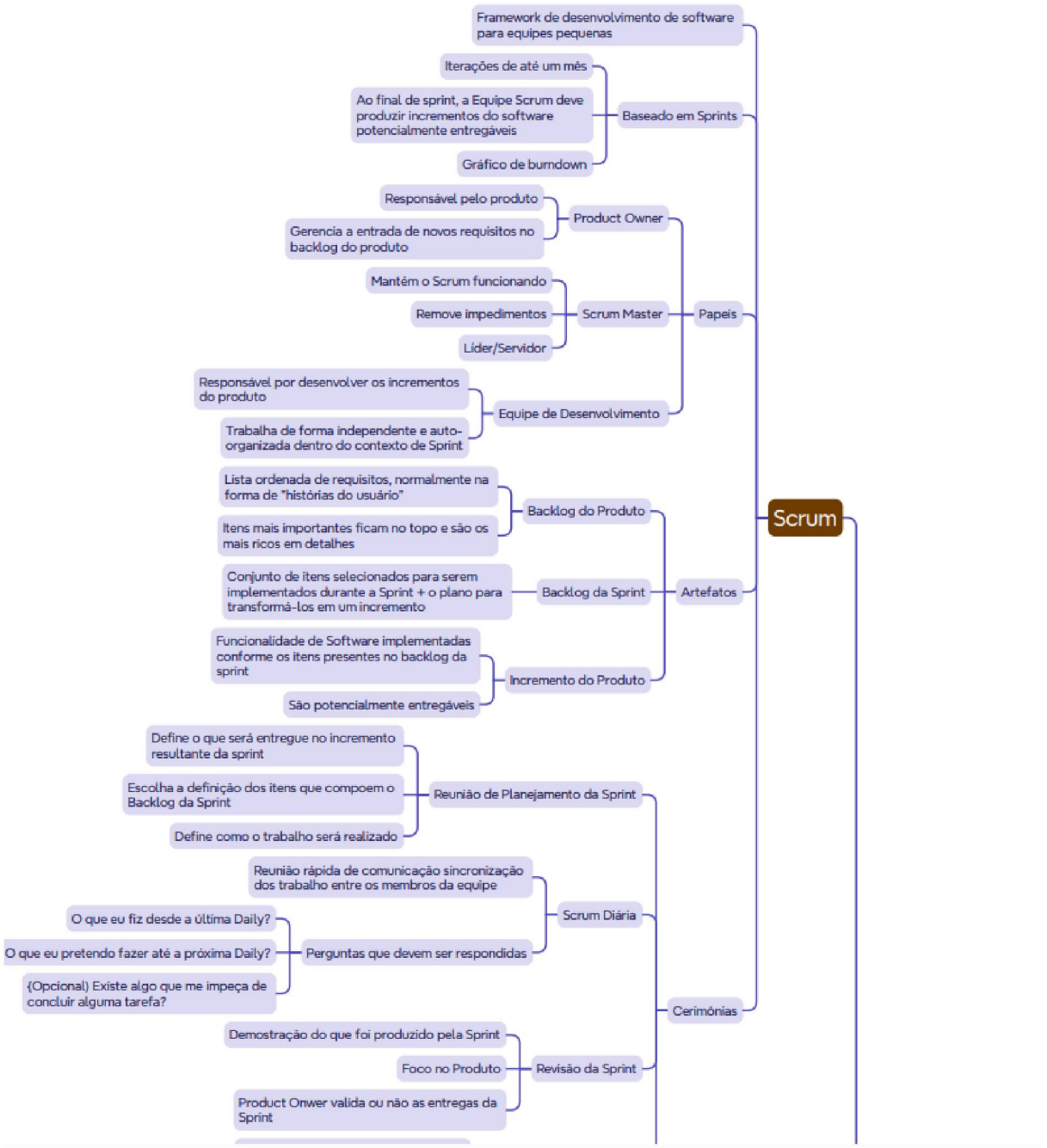
Fonte: O autor (2025)

Figura 3 – Parte do mapa mental centralizado com duas informações sobre princípios de entrega contínua e maturidade em processos de software para resumo de informações sobre Métodos Ágeis



Fonte: O autor (2025)

Figura 4 – Parte do mapa mental sobre SCRUM para resumo de informações sobre Métodos Ágeis



Fonte: O autor (2025)

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

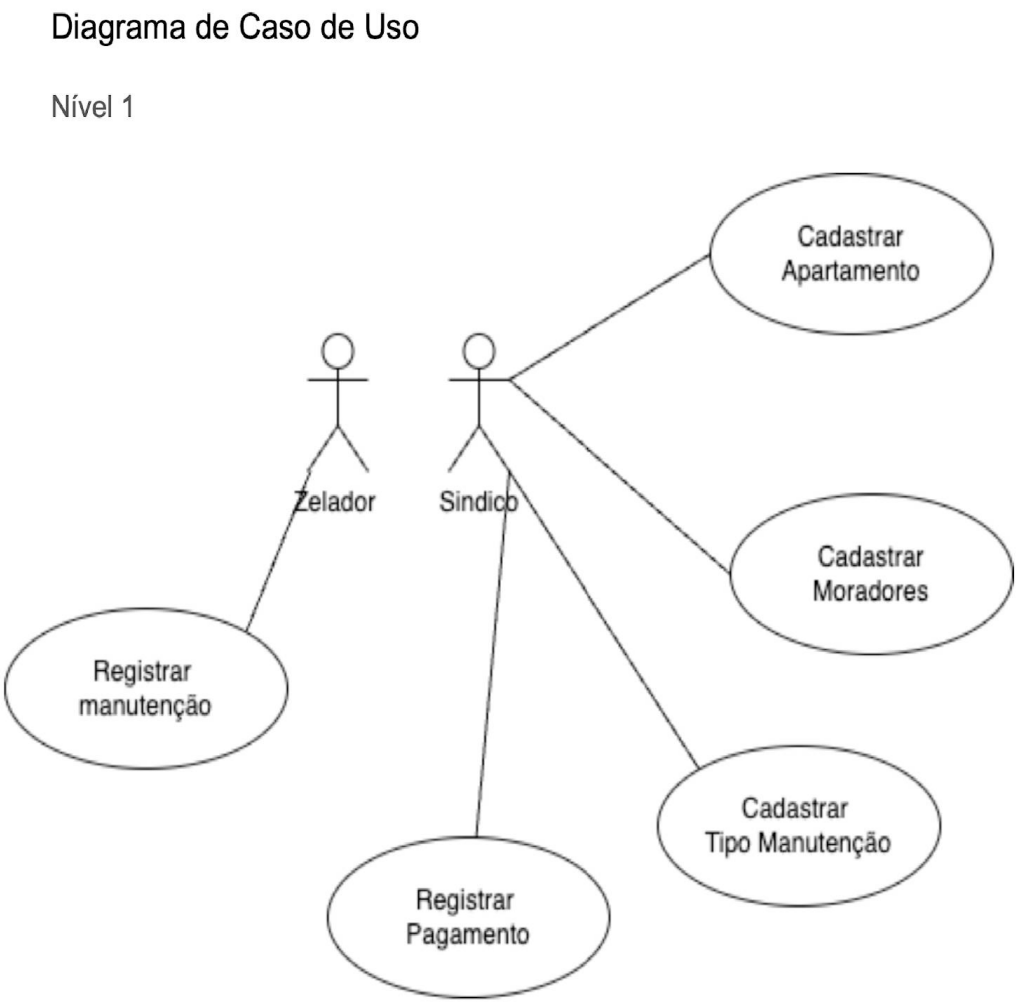
O enfoque dessa disciplina foi entender o conjunto integrado de diagramas UML (Unified Modeling Language), histórias de usuário dos quais transformam requisitos do *stakeholder* em um *backlog* executável, entre outros diagramas importantes para a modelagem. A modelagem é uma das tarefas de todo desenvolvimento e, neste caso, ela foca em modelagem ágil, com a importância de validar a qualidade do produto desenvolvido. A modelagem foi sendo modificada e evoluindo para que ela possa se adequar ao mercado, ou seja, fazer os diagramas e artefatos apenas na medida exata em que sejam úteis para a produção do software, gerando valor ao processo (Wojciechowski, 2021). O artefato criado como resultado da disciplina exemplifica a modelagem e incentiva a colaboração que sustenta o desenvolvimento ágil. Sendo que cada diagrama foi produzido para eliminar desperdício de documentação, garantindo ao mesmo tempo uma comunicação clara entre *stakeholders* e equipe de desenvolvimento.

Em conjunto com o desenvolvimento ágil, foi enfatizado o conceito de Domain-Driven Design (Evans, 2003), que centra o desenvolvimento em um modelo de domínio rico, elaborado colaborativamente com especialistas e delimitado em *bounded contexts*, garantindo que cada artefato produzido agregue valor direto ao software.

Dessa forma, a integração entre Modelagem Ágil, entre o suficiente de documentação e promove colaboração contínua, com domínio na resolução dos principais problemas para o cliente e eliminando o desperdício adjunto com a concentração de esforços para que agregar valor real ao produto. Em síntese, essa disciplina garante curtos ciclos de *feedbacks*, comunicação clara entre *stakeholders* e desenvolvedores e um código capaz de evoluir com segurança à medida que o mercado avança.

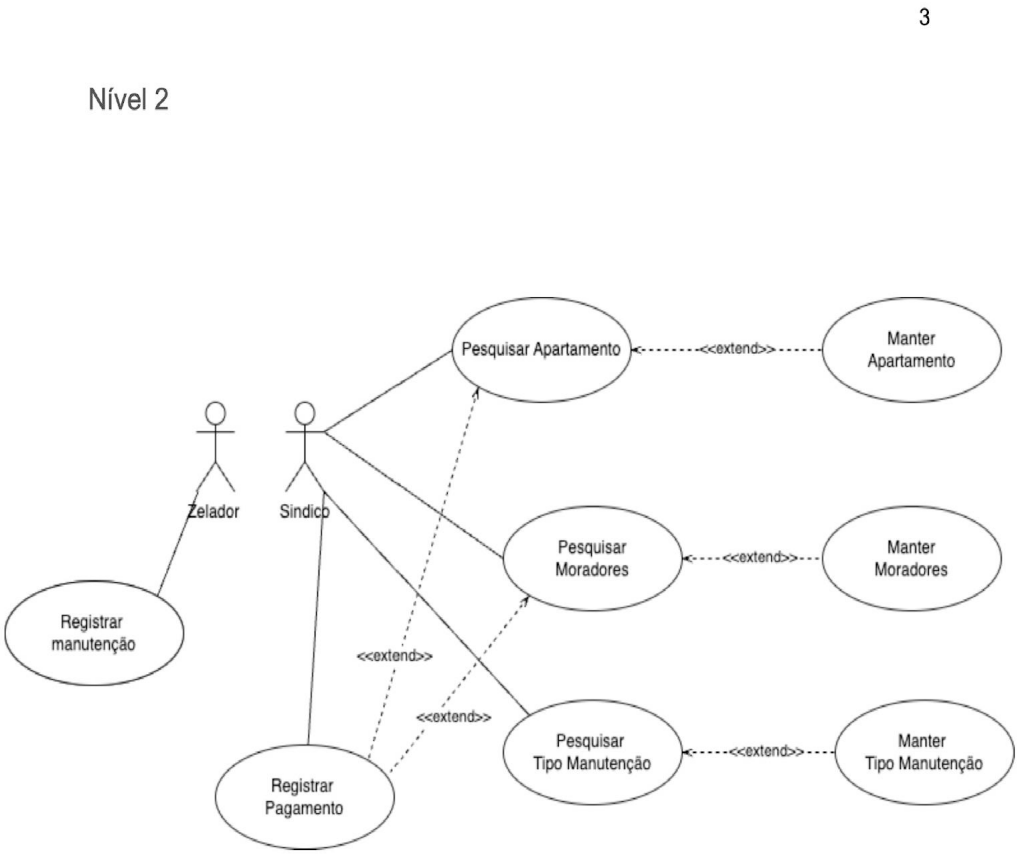
3.1 ARTEFATOS DO PROJETO

Figura 5 – Modelo de Diagrama de Casos de Uso de nível 1 para o sistema proposto



Fonte: O autor (2025)

Figura 6 – Modelo de Diagrama de Casos de Uso de nível 2 para o sistema proposto

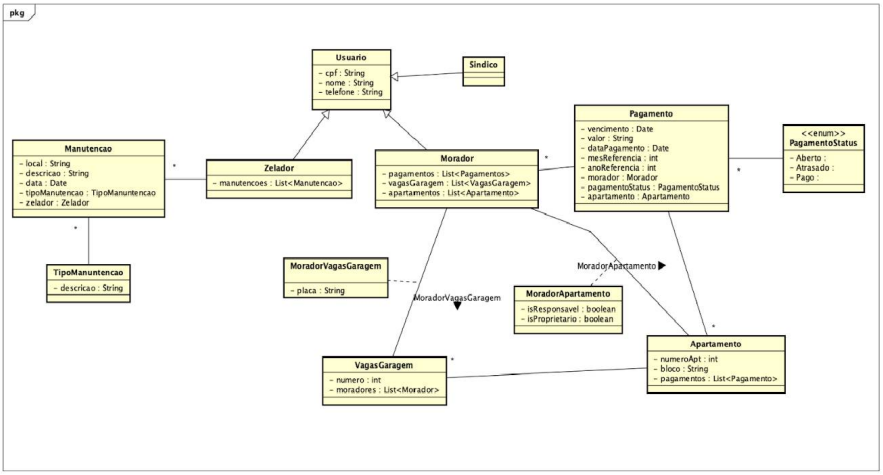


Fonte: O autor (2025)

Figura 6 – Modelo de Diagrama de classe para o sistema proposto

4

Diagrama de Classe



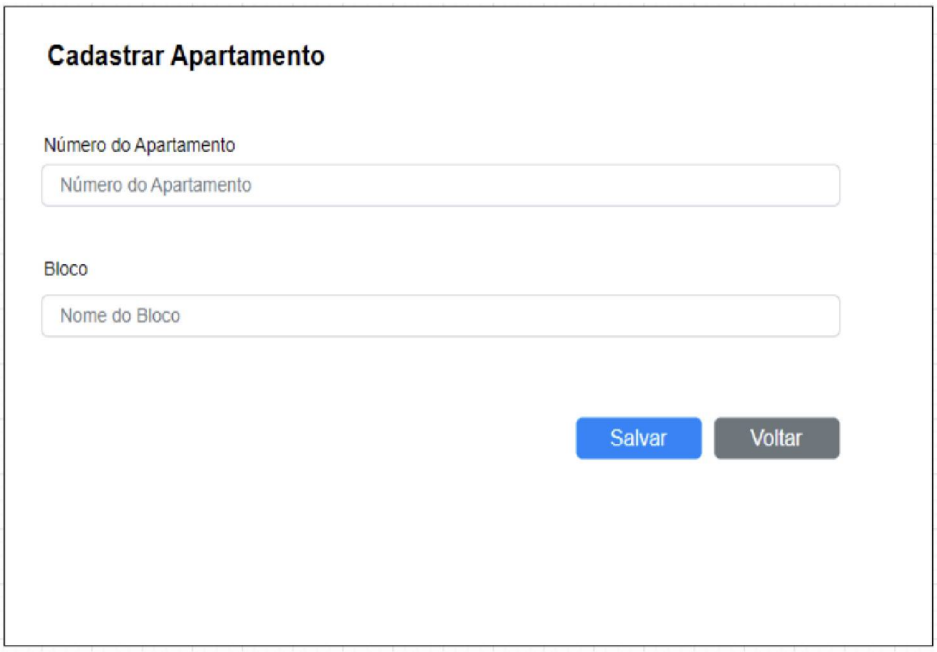
Fonte: O autor (2025)

Figura 7 – Requisito Funcional para o sistema proposto

HU002 – Manter Apartamento

Sendo	O Síndico
Quero	Manter os dados dos apartamentos
Para	Que seus dados fiquem atualizados

Desenho da Tela



Critérios de Aceitação

1. Deve receber o parâmetro da tela de pesquisa e configurar a tela
2. Deve voltar à tela anterior
    a. Critérios exclusivos para (Parâmetros Novos e Alterar)
3. Não deve prosseguir com número de apartamento em branco e único

Fonte: O autor (2025)

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

Na disciplina de gerenciamento materializou-se dois artefatos finais, que se complementam para a gestão ágil no centro do ciclo de vida do Sistema de Gestão de condomínio, criando assim um plano de *release* e testando na prática com um quadro Kanban, métricas e relatórios.

