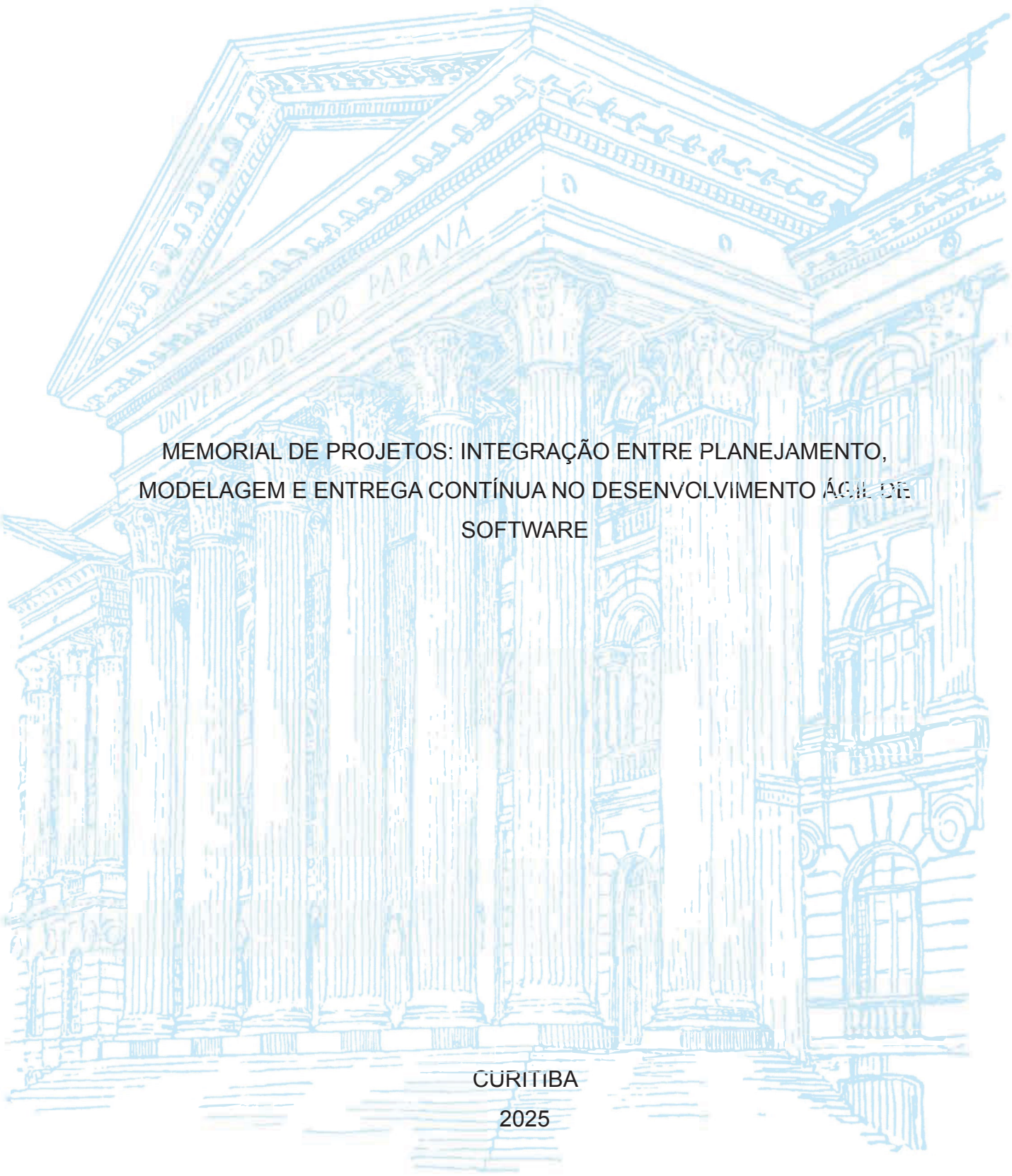
UNIVERSIDADE FEDERAL DO PARANÁ

ANDERSON ANTONIO RUBEL



MEMORIAL DE PROJETOS: INTEGRAÇÃO ENTRE PLANEJAMENTO, MODELAGEM E ENTREGA CONTÍNUA NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

CURITIBA

2025

ANDERSON ANTONIO RUBEL

MEMORIAL DE PROJETOS: INTEGRAÇÃO ENTRE PLANEJAMENTO, MODELAGEM E ENTREGA CONTÍNUA NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

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

Orientador: Prof. Dr. JAIME WOJCIECHOWSKI

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 **ANDERSON ANTONIO RUBEL**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO ENTRE PLANEJAMENTO, MODELAGEM E ENTREGA CONTÍNUA NO DESENVOLVIMENTO ÁGIL DE SOFTWARE**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 18 de Agosto de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial tem como objetivo apresentar a trajetória formativa vivenciada no curso de Especialização em Desenvolvimento Ágil de Software, evidenciando a integração entre teoria e prática no contexto do desenvolvimento de sistemas. Ao longo das disciplinas, foram desenvolvidos diversos artefatos que representaram etapas essenciais do ciclo de vida de software: documentos de requisitos, modelagens funcionais e estruturais, planos de release, quadros Kanban, protótipos de interface, códigos-fonte para aplicações web e mobile, scripts de testes automatizados e configurações de infraestrutura com Docker e GitLab.Esses materiais foram construídos a partir da aplicação de princípios ágeis, como iteração contínua, foco no cliente, entrega incremental, colaboração entre times e adaptação rápida a mudanças. A disciplina de Métodos Ágeis forneceu a base conceitual que permeou todas as demais. As práticas de UX foram aplicadas à prototipação de soluções centradas no usuário, enquanto os conhecimentos de DevOps e Testes possibilitaram integração e validação contínuas. O planejamento de projetos e a modelagem de sistemas garantiram organização e coerência técnica aos desenvolvimentos realizados.A integração entre os conteúdos reforçou uma visão completa e aplicada do desenvolvimento ágil, aproximando o ambiente acadêmico da realidade prática de projetos modernos de software.