No plano de *release* (GAP1) a equipe calculou uma velocidade média de 4 pontos por *sprint* semanal, sendo 11 iterações para entregar 44 pontos de escopo. A abordagem segue as recomendações de planejamento incremental do Scrum Guide (Schwaber & Sutherland, 2020) e a técnica de *story-pointing* descrita por Cohn (2004), fazendo o escopo flexível e provendo a previsibilidade das tarefas. Já no quadro do Kanban, inspirado nos princípios de fluxo contínuo de Anderson (2010) e Kniberg & Skarin (2010), consegue observar o limite explícitos de WIP e acompanhamento por Cumulative Flow Diagram, Métricas de *lead time*, *cycle time* e velocidade, todas métricas sendo descritas por Dinardo (2020).

O projeto tem como princípio instruir o plano de *releases*, promovendo previsibilidade (PMI, 2017) e gestão visual do fluxo de trabalho através do Kanban como ferramenta e assim antecipando possíveis mudanças sem perder ritmo (Anderson, 2010). Esse processo garante os principais princípios do manifesto ágil, entrega contínua de software com valor e transparência, de forma padronizada tornando entregas frequentes com incremento de valor ao produto desenvolvido (Humble & Farley, 2011).

4.1 ARTEFATOS DO PROJETO

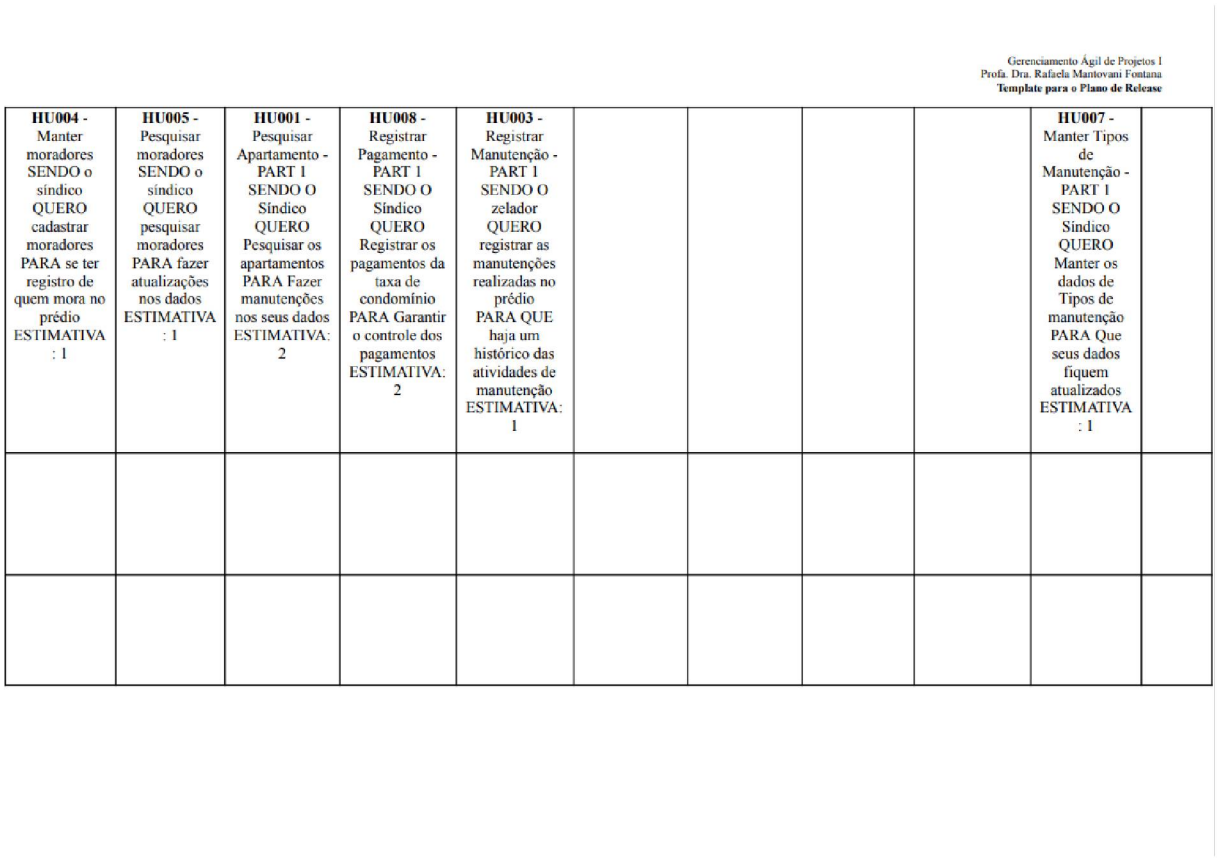
Figura 8 – *Template* para o plano de r*elease* das interações 1 até a interação 11

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

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4	Iteração/Sprint 5	Iteração/Sprint 6	Iteração/Sprint 7	Iteração/Sprint 8	Iteração/Sprint 9	Iteração/Sprint 10	Iteração/Sprint 11
Data Início: 06/05/24	Data Início: 13/05/24	Data Início: 20/05/24	Data Início: 27/05/24	Data Início: 03/05/24	Data Início: 10/06/24	Data Início: 17/06/24	Data Início: 24/06/24	Data Início: 01/06/24	Data Início: 08/07/24	Data Início: 15/07/24
Data Fim: 10/05/24	Data Fim: 17/05/24	Data Fim: 24/05/24	Data Fim: 31/05/24	Data Fim: 07/05/24	Data Fim: 14/06/24	Data Fim: 21/06/24	Data Fim: 28/06/24	Data Fim: 05/06/24	Data Fim: 12/07/24	Data Fim: 19/07/24
HU001 - Pesquisar Apartamento - PART 1 SENDO O Síndico QUERO Pesquisar os apartamentos PARA Fazer manutenções nos seus dados ESTIMATIVA: 3	**HU001 -** Pesquisar Apartamento - PART 2 SENDO O Síndico QUERO Pesquisar os apartamentos PARA Fazer manutenções nos seus dados ESTIMATIVA: 3	**HU002 -** Manter Apartamento - PART 1 SENDO O Síndico QUERO Manter os dados dos apartamentos PARA Que seus dados fiquem atualizados ESTIMATIVA: 2	**HU002 -** Manter Apartamento - PART 2 SENDO O Síndico QUERO Manter os dados dos apartamentos PARA Que seus dados fiquem atualizados ESTIMATIVA: 2	**HU002 -** Manter Apartamento - PART 3 SENDO O Síndico QUERO Manter os dados dos apartamentos PARA Que seus dados fiquem atualizados ESTIMATIVA: 3	**HU008 -** Registrar Pagamento - PART 2 SENDO O Síndico QUERO Registrar os pagamentos da taxa de condomínio PARA Garantir o controle dos pagamentos ESTIMATIVA: 3	**HU008 -** Registrar Pagamento - PART 3 SENDO O Síndico QUERO Registrar os pagamentos da taxa de condomínio PARA Garantir o controle dos pagamentos ESTIMATIVA: 5	**HU008 -** Registrar Pagamento - PART 4 SENDO O Síndico QUERO Registrar os pagamentos da taxa de condomínio PARA Garantir o controle dos pagamentos ESTIMATIVA: 3	**HU003 -** Registrar Manutenção - PART 2 SENDO O zelador QUERO registrar as manutenções realizadas no prédio PARA QUE haja um histórico das atividades de manutenção ESTIMATIVA: 4	**HU006 -** Pesquisar Tipo de Manutenção SENDO O Síndico QUERO Pesquisar os tipos de manutenção PARA Fazer a gestão dos seus dados ESTIMATIVA: 3	**HU007** - Manter Tipos de Manutenção - PART 2 SENDO O Síndico QUERO Manter os dados de Tipos de manutenção PARA Que seus dados fiquem atualizados ESTIMATIVA: 4

Fonte: O autor (2025)

Figura 9 – *Template* para o plano de r*elease* das interações 1 até a interação 11 tarefas HU004, HU005, HU001, HU008, HU003 e HU007



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

HU004 - Manter moradores SENDO o síndico QUERO cadastrar moradores PARA se ter registro de quem mora no prédio ESTIMATIVA : 1	**HU005 -** Pesquisar moradores SENDO o síndico QUERO pesquisar moradores PARA fazer atualizações nos dados ESTIMATIVA : 1	**HU001 -** Pesquisar Apartamento - PART 1 SENDO O Síndico QUERO Pesquisar os apartamentos PARA Fazer manutenções nos seus dados ESTIMATIVA: 2	**HU008 -** Registrar Pagamento - PART 1 SENDO O Síndico QUERO Registrar os pagamentos da taxa de condomínio PARA Garantir o controle dos pagamentos ESTIMATIVA: 2	**HU003 -** Registrar Manutenção - PART 1 SENDO O zelador QUERO registrar as manutenções realizadas no prédio PARA QUE haja um histórico das atividades de manutenção ESTIMATIVA: 1					**HU007 -** Manter Tipos de Manutenção - PART 1 SENDO O Síndico QUERO Manter os dados de Tipos de manutenção PARA Que seus dados fiquem atualizados ESTIMATIVA : 1	

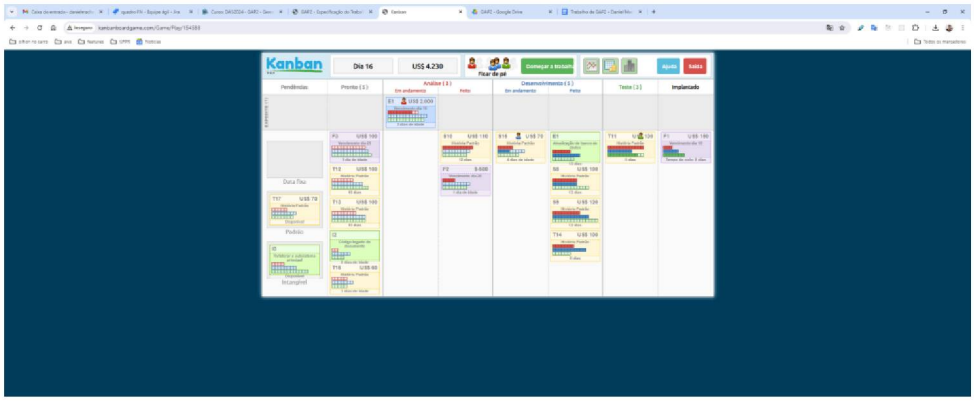
Fonte: O autor (2025)

Figura 10 – Plano de *release* com exemplo prático

Nome: Daniel Machado Pintos

Prints Solicitados:

dia 15



Fonte: O autor (2025)

Figura 11 – Gráfico de *burndown* do projeto

gráfico:

Fonte: O autor (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

Para o projeto dessa disciplina consistiu em implementar o *backend*, as funcionalidades, de um sistema bancário simplificado em Java, capaz de cadastrar clientes, gerenciar contas-correntes e aplicações de investimento, persistir transações em MySQL (banco de dados) via padrão DAO (Data Access Object) com JDBC, isolamento de lógica de persistência que reduz acoplamento e facilita refatorações, alinhando-se a boas práticas de camadas limpas – conceito originalmente formalizado em Core J2EE Patterns (Alur, Cruspi & Malks, 2023).

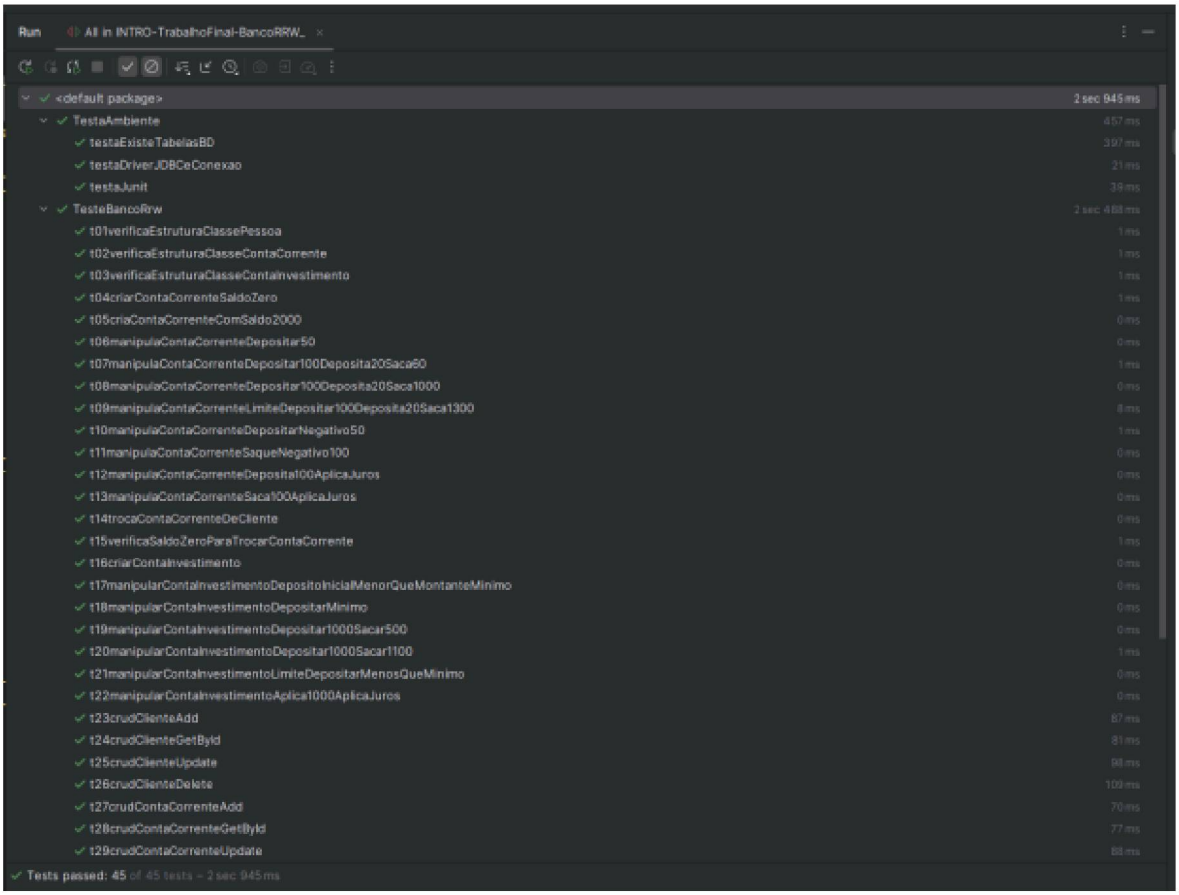
O professor forneceu 42 testes JUnit, ferramenta em Java para poder realizar TDD (Test-Driven Development), escrever testes antes da implementação de código, detectando defeitos mais cedos e produzindo design modular estudados por Beck (2003) e Daka & Fraser (2014). A meta era que pelo menos 40 dos testes passassem, tornando os testes verdes que resultaria em sucesso no teste. Além disso, o projeto foi configurado em um pipeline que compila, roda testes e gera relatórios a cada *push*, reforçando os princípios de entrega frequente e *feedbacks* rápidos.

Em avanço com a disciplina, havia um arquivo Dockerfile, uma imagem Docker para o banco de dados, que por sua vez é conteinerização de recursos que empacotam aplicações e todas suas dependências, permitindo que elas sejam executadas de forma idêntica em qualquer sistema operacional – Merkel (2014). E aplicando por sua vez o conceito de CI (integração contínua).

O projeto também utilizou princípios de Clean Code, para torná-lo um projeto limpo e orientado a objetos, conforme Deitel (2010) e Robert C. Martin (2008), promovendo legibilidade e manutenção — pilares de equipes ágeis sustentáveis. Ao combinar TDD, DAO, CI e Clean Code em um projeto de exemplo, sendo próximo e realista, a disciplina mostra os principais fundamentos de programação adjunto com o desenvolvimento ágil. Essa abordagem prepara o aluno para o mercado com uma visão prática, e com abordagens ágeis e padrões limpos de código, que serve para fundamento de todo o curso de Especialização.

5.1 ARTEFATOS DO PROJETO

Figura 12 – Testes com status de sucesso proposto pela disciplina



Fonte: O autor (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

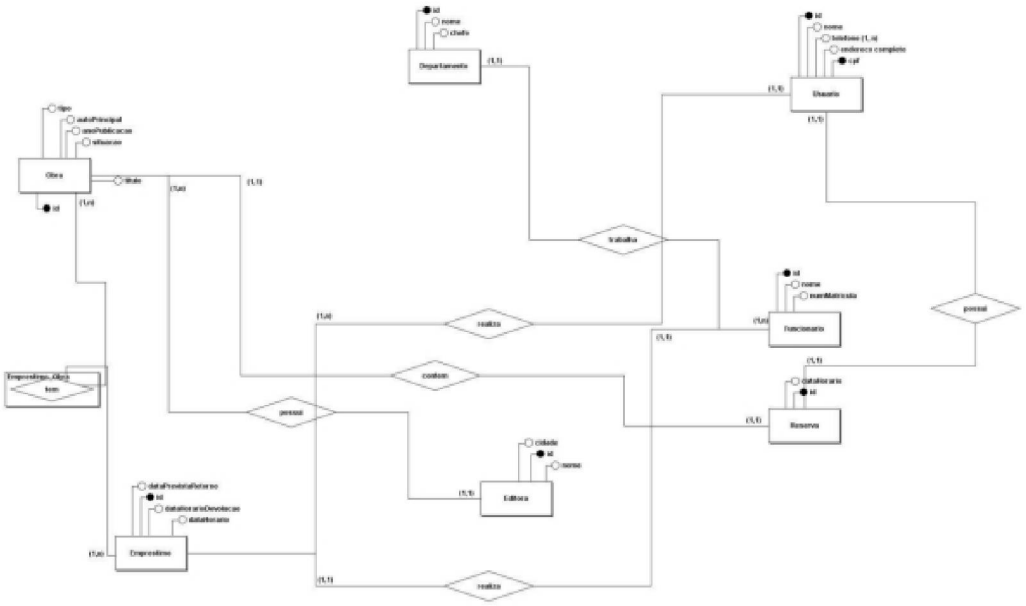
Na disciplina Banco de Dados (BD) a forma prática de concretização dos conhecimentos foi a realização de um projeto para um sistema de Controle de Vacinação e percorrendo todos os passos para a criação de um banco de dados, desde a modelagem partindo de um Modelo Entidade-Relacionado (MER), Esquema lógico relacional completo, com DDL testável e dados de exemplo assim como seus devidos *scripts* para geração. Também para garantir boas práticas descritas por Codd e Amber (1970) aplicou-se a prática de normalização até 3FN para reduzir redundâncias e garantir integridade.

A importância dessa disciplina é crucial para times ágeis que visam a implementação de dados, rapidez e garantia de mudanças frequentes assim abordado pelo modelo de Chen (1976) no qual para cada *sprint* é refinada o diagrama. O objetivo do projeto também era proteger dados críticos (pacientes, doses, campanhas) contra falhas em *deploys* contínuos e visando aplicação de Test Driven Database Development, para encurtar *feedbacks* e permitir que o modelo evolua a cada interação.

Dominar o modelos conceituais e lógicos, normalização e implementação do banco de dados empodera equipe ágeis a iterar sem comprometer a qualidade do dado. Desse modo, o aluno sai habilitado a evoluir esquemas em produção de forma segura, apoiar microsserviços orientados a dados e conseguir sustentar análises em tempo real, a partir do domínio de transações ACID – Atômicas, Consistentes, Isoladas e Duráveis.

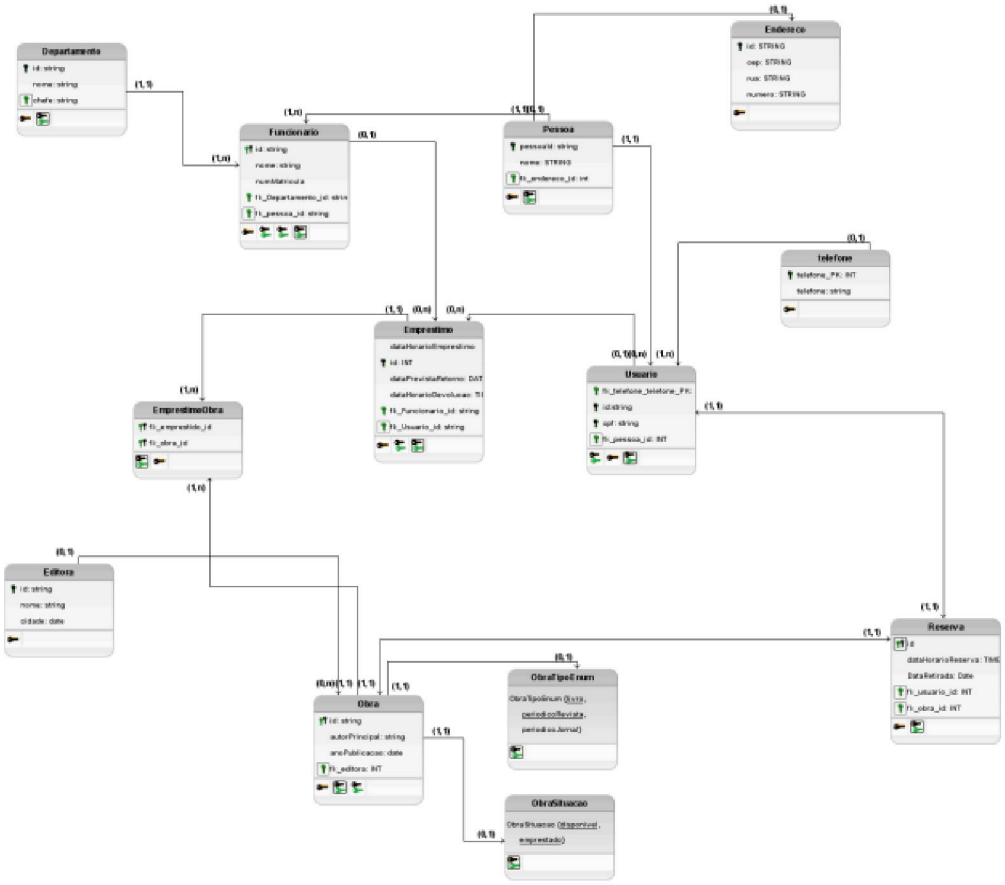
6.1 ARTEFATOS DO PROJETO

Figura 13 – Modelo de banco de dados relacional proposto



Fonte: O autor (2025)

Figura 14 – Modelo de banco de dados lógico proposto



Fonte: O autor (2025)

Figura 15 – Primeira parte dos comandos gerados para a criação da tabela Paciente e Vacina

```sql
-- Tabela Paciente
CREATE TABLE IF NOT EXISTS  Paciente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    data_nascimento DATE NOT NULL,
    endereco VARCHAR(255) NOT NULL
);

-- Tabela Vacina
CREATE TABLE IF NOT EXISTS  Vacina (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

Fonte: O autor (2025)

Figura 16 – Segunda parte dos comandos gerados para a criação da tabela Campanha, Aplicação e Dose

```
    nome VARCHAR(100) NOT NULL,
    fabricante VARCHAR(100) NOT NULL,
    doses_totais INT NOT NULL
);

-- Tabela Campanha
CREATE TABLE IF NOT EXISTS  Campanha (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    data_inicio DATE NOT NULL,
    data_fim DATE NOT NULL
);

-- Tabela Aplicacao
CREATE TABLE IF NOT EXISTS Aplicacao (
    id INT AUTO_INCREMENT PRIMARY KEY,
    paciente_id INT NOT NULL,
    vacina_id INT NOT NULL,
    dose_numero INT NOT NULL,
    data_aplicacao DATE NOT NULL,
    campanha_id INT,
    UNIQUE (paciente_id, vacina_id, dose_numero),
    FOREIGN KEY (paciente_id) REFERENCES Paciente(id),
    FOREIGN KEY (vacina_id) REFERENCES Vacina(id),
    FOREIGN KEY (campanha_id) REFERENCES Campanha(id)
);

-- Tabela Dose
CREATE TABLE IF NOT EXISTS Dose (
    id INT AUTO_INCREMENT PRIMARY KEY,
    vacina_id INT NOT NULL,
    aplicacao_id INT NOT NULL,
    numero INT NOT NULL,
    data_validade DATE NOT NULL,
    UNIQUE (vacina_id, aplicacao_id),
    FOREIGN KEY (vacina_id) REFERENCES Vacina(id),
    FOREIGN KEY (aplicacao_id) REFERENCES Aplicacao(id)
);
```

Fonte: O autor (2025)

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

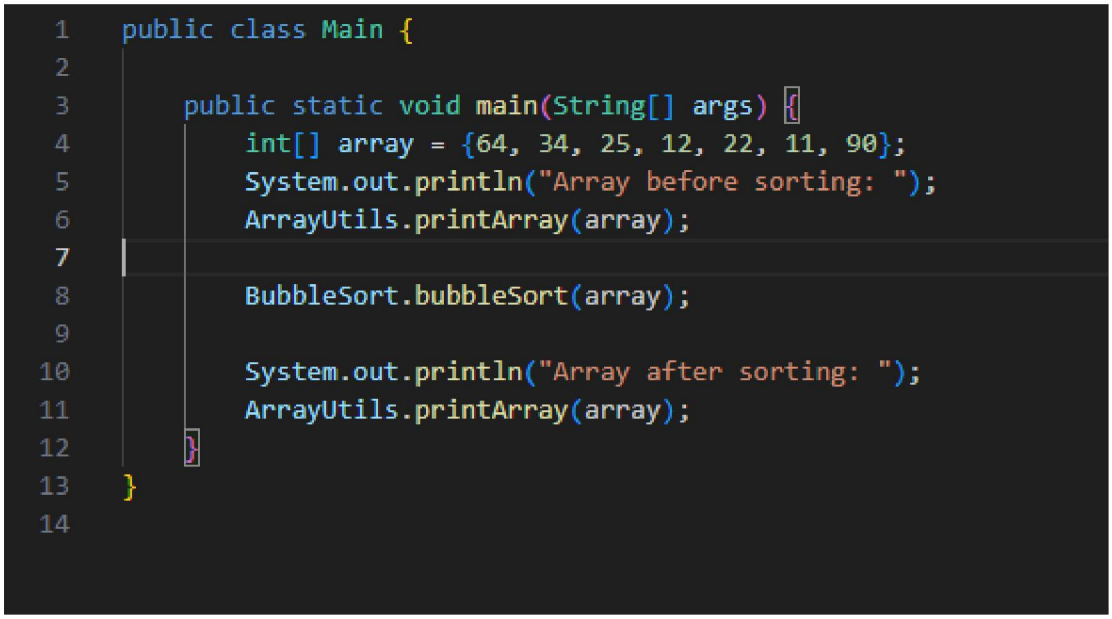
Em Aspectos Ágeis de Programação (AAP) essa disciplina trata de conjuntos de boas práticas de Clean Code, TDD, programação em Par, refatoração incremental, automação e entre outras abordagens para uma construção de um código que consiga ser bem escrito facilitando a manutenção ou até mesmo a criação de nova funcionalidade. Para exemplificação dessa matéria foi decidido refatorar o algoritmo BubbleSort em Java. E seguindo a abordagem de Kent Beck (2003), onde cada ciclo vermelho-verde-refatora que consiste em evoluir o código até passar 100% dos testes JUnit, que os mesmos foram criados antes da produção do código para garantir regressão.

Para o código final do projeto a ideia é seu foco ser em agilidade, adotando nomes significativos, funções pequenas e uma classe com responsabilidade única, essas práticas são o Clean Code (CC) em ação. Para melhorar a estrutura do código, foi realizada a extração de métodos, encapsulamento e eliminação do código duplicado seguindo os padrões propostos de Fowler (2018). E como proposto no trabalho, para possíveis melhorias no futuro do projeto, poderia também as seguintes abordagens: Programação em Par (sessões ping-pong – alternando entre motorista e navegador a cada teste) e Automação (pipelines de integração contínua que compila e executa os testes a cada *push*, gerando métricas de cobertura que alimentam retrospectivas de qualidade).

AAP em sua essência demonstra a qualidade interna do código, que são métricas de código limpo e testável minimizando a dívida técnica, permitindo que times ágeis entreguem valor contínuo de forma sustentável. Ao praticar refatoração contínua dirigida por testes em pares, o código ganha motor de velocidade como trata Kent Beck (2003). E assim para o projeto simples que foi a exemplificação da disciplina é possível colocar os assuntos teóricos em exemplos práticos.

7.1 ARTEFATOS DO PROJETO

Figura 17 – Classe Main do Projeto para execução dos códigos



```
1   public class Main {
2
3       public static void main(String[] args) {
4           int[] array = {64, 34, 25, 12, 22, 11, 90};
5           System.out.println("Array before sorting: ");
6           ArrayUtils.printArray(array);
7
8           BubbleSort.bubbleSort(array);
9
10          System.out.println("Array after sorting: ");
11          ArrayUtils.printArray(array);
12      }
13  }
14
```

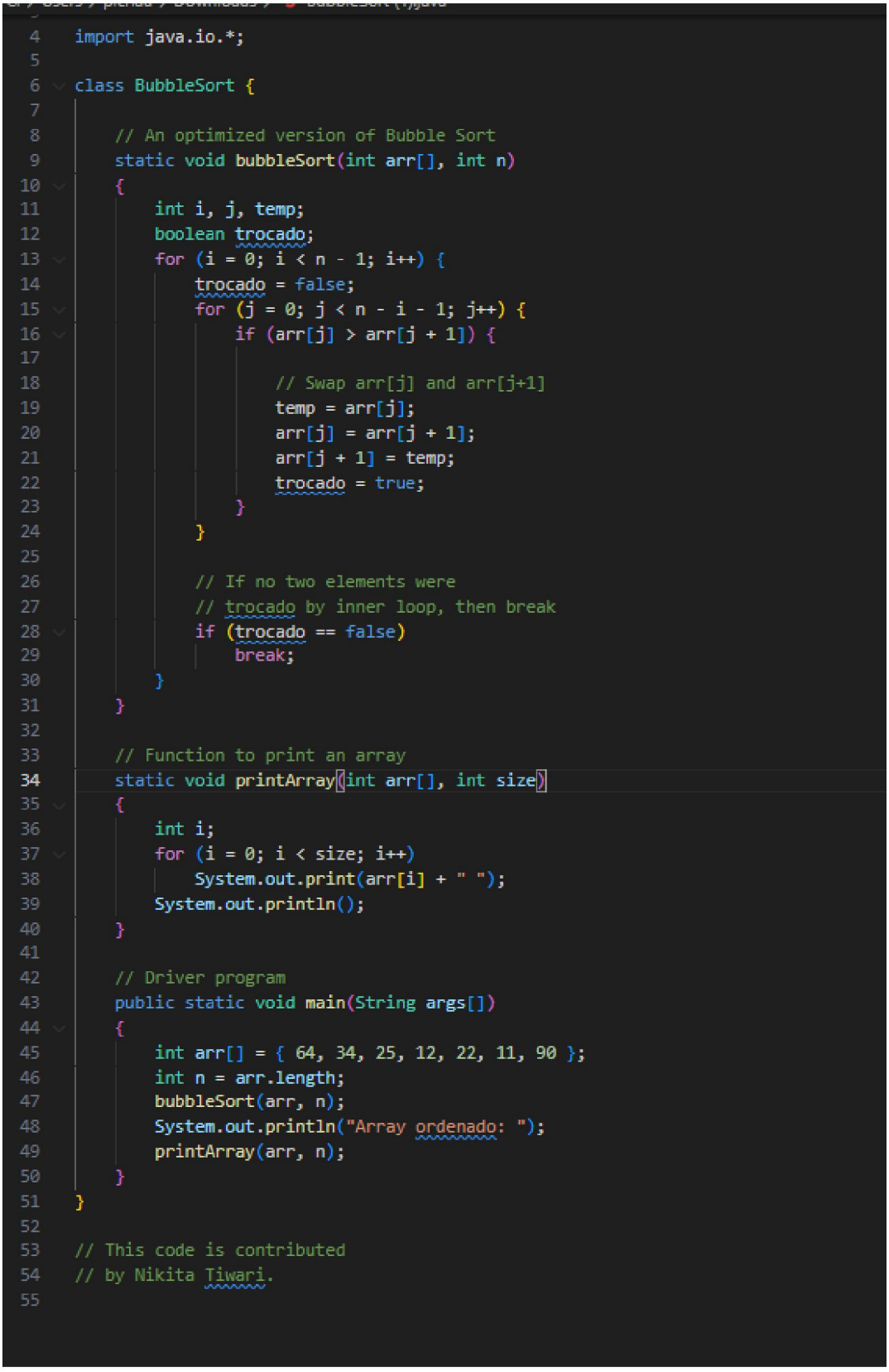
Fonte: O autor (2025)

Figura 18 – Classe para testar do projeto para realizar os testes com o JUnit

```java
import org.junit.Test;
import static org.junit.Assert.assertArrayEquals;

public class BubbleSortTest {

    @Test
    public void testBubbleSort() {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        int[] sortedArr = {11, 12, 22, 25, 34, 64, 90};
        BubbleSort.bubbleSort(arr);
        assertArrayEquals(sortedArr, arr);
    }

    @Test
    public void testEmptyArray() {
        int[] arr = {};
        int[] sortedArr = {};
        BubbleSort.bubbleSort(arr);
        assertArrayEquals(sortedArr, arr);
    }

    @Test
    public void testSingleElementArray() {
        int[] arr = {1};
        int[] sortedArr = {1};
        BubbleSort.bubbleSort(arr);
        assertArrayEquals(sortedArr, arr);
    }

    @Test
    public void testAlreadySortedArray() {
        int[] arr = {1, 2, 3, 4, 5};
        int[] sortedArr = {1, 2, 3, 4, 5};
        BubbleSort.bubbleSort(arr);
        assertArrayEquals(sortedArr, arr);
    }

    @Test
    public void testReverseSortedArray() {
        int[] arr = {5, 4, 3, 2, 1};
        int[] sortedArr = {1, 2, 3, 4, 5};
        BubbleSort.bubbleSort(arr);
        assertArrayEquals(sortedArr, arr);
    }
}
```

Fonte: O autor (2025)

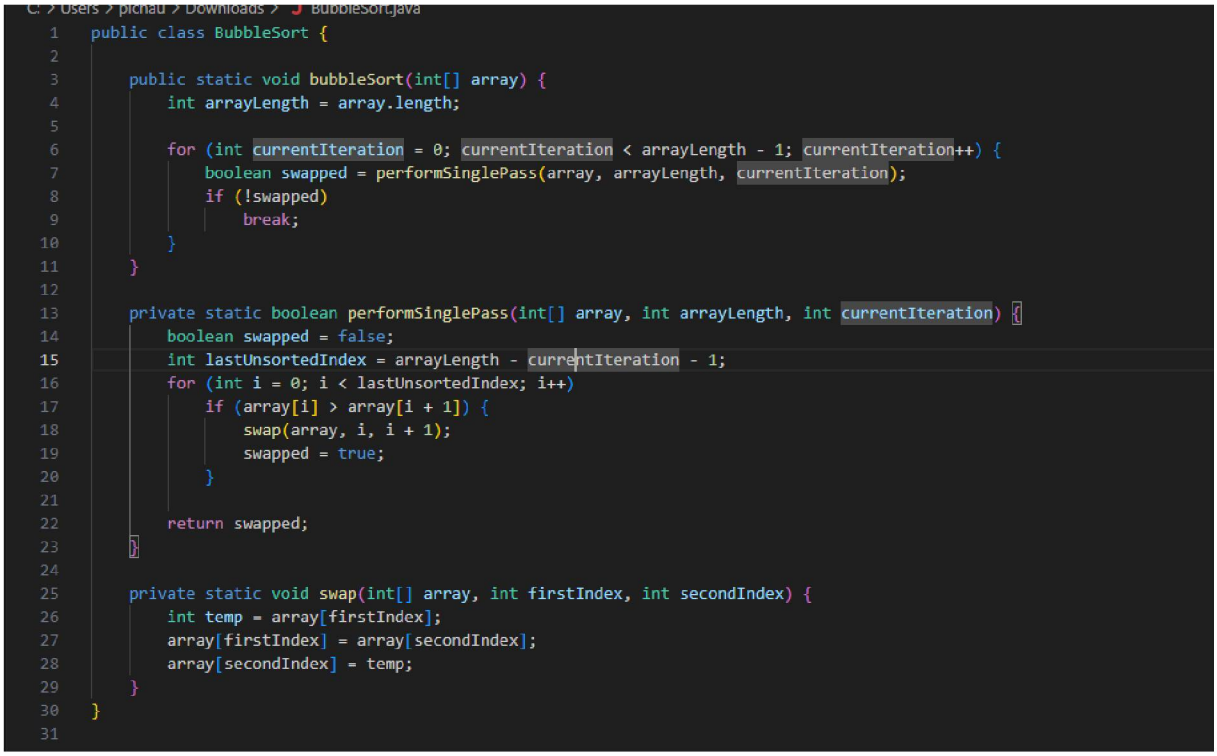
Figura 19 – Classe fornecida pelo professor da disciplina (Bubble Sort)



```java
import java.io.*;

class BubbleSort {

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

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

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

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

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

// This code is contributed
// by Nikita Tiwari.

```

Fonte: O autor (2025)

Figura 20 – Classe refatorada pelo aluno seguindo os padrões da disciplina



Fonte: O autor (2025)

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

Um dos requisitos para o mercado de trabalho contemporâneo é a formação do conhecimento de desenvolvimento web, onde cada dia mais se torna a necessidade de ter domínios das técnicas para criação de requisitos dos Stakeholders em algo concreto. Para isso, as disciplinas Desenvolvimento Web 1 e 2 foram inseridas na grade curricular do aluno. Onde o foco consiste no ciclo de desenvolvimento de uma aplicação prática de um protótipo *frontend* na tecnologia Angular, do qual seria capaz de gerenciar o conceito CRUD (Create, Read, Update, Delete) da ideia de aluno, curso e matrículas.

O foco da disciplina seria a iteração incremental orientada ao valor, no qual foi dividida em duas frentes, na primeira parte (Web1) tem a orientação de criar uma interface e o sistema para Alunos e Cursos. Usando a tecnologia Angular no backend para implementações de funcionalidades lógicas, Bootstrap para interfaces padronizadas e intuitivas no *frontend*, e o Local Storage para salvar os dados do usuário sem a necessidade de integração com o banco.

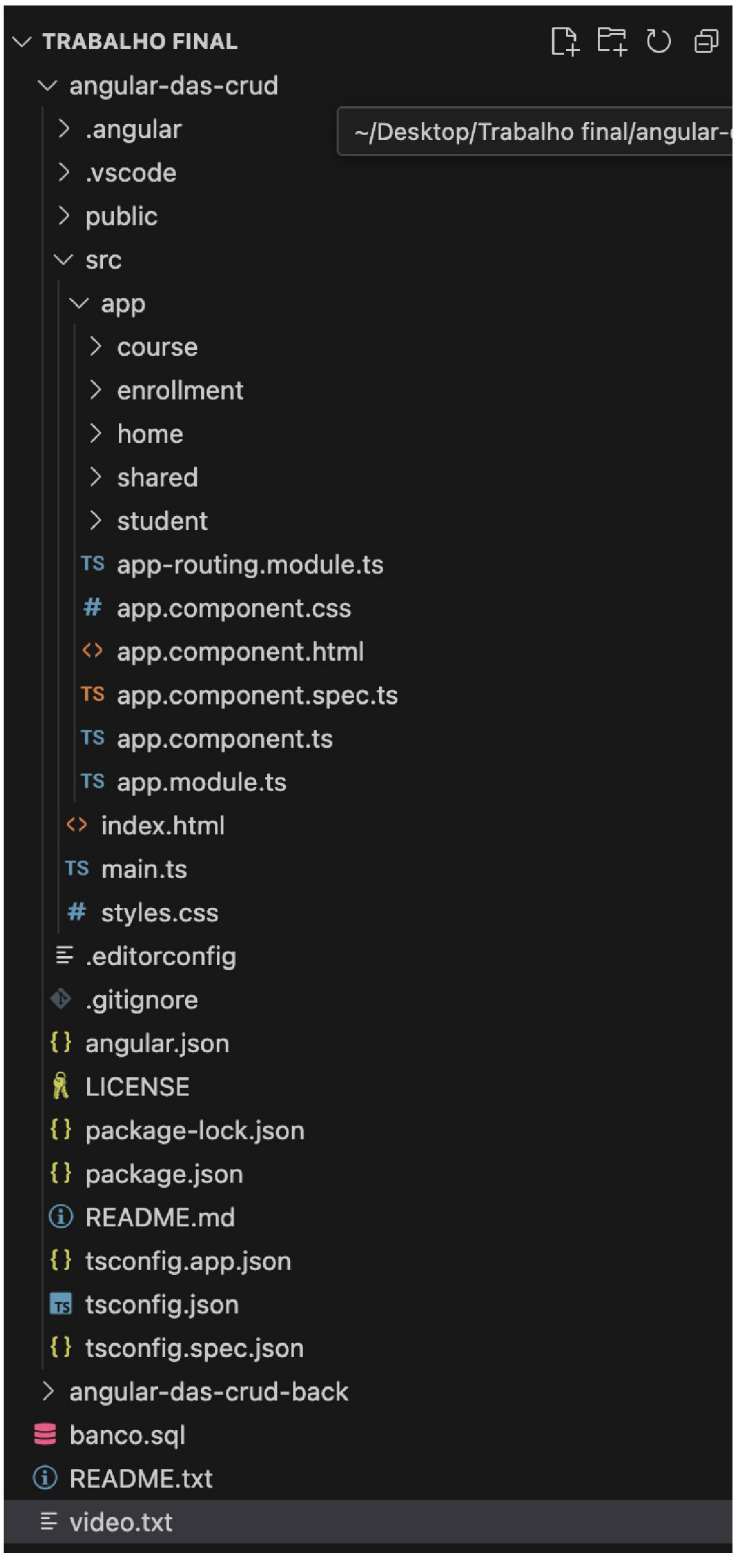
Já na segunda parte da disciplina (Web2) foi ampliado o escopo para conter o CRUD de Matrícula e migrar os dados do *backend* para Sprint Boot, Spring Data JPA e Postgres, que são padrões de mercado para desenvolvimento web. Nesse mesmo conceito foi adotado teste de integração e *scripts* SQL versionados como aponta o Beck (2001) onde ressalta o valor de software funcionando em vez de ter uma documentação abrangente.

O projeto comprova a incrementalidade, *feedback* rápido e automação ao acelerarem a construção de sistema escalável e robusto para atender as demandas do mercado atual. E também adota práticas ágeis como Thin Vertical Slices, começar entregando fatias verticais conectadas à interface (Hunt & Thomas, 2020), utilizar de TDD para facilitar a validação e credibilidade de um bom código, reúso de componentes e refatoração continuação (Fowler, 2018) para eliminar duplicação.

E a disciplina tem integração íntima com outras disciplinas do curso, como, por exemplo, no esquema de Aluno-Curso-Matrícula derivada do MER, normalizada e entendida na disciplina anterior de Banco de Dados (BD). No caso a disciplina funciona como exemplo prático dos conceitos vistos anteriormente e por este motivo seu valor acadêmico e profissional é elevado.

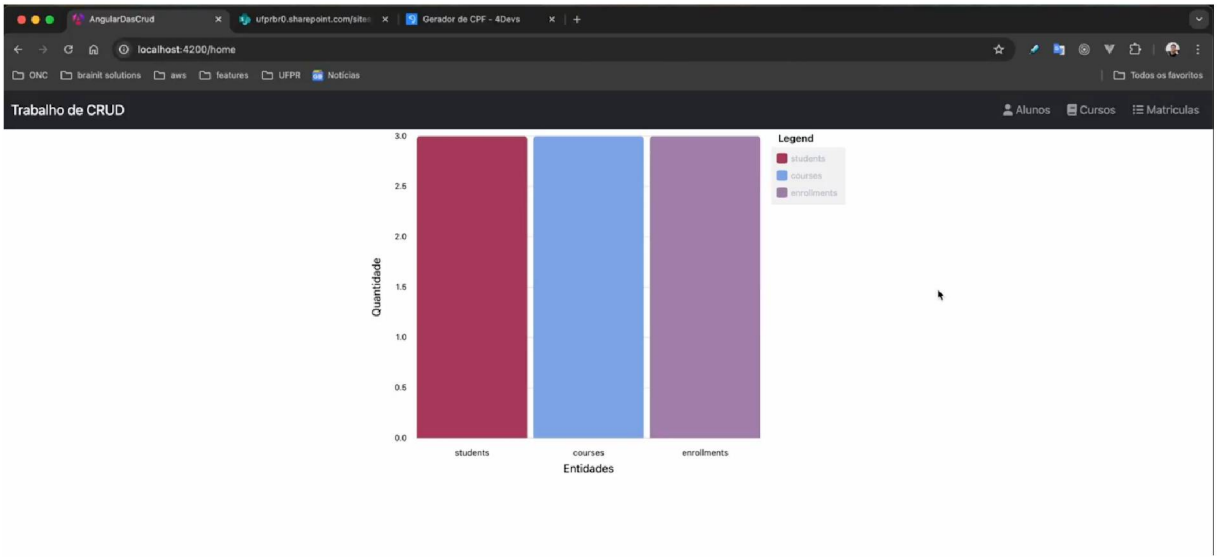
8.1 ARTEFATOS DO PROJETO

Figura 21 – Estrutura de pastas do projeto no ambiente de desenvolvimento



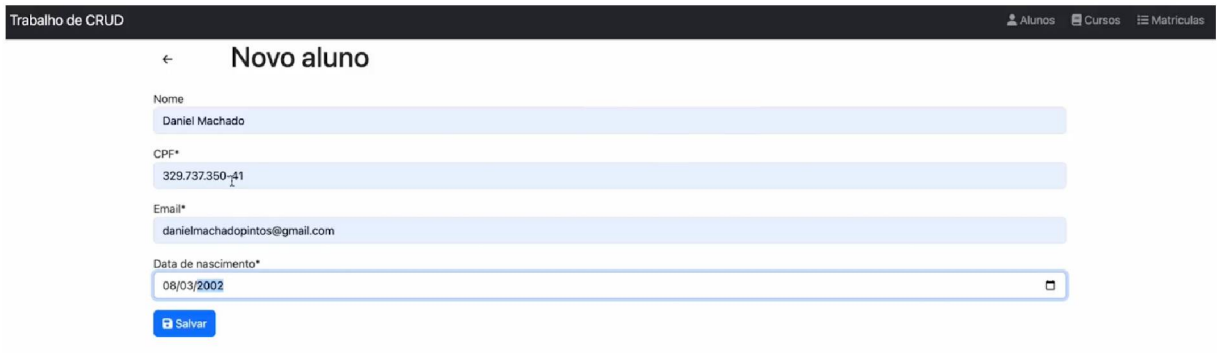
Fonte: O autor (2025)

Figura 22 – Tela de Início do projeto em ambiente de desenvolvimento



Fonte: O autor (2025)

Figura 23 – Tela de Cadastro de Aluno em ambiente de desenvolvimento



Fonte: O autor (2025)

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

O UX (User Experience) existe há décadas no mercado, onde o usuário é o centro e a disciplina concentra-se em criar produtos e serviços que ofereçam experiências significativas e satisfatórias para atender necessidades ou desejos do usuário tendo em vista uma forma eficaz e agradável de resolver o mesmo (Curcio, 2021). A abordagem de Design Centrado no Usuário (UCD) é fundamental para a mesma, onde coloca as necessidades do usuário como enfoque e que serve de base para o projeto da disciplina.

O projeto consiste em criar um aplicativo ou site fundamentado nos ensinamentos da disciplina, justificar sua função e seu valor de mercado, desenvolver 5 telas e explicar a escolha de cor, layout, fonte entre outros pontos, que foram trabalhados na disciplina. Dando um enfoque constante em como seria a experiência do usuário em relação ao poder de decisão, agradabilidade visual, intuitividade, e o principal que seria o *feedback*. Utilizando esses conceitos foi demonstrado para um possível usuário do produto, sendo um site de busca treinadores, e apontando correções e melhorias que poderiam ser feitas.

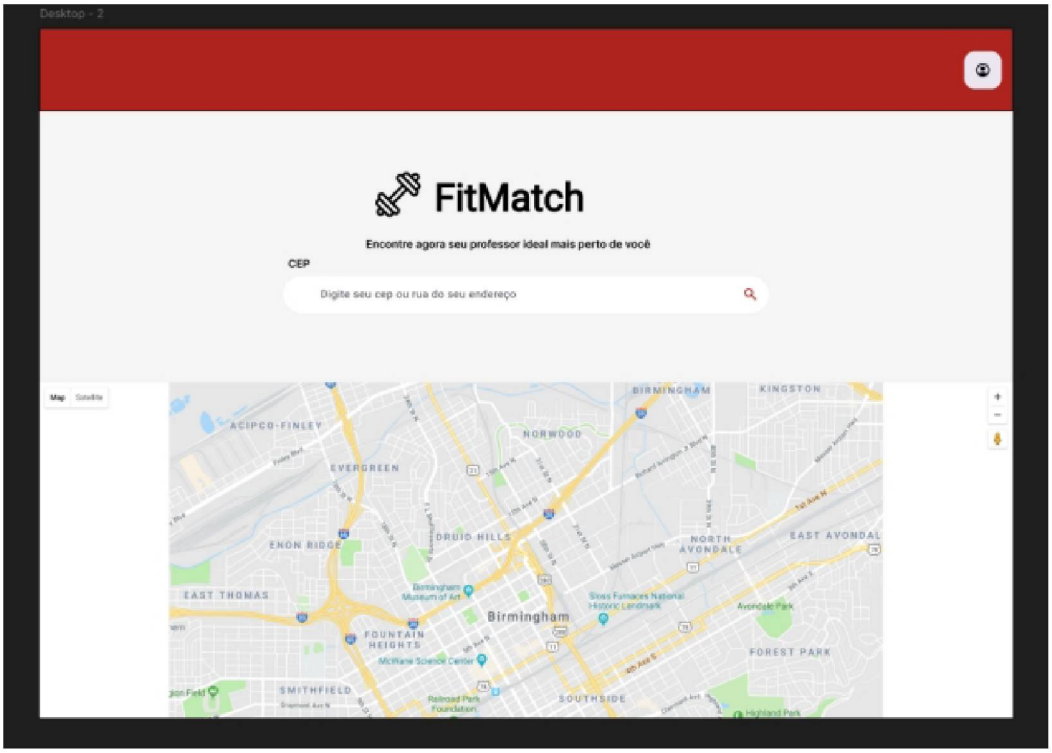
O desenvolvimento do projeto, resume-se na tentativa de procurar um *personal trainer* acessível a rotina, orçamento e experiência do usuário. Como sendo um projeto de 5 páginas, sendo só o essencial para o usuário, sendo um MVP (Minimum Viable Product) - a versão mais simples e enxuta de um produto (Zanette, 2024).

Para os padrões de mercado é indispensável o estudo de UX nos dias atuais, segundo Camara (2023) “uma boa experiência do usuário reduz as taxas de abandono dos carrinhos [...] mantendo o seu cliente engajado na compra”. Camara afirma que é o refinamento do produto, sendo que pode ser um produto criado ou um produto que passará por melhorias. E a disciplina relaciona-se com outros tópicos justamente por trazer essa visão de usuário, atendendo as demandas e desejos do *stakeholder*. Relaciona intimamente com a disciplina WEB, pois é o arcabouço e refinamento do mesmo.

As disciplinas de Desenvolvimento Mobile 1 e 2 foram cruciais para aplicar o desenvolvimento ágil na prática. O projeto resultou em um aplicativo funcional, destacando a importância de *feedbacks* rápidos e entregas incrementais. Este projeto integrou conhecimentos de MAG1, MAG2, GAP1, GAP2, INTRO, BD, AAP, WEB1, WEB2, UX, INFRA e TEST, consolidando o aprendizado de diversas áreas para o desenvolvimento ágil e contínuo.

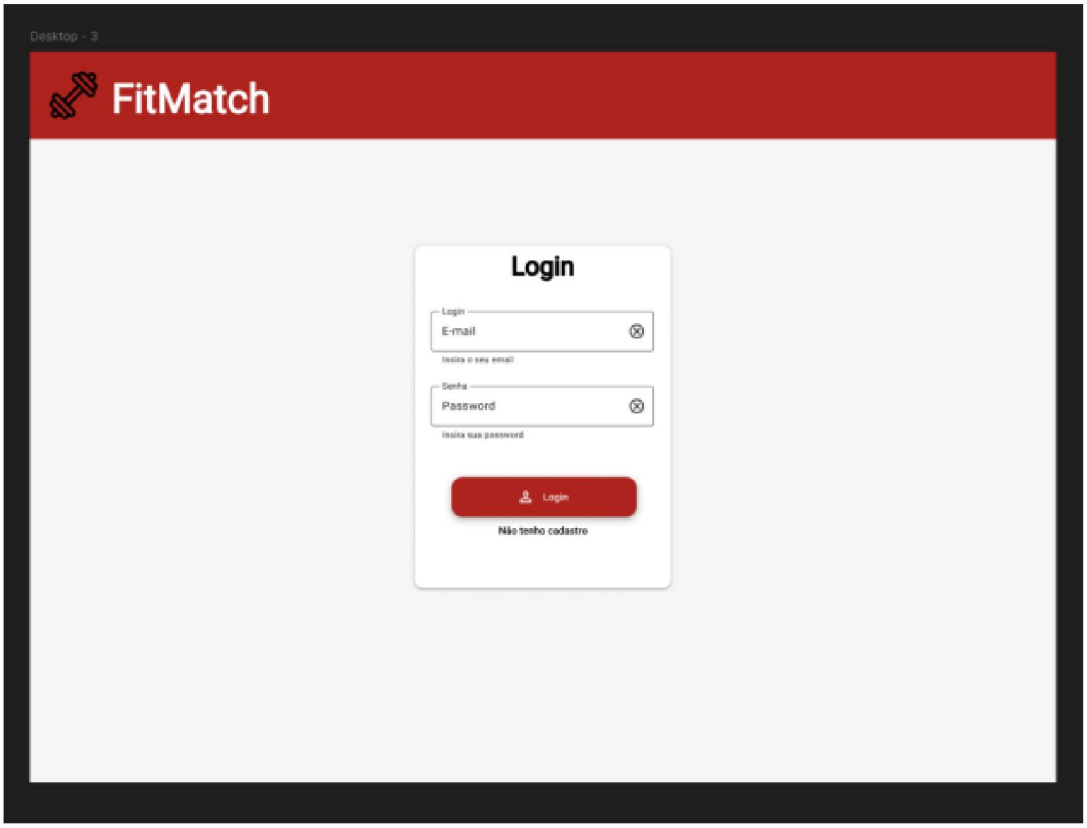
9.1 ARTEFATOS DO PROJETO

Figura 24 – Tela da Dashboard do projeto Fit Match



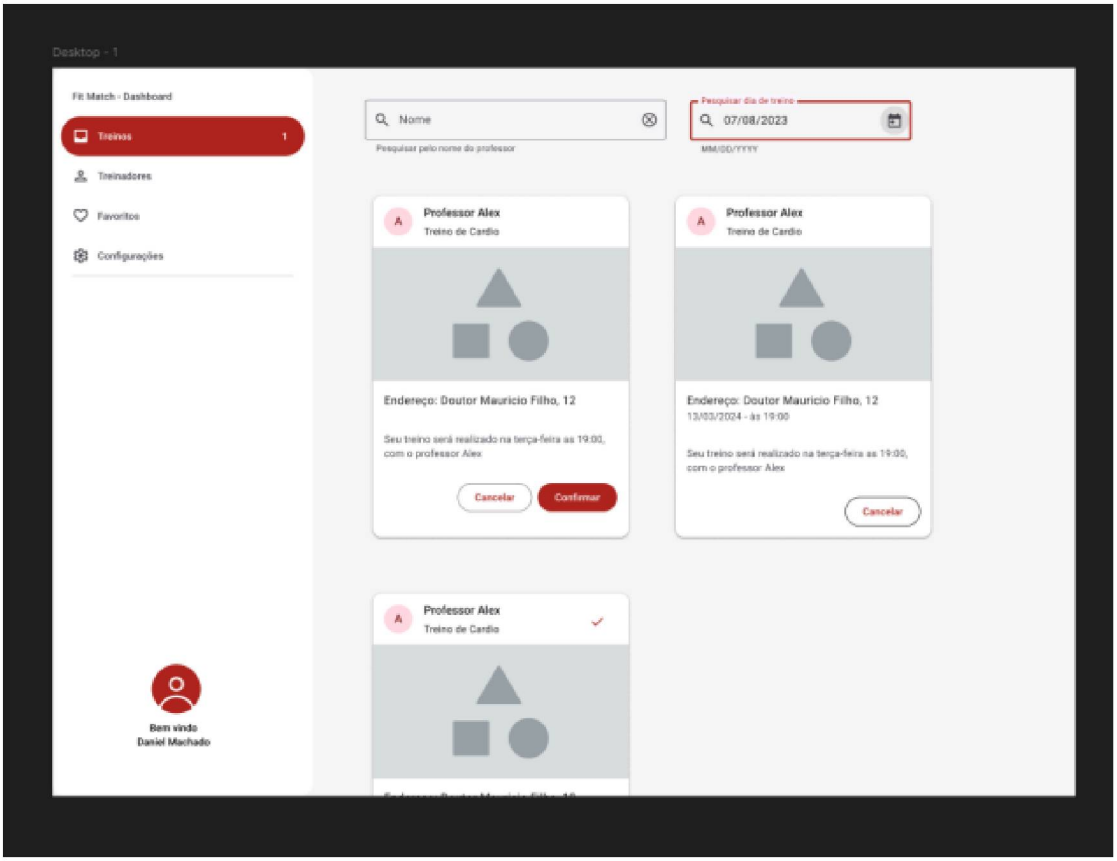
Fonte: O autor (2025)

Figura 25 – Tela de Login do projeto Fit Match



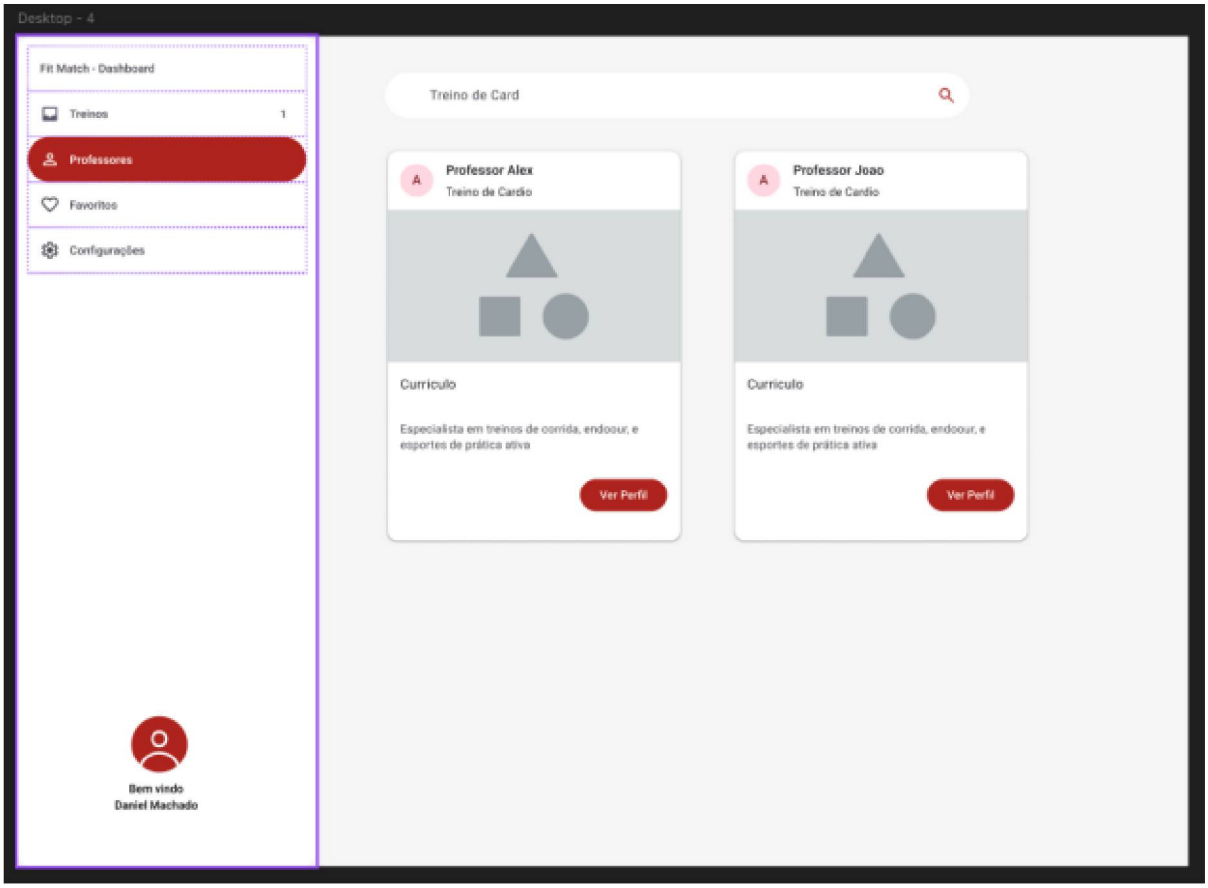
Fonte: O autor (2025)

Figura 26 – Tela de Pesquisa no projeto Fit Match por personal trainer com seletor de data e nome e visualização do professor



Fonte: O autor (2025)

Figura 27 – Tela de Pesquisa no projeto Fit Match por personal trainer com filtro nome



Fonte: O autor (2025)

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

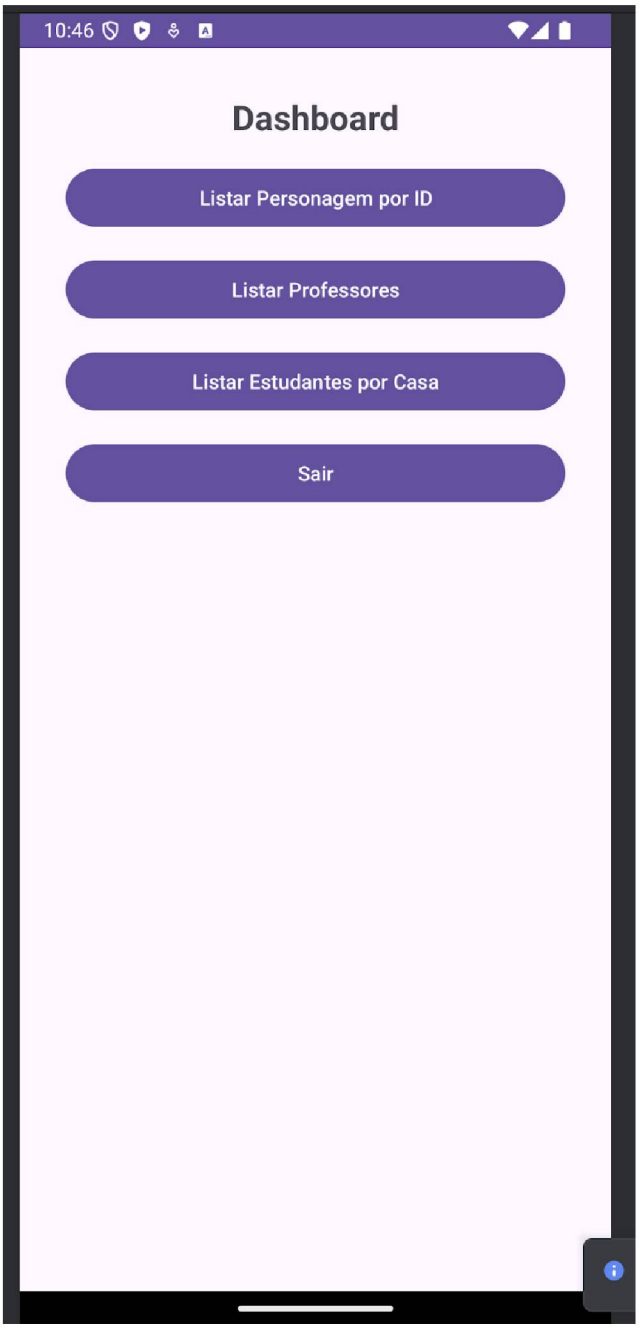
Nos módulos MOB1(Desenvolvimento Mobile I) e MOB2(Desenvolvimento Mobile II) consiste na prática de desenvolver aplicações mobiles em apps Android utilizando Kotlin, essas aplicações com aplicabilidade no mercado atual, foram inúmeras, desde de projetos iniciais para entender a estrutura de projetos mobiles até aplicações onde era possível realizar a persistência no banco de dados local (SQLite), aprender também sobre listas dinâmicas utilizando de Recyclerview, método que usa integração com o *storage* interno para atualizar os itens.

Na disciplina MOB1 foram feitas interações curtas no modelo de *mini-app* como o *story slice*, UI mínima + regra de negócio fechada (Kniberg, 2010). É adotado o MVVM (View Modal + Live Data) para separar a UI de lógica, e assim ajudando na refatoração e testes (Glauber, 2022). Já para a disciplina MOB2 foi introduzida FileStorage e SQLite, em migrações de *schema* que foram versionadas para suportar o *deploy* contínuo, fundamentado na disciplina de banco de dados (BD). Seguindo os padrões e princípios de Martin (2008) nos projetos foram utilizadas de componentes reutilizáveis e Clean Kotlin, que são diretivas para tornar o código conciso.

O projeto final da disciplina MOB2 que é a síntese das duas matérias, foi a criação de uma aplicação que utiliza se dos conhecimentos sobre *web services*, que faz correlação com a matéria de WEB1, e também conhecimentos com o Corrotinas, tendo uma interface que fosse minimamente intuitiva para o usuário. E pudesse enquadrar os aprendizados da disciplina estudada. E ao percorrer UI, lógica, persistência e distribuição em 6 aplicativos incrementais o aluno tem capacidade de trabalhar em projetos que tenham o Lean Startup Loop (Ries, 2011) – construir → medir → aprender. Que são habilidades essenciais para um time que utiliza de métodos ágeis mobile.

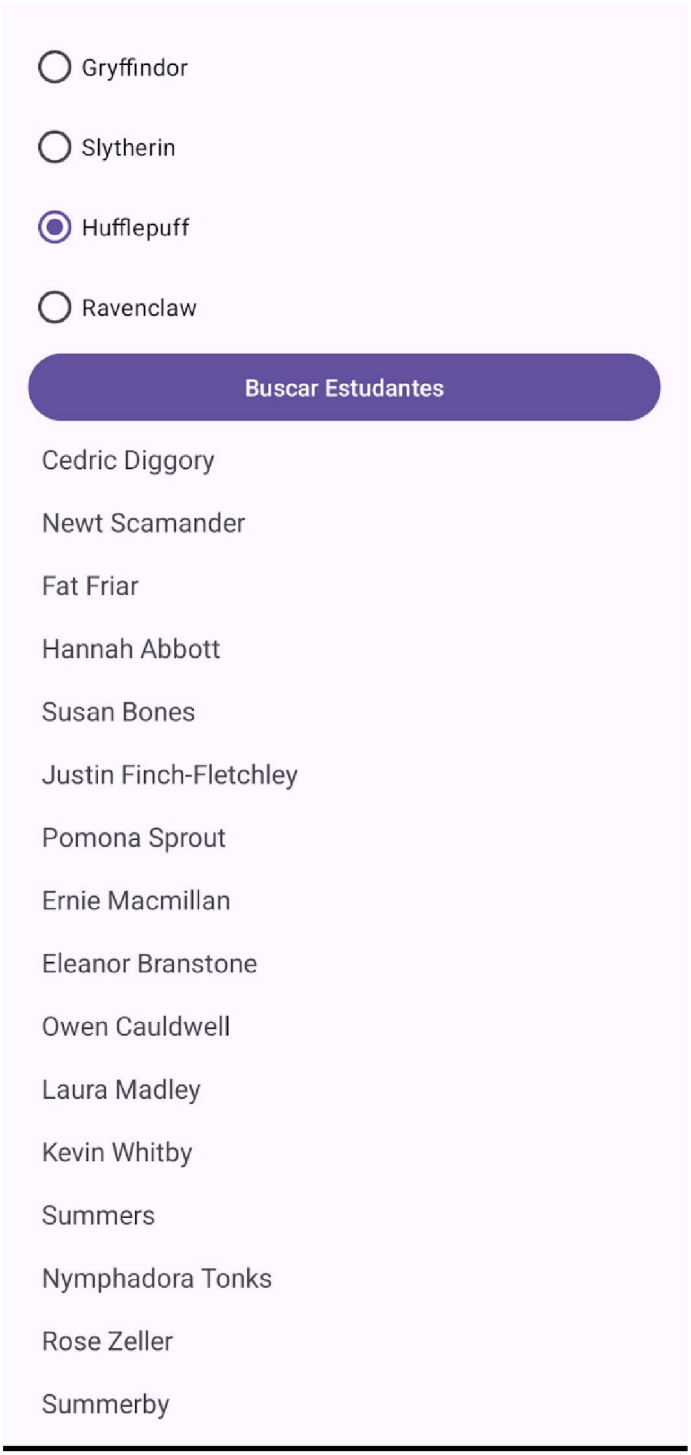
10.1 ARTEFATOS DO PROJETO

Figura 28 – Tela mobile do projeto da disciplina, para poder listar professores, listar estudantes por casa, listar personagem por ID e sair do aplicativo



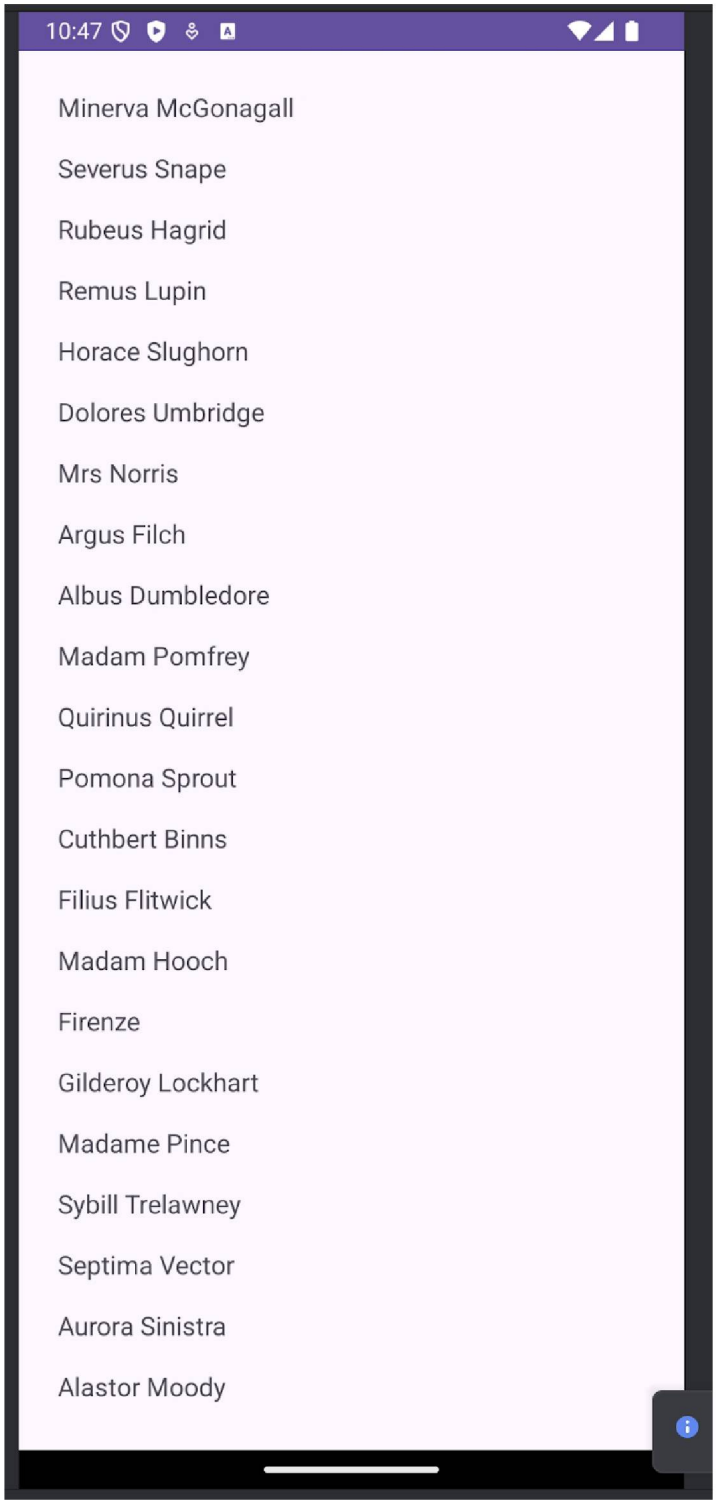
Fonte: O autor (2025)

Figura 29 – Tela mobile para poder listar estudantes por casa



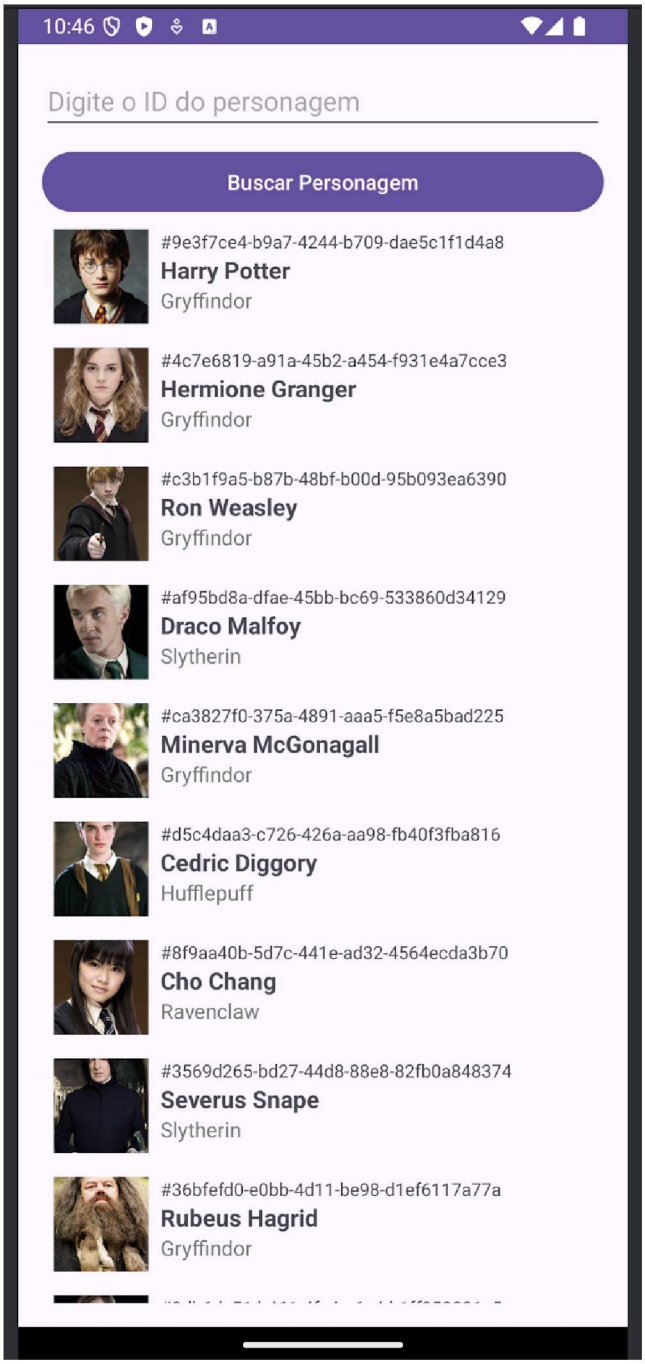
Fonte: O autor (2025)

Figura 30 – Tela mobile para listar professores



Fonte: O autor (2025)

Figura 31 – Tela mobile para listar personagem e buscar por ID do personagem



Fonte: O autor (2025)

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

Para uma boa construção de um sistema, é necessário ter sua estrutura correta, mas mais importante é sua infraestrutura, que são componentes físicos ou em *cloud* que vão sustentar o sistema à vida. Então a disciplina de Infraestrutura para Desenvolvimento e Implementação de Software (DevOps) culmina na prática e entendimento para ferramentas que permitam o *deploy* contínuo, prática vista que é essencial para metodologia ágil. Mas também a disciplina se entende para ensinamentos de versionamento, práticas comuns, diferentes maneiras que poderia ser realizado o *deploy* da aplicação.

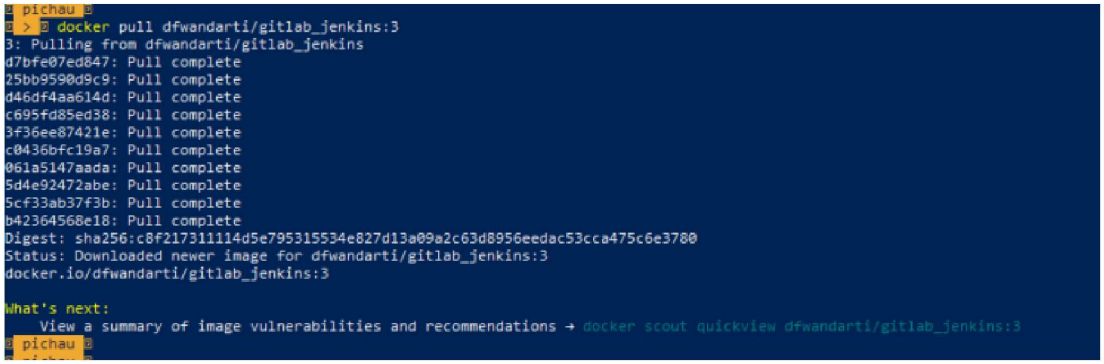
Depois que o aluno possui essa base de montagem para uma esteira CI/CD em produção sendo esta a orquestração de *build*, teste, encapsulamento e *deploy* dos recursos em nuvem, é apresentado o trabalho da disciplina, sendo este exemplo prático de uma imagem *docker* padronizada, fazer um *commit* e subir para um repositório hospedado pelo professor, o que simularia um ambiente final de produção. E assim comunicando na entrega contínua de valor e integração contínua, prática vista em MADS, mas que também é assegurada pela qualidade em MAG – trunk-based que é o método para favorecer o WIP e aumentar o *flow*, ecoando os princípios Kanban e modelagem incremental de acordo Humble (2011).

Visto que o projeto tem integração com outras disciplinas, e um alto grau de valor acadêmico e profissional ao provisionar, containerização, verificar métricas de observabilidade, CI/CD e IaC em um único projeto, a disciplina entrega a base fundamental para as demais equipes. Incorporando a cultura de melhoria contínua essencial para ambientes de engenharia de software modernos.

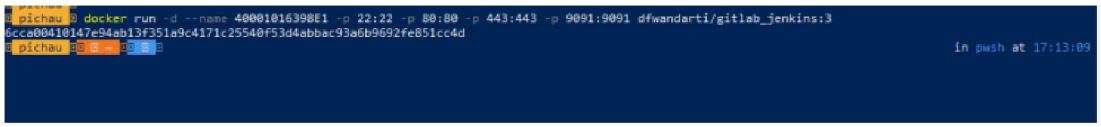
11.1 ARTEFATOS DO PROJETO

Figura 32 – Passos que foram executados através do terminal para poder realizar a execução do projeto

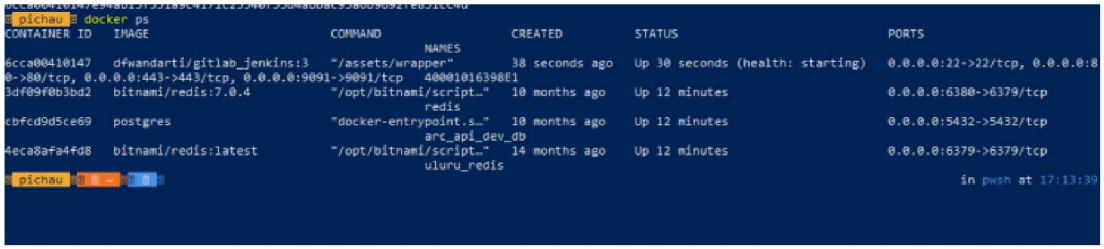
Passo 1:



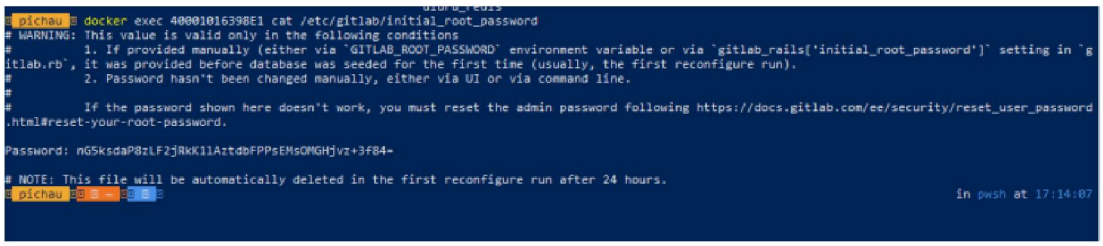
Passo 2:



Passo 3:



Passo 4:



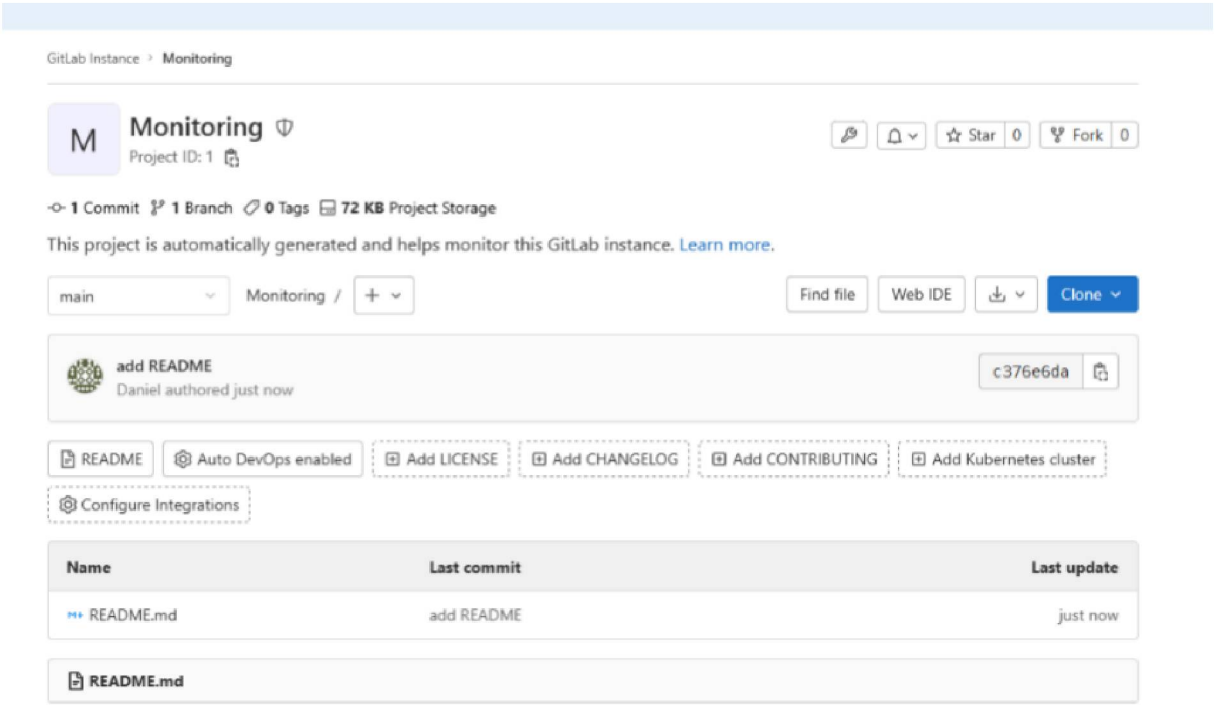
Fonte: O autor (2025)

Figura 33 – Comandos realizados dentro do container do Docker para realização de um commit

```
# ls
download-package  gitlab.rb  setup  sshd_config  update-permissions  wrapper
# git clone http://6cca00410147/gitlab-instance-24fc495c/Monitoring.git
Cloning into 'Monitoring'...
Username for 'http://6cca00410147': root
Password for 'http://root@6cca00410147':
remote: HTTP Basic: Access denied
fatal: Authen: not foundiled for 'http://6cca00410147/gitlab-instance-24fc495c/Monitoring.git/'
# ^[[A^[[A
# git clone http://6cca00410147/gitlab-instance-24fc495c/Monitoring.git
Cloning into 'Monitoring'...
Username for 'http://6cca00410147': root
Password for 'http://root@6cca00410147':
warning: You appear to have cloned an empty repository.
# ls
Monitoring  download-package  gitlab.rb  setup  sshd_config  update-permissions  wrapper
# cd Monitoring
# git switch -c main
Switched to a new branch 'main'
# touch README.md
# git add README.md
# git commit -m "add README"
[main (root-commit) c376e6d] add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
# git push -u origin main
Username for 'http://6cca00410147': root
Password for 'http://root@6cca00410147':
remote: HTTP Basic: Access denied
fatal: Authentication failed for 'http://6cca00410147/gitlab-instance-24fc495c/Monitoring.git/'
# git push -u origin main
Username for 'http://6cca00410147': root
Password for 'http://root@6cca00410147':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 209 bytes | 209.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To http://6cca00410147/gitlab-instance-24fc495c/Monitoring.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
#
```

Fonte: O autor (2025)

Figura 34 – Tela do Gitlab com o commit realizado dentro da imagem do docker, mostrando o sucesso da execução do projeto



Fonte: O autor (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados visa assegurar a qualidade do código, minimizando retrabalhos e agilizando os ciclos de *feedback*, entre outros benefícios. Esta disciplina aborda os métodos de realização de testes, bem como as melhores práticas para sua implementação. A estrutura curricular divide-se em três blocos: Testes Unitários com JUnit, uma ferramenta para escrita de testes em Java; Mock/Stubs e boas práticas de código testável; e Testes End-to-End com Playwright. O objetivo primordial é a adoção prática de uma pirâmide de testes robusta (Conh, 2009) e a aplicação do princípio *shift-left*, que preconiza a construção da qualidade desde o primeiro *commit*.

No contexto do projeto final da disciplina, o aluno, ao assimilar os conceitos de JUnit e TDD, confronta-se com a necessidade de desenvolver um código testável. Essa abordagem visa reduzir dependências, favorecer a injeção por construtor e aplicar a Lei de Deméter, em conformidade com os preceitos de Fowler (2018). Tal prática permite uma reformatação mais segura do código, prevenindo a quebra de outros componentes. Adicionalmente, o projeto incorpora a utilização de Mocks, Stubs, Dummies e Spies para isolar integrações externas, acelerando os testes e mitigando falsos positivos (Freeman & Pryce, 2010).

A disciplina de Testes encontra-se integrada às demais disciplinas do curso, dada sua relevância na fase inicial da codificação de projetos, destacando-se também sua importância em Web1/Web2 e introdução à programação. Essa interconexão visa elevar a qualidade do código e do produto final. Conceitos como “Done significa testado” são enfatizados, indicando que uma tarefa só pode ser considerada entregue e validada pelo Product Owner (P.O.) após passar por todo o pipeline de codificação e ser testada pelos programadores. Somente então ela pode ser submetida a outras validações dentro do processo de metodologias ágeis, alinhando-se aos conhecimentos adquiridos em MAP1 e proporcionando *feedbacks* mais ápidos. O aluno desenvolve a competência de escrever código testável, selecionar níveis de verificação adequados e integrar os testes em pipelines, habilidade esta abordada em Infraestrutura e DevOps.

12.1 ARTEFATOS DO PROJETO

Figura 35 – Código para execução do teste utilizando Playwright

AnotepadTest.java — Abrir com IntelliJ IDEA

```
package br.ufpr.test;
import com.microsoft.playwright.*;
import com.microsoft.playwright.options.WaitForSelectorState;
import com.microsoft.playwright.options.WaitUntilState;
import org.junit.jupiter.api.*;

public class AnotepadTest {
    static Playwright playwright;
    static Browser browser;
    BrowserContext context;
    Page page;

    @BeforeAll
    static void setupAll() {
        playwright = Playwright.create();
        browser = playwright.chromium().launch(new BrowserType.LaunchOptions().setHeadless(false));
    }

    @AfterAll
    static void tearDownAll() {
        browser.close();
        playwright.close();
    }

    @BeforeEach
    void setup() {
        context = browser.newContext();
        page = context.newPage();
    }

    @AfterEach
    void tearDown() {
        context.close();
    }

    @Test
    void testCriacaoNotaAnotepad() {
        // Navega até o site com timeout aumentado e aguardando o DOMContentLoaded
        page.navigate("https://pt.anotepad.com", new Page.NavigateOptions()
                .setTimeout(60000) // aumenta para 60 segundos
                .setWaitUntil(WaitUntilState.DOMCONTENTLOADED)); // espera até que o DOM esteja carregado

        // Preenche o campo de título
        page.fill("#edit_title", "Entrega trabalho TEST DAS 2024");

        // Preenche o campo de conteúdo com nome e matrícula
        page.fill("#edit_textarea", "Nome: João Silva\nMatrícula: 123456");

        // Clica no botão "Salvar"
        page.click("#btnSaveNote");

        // Aguarda que o elemento de mensagem de sucesso seja visível
        Locator msgSuccess = page.locator("p.alert.alert-warning").first();
        msgSuccess.waitFor(new Locator.WaitForOptions().setState(WaitForSelectorState.VISIBLE));

        // Recupera o texto da mensagem e verifica se contém a substring esperada
        String successText = msgSuccess.innerText();
        Assertions.assertTrue(successText.contains("Você salvou sua nota como"),
                "A nota não foi salva corretamente. Texto exibido: " + successText);
    }
}
```

Fonte: O autor (2025)

13 CONCLUSÃO

Este Memorial de Projetos detalha o aprendizado e aplicação de conceitos de Desenvolvimento Ágil de Software, integrando conhecimentos de diversas disciplinas para criar artefatos práticos.

Iniciando com Engenharia de Software (MADS), o documento aborda modelagem ágil (MAG1, MAG2), gerenciamento de projetos (GAP1, GAP2) e implementação técnica (INTRO, BD, AAP, WEB1, WEB2). As disciplinas também cobriram UX, desenvolvimento mobile (MOB1, MOB2) e infraestrutura/testes (INFRA, TEST). A integração desses conhecimentos demonstra a interconexão das áreas de desenvolvimento ágil, porém, a adoção prática enfrenta desafios como: Cultura Organizacional (Resistência à mudança e dificuldade em aceitar experimentação) Escala e Complexidade (Coordenação e sincronização em grandes projetos), Gerenciamento de Expectativas (Alinhamento de *stakeholders* com a natureza iterativa), Técnica (Equilíbrio entre qualidade e velocidade de entrega), DevOps e Automação (Carência de ferramentas e cultura adequada) e Talentos Qualificados (Escassez de profissionais multifuncionais).

O memorial busca demonstrar a aplicação prática de metodologias ágeis, destacando a necessidade contínua de capacitação, mudança cultural e investimento em infraestrutura para aproveitar os benefícios do desenvolvimento ágil de software.

14 REFERÊNCIAS

ALUR, M.; CRUPI, J.; MALKS, E. Core J2EE Patterns: Best Practices and Design Strategies. 2. ed. Upper Saddle River: Prentice Hall, 2003. Disponível em: https://www.informit.com/store/core-j2ee-patterns-best-practices-and-design-strategies-9780131422460. Acesso em: 7 ago. 2025.

AMBLER, S. W. Introduction to Data Normalization. AgileData.org, 2023. Disponível em: https://agiledata.org/essays/dataNormalization.html. Acesso em: 7 ago. 2025.

ANDERSON, D. J. Kanban: Successful Evolutionary Change for Your Technology Business. Sequim: Blue Hole Press, 2010. Disponível em: https://kanban.university/books/kanban-successful-evolutionary-change-for-your-technology-business/. Acesso em: 7 ago. 2025.

ANDROID DEVELOPERS. Guide to App Architecture. 2024. Disponível em: https://developer.android.com/topic/architecture?hl=pt-br. Acesso em: 7 ago. 2025.

BECK, K. Test-Driven Development: By Example. Boston: Addison-Wesley, 2003. ISBN 0321146530. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/test-driven-development-by-example/P200000002858/9780321146533. Acesso em: 7 ago. 2025.

BECK, K.; et al. Manifesto for Agile Software Development. 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 7 ago. 2025.

BURNS, B.; BEDA, J.; HIGHTOWER, K. Kubernetes Up & Running. 3. ed. Sebastopol: O'Reilly, 2022. Disponível em: https://www.oreilly.com/library/view/kubernetes-up-and/9781098110208/. Acesso em: 7 ago. 2025.

CAMARA, L. O que é User Experience: Entenda a importância para sua empresa. Camara UX®, 2025. Disponível em: https://camaraux.com.br/o-que-e-user-experience/. Acesso em: 30 jul. 2025.

CHACON, S.; STRAUB, B. Pro Git. 3. ed. Nova York: Apress, 2020. Disponível em: https://git-scm.com/book/en/v2. Acesso em: 7 ago. 2025.

CHEN, P. P. The Entity-Relationship Model — Toward a Unified View of Data. ACM Transactions on Database Systems (TODS), v. 1, n. 1, p. 9-36, 1976. Disponível em: https://doi.org/10.1145/320434.320440. Acesso em: 7 ago. 2025.

COHN, M. Succeeding with Agile. Boston: Addison-Wesley, 2009. cap. 13. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/succeeding-with-agile-software-development-using-scrum/P200000002864/9780321579362. Acesso em: 7 ago. 2025.

COHN, M. User Stories Applied: For Agile Software Development. Boston: Addison-Wesley, 2004. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/user-stories-applied-for-agile-software-development/P200000002859/9780321205681. Acesso em: 7 ago. 2025.

CURCIO, K. P. C. An approach for user experience design integration into agile software development. 2021. Tese (Doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2021. Disponível em: https://repositorio.pucpr.br/jspui/handle/123456789/31100. Acesso em: 7 ago. 2025.

DAKA, E.; FRASER, G. A Survey on Unit Testing Practices and Problems. Proceedings of the 25th IEEE International Symposium on Software Reliability Engineering (ISSRE 2014), Naples, p. 201-211, nov. 2014. DOI: 10.1109/ISSRE.2014.11. Disponível em: https://doi.org/10.1109/ISSRE.2014.11. Acesso em: 7 ago. 2025.

DEITEL, H. M.; DEITEL, P. J. Java How to Program. 8. ed. Upper Saddle River: Prentice Hall, 2010. Disponível em: https://deitel.com/java-how-to-program-8-e/. Acesso em: 7 ago. 2025.

DINARDO, S. Agile Metrics in Action. Shelter Island: Manning, 2020. Disponível em: https://www.manning.com/books/agile-metrics-in-action. Acesso em: 7 ago. 2025.

EVANS, E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston: Addison-Wesley, 2003. Disponível em: https://www.informit.com/store/domain-driven-design-tackling-complexity-in-the-heart-9780321125217. Acesso em: 7 ago. 2025.

FOWLER, M. Refactoring: Improving the Design of Existing Code. 2. ed. Boston: Addison-Wesley, 2018. Disponível em: https://martinfowler.com/books/refactoring.html. Acesso em: 7 ago. 2025.

FREEMAN, S.; PRYCE, N. Growing Object-Oriented Software, Guided by Tests. Boston: Addison-Wesley, 2010. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/growing-object-oriented-software-guided-by-tests/P200000002860/9780321503626. Acesso em: 7 ago. 2025.

GLAUBER, N. Dominando o Android com Kotlin. São Paulo: Novatec, 2022. Disponível em: https://novatec.com.br/livros/dominando-android-com-kotlin/. Acesso em: 7 ago. 2025.

HUMBLE, J.; FARLEY, D. Continuous Delivery. Boston: Addison-Wesley, 2011. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/continuous-delivery-reliable-software-releases-through-build-test-and-deployment-automation/P200000002862/9780321601919. Acesso em: 7 ago. 2025.

IEEE. Systems and software engineering — Vocabulary. ISO/IEC/IEEE 24765:2017(E). Geneva: International Organization for Standardization, 2017. Disponível em: https://www.iso.org/standard/71952.html. Acesso em: 7 ago. 2025.

JEMEROV, D.; ISAKOV, S. Kotlin in Action. Shelter Island: Manning, 2017. Disponível em: https://www.manning.com/books/kotlin-in-action. Acesso em: 7 ago. 2025
KIM, G.; BEHR, K.; SPAFFORD, G. The Phoenix Project. Rio de Janeiro: Alta Books, 2013. Disponível em: https://www.thephoenixprojectbook.com/. Acesso em: 7 ago. 2025.

KIM, G.; BEHR, K.; SPAFFORD, G.; HUMBLE, J. The DevOps Handbook. 2. ed. Portland: IT Revolution, 2020. Disponível em: https://itrevolution.com/product/the-devops-handbook/. Acesso em: 7 ago. 2025.

KNIBERG, H. Lean from the Trenches. Dallas: Pragmatic Bookshelf, 2010. Disponível em: https://pragprog.com/titles/hklean/lean-from-the-trenches/. Acesso em: 7 ago. 2025.

KNIBERG, H.; SKARIN, M. Kanban and Scrum – Making the Most of Both. C4Media, 2010. Disponível em: https://www.infoq.com/minibooks/kanban-scrum-minibook/. Acesso em: 7 ago. 2025.

LECHETA, R. Android Essencial com Kotlin. São Paulo: Casa do Código, 2020. Disponível em: https://www.casadocodigo.com.br/products/livro-android-essencial-kotlin. Acesso em: 7 ago. 2025.

MAJORS, C.; FONT-JONES, L.; MIRANDA, G. Observability Engineering. Sebastopol: O'Reilly, 2022. Disponível em: https://www.oreilly.com/library/view/observability-engineering/9781492076441/. Acesso em: 7 ago. 2025.

MARTIN, R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Boston: Prentice Hall, 2008. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/clean-code-a-handbook-of-agile-software-craftsmanship/P200000002870/9780132350884. Acesso em: 7 ago. 2025.

MERKEL, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal, n. 239, 2014. DOI: 10.5555/2600239.2600241. Disponível em: https://dl.acm.org/doi/10.5555/2600239.2600241. Acesso em: 7 ago. 2025.
MESZAROS, G. xUnit Test Patterns. Boston: Addison-Wesley, 2007. Disponível em: https://www.informit.com/store/xunit-test-patterns-refactoring-test-code-9780131495050. Acesso em: 7 ago. 2025.

MICROSOFT. Playwright Docs: Testing Web Apps with Playwright. 2024. Disponível em: https://playwright.dev/docs/intro. Acesso em: 7 ago. 2025.

MOSKALA, M. Effective Kotlin. Leanpub, 2021. Disponível em: https://leanpub.com/effectivekotlin. Acesso em: 7 ago. 2025.

RIES, E. The Lean Startup. Nova York: Crown Business, 2011. Disponível em: https://theleanstartup.com/. Acesso em: 7 ago. 2025.

SCHWABER, K.; SUTHERLAND, J. The Scrum Guide. Scrum.org, 2020. Disponível em: https://scrumguides.org/. Acesso em: 7 ago. 2025.

VITALINO, J. Descomplicando o Docker. São Paulo: Novatec, 2016. Disponível em: https://novatec.com.br/livros/descomplicando-docker/. Acesso em: 7 ago. 2025.

WANG, R. Infrastructure as Code: Patterns and Practices. Shelter Island: Manning, 2020. Disponível em: https://www.manning.com/books/infrastructure-as-code. Acesso em: 7 ago. 2025.

WOJCIECHOWSKI, J. Análise de Sistemas. 1. ed. Curitiba: IESDE Brasil, 2021. ISBN 978-65-5821-013-9. Disponível em: https://www.videolivraria.com.br/analise-de-sistemas. Acesso em: 7 ago. 2025.

ZANETTE, F. MVP (Minimum Viable Product, ou Produto Mínimo Viável): guia prático. RD Station, 2025. Disponível em: https://www.rdstation.com/blog/marketing/mvp-minimo-produto-viavel/. Acesso em: 30 jul. 2025.