Palavras-chave: Desenvolvimento Ágil, Engenharia de Software, Integração Contínua, Modelagem de Sistemas.

ABSTRACT

This memorial aims to present the educational journey experienced throughout the Postgraduate Course in Agile Software Development, highlighting the integration between theory and practice in the context of software engineering. Throughout the modules, several artifacts were developed to represent key stages of the software development lifecycle: requirement documents, functional and structural models, release planning, Kanban boards, interface prototypes, source code for web and mobile applications, automated test scripts, and infrastructure configurations using Docker and GitLab.These materials were created based on agile principles such as continuous iteration, customer focus, incremental delivery, team collaboration, and rapid adaptation to change. The Agile Methods module provided the conceptual foundation that supported all others. UX practices were applied to user-centered prototyping, while DevOps and Testing disciplines enabled integration and continuous validation. Project planning and system modeling ensured organization and technical consistency in the solutions developed.The integration of all content reinforced a complete and applied understanding of agile development, bringing academic learning closer to the practical reality of modern software projects.

Keywords: Agile Development, Software Engineering, Continuous Integration, System Modeling.

SUMÁRIO

1 PARECER TÉCNICO

O presente parecer técnico tem como objetivo apresentar uma análise integrada dos projetos realizados ao longo das disciplinas da Especialização em Desenvolvimento Ágil de Software, destacando como cada uma contribuiu para a aplicação prática dos princípios ágeis. As atividades desenvolvidas evidenciam uma abordagem iterativa, incremental e colaborativa, base fundamental dos métodos ágeis de desenvolvimento (Beck *et al.*, 2001).

A disciplina Métodos Ágeis de Desenvolvimento de Software (MADS) forneceu os fundamentos teóricos dos métodos Scrum, Kanban e Extreme Programming (XP), consolidando práticas como planejamento incremental, papéis e cerimônias ágeis, entrega contínua e adaptação baseada em feedback. Esses conceitos se tornaram a espinha dorsal metodológica dos demais projetos, alinhando-se às recomendações de Cohn (2010) sobre priorização baseada em valor de negócio e entregas rápidas.

Nas disciplinas Gerenciamento Ágil de Projetos I e II (GAP1 e GAP2), foram elaborados artefatos como o plano de release, histórias de usuário e simulações com o framework Kanban. Essas ferramentas contribuíram para organizar o escopo, priorizar entregas e melhorar o fluxo de trabalho, preparando o terreno para aplicações práticas desenvolvidas em outras unidades curriculares. Segundo Anderson (2010), o Kanban é essencial para promover um fluxo contínuo e reduzir gargalos no desenvolvimento, objetivo refletido nas práticas adotadas nos trabalhos.

A integração dos conceitos foi reforçada nas disciplinas de Modelagem Funcional e Estrutural, com a construção de diagramas de casos de uso, diagramas de classes, sequência e atividades que representaram cenários reais de interação entre usuário e sistema. Esses modelos facilitaram a comunicação entre áreas técnicas e de negócio, permitindo refinar e validar as soluções antes da implementação, conforme defendido por Sommerville (2011) no uso de modelagem como ferramenta de alinhamento de requisitos.

A disciplina UX no Desenvolvimento Ágil de Software complementou esse processo ao colocar o usuário no centro das decisões, por meio da criação de protótipos funcionais, testes de usabilidade e coleta de feedback. Essa abordagem reforçou práticas recomendadas por Nielsen e Budiu (2012), sendo fundamental

para as decisões de design adotadas nos projetos Web (WEB1 e WEB2) e Mobile (MOB1 e MOB2), resultando em aplicações navegáveis com interfaces intuitivas, codificadas com base em boas práticas e orientações centradas no usuário.

Do ponto de vista da sustentação técnica, a disciplina Infraestrutura e DevOps (INFRA) permitiu aplicar conceitos de integração e entrega contínua, por meio da configuração de ambientes com Docker, GitLab e Jenkins. Essa prática segue a visão de Kim *et al.* (2016) sobre automação de pipelines como fator crítico para manter a qualidade e a velocidade em ambientes ágeis.

A qualidade do código foi aprimorada na disciplina Aspectos Ágeis de Programação (AAP), por meio da aplicação de técnicas de clean code e refatorações, práticas reforçadas em Testes Automatizados (TEST), onde scripts foram elaborados para validar funcionalidades em ambiente controlado, garantindo segurança em entregas frequentes.

As bases da programação foram consolidadas na disciplina Introdução à Programação (INTRO), com o desenvolvimento de sistemas simples com acesso a banco de dados, enquanto Banco de Dados (BD) forneceu a estrutura necessária para persistência de informações, por meio de modelagem relacional e uso de SQL.

Todos esses projetos estão organizados neste memorial, por disciplina, evidenciando como os entregáveis se conectam em um fluxo de desenvolvimento ágil completo: da concepção à entrega, passando por planejamento, modelagem, codificação, testes, infraestrutura e validação com usuários. Como apontam Pressman e Maxim (2016), o desenvolvimento ágil exige disciplina, foco no cliente e capacidade de adaptação — características visivelmente incorporadas nas atividades desenvolvidas ao longo do curso. A constante interação entre teoria e prática aproximou o ambiente acadêmico da realidade de projetos modernos de software.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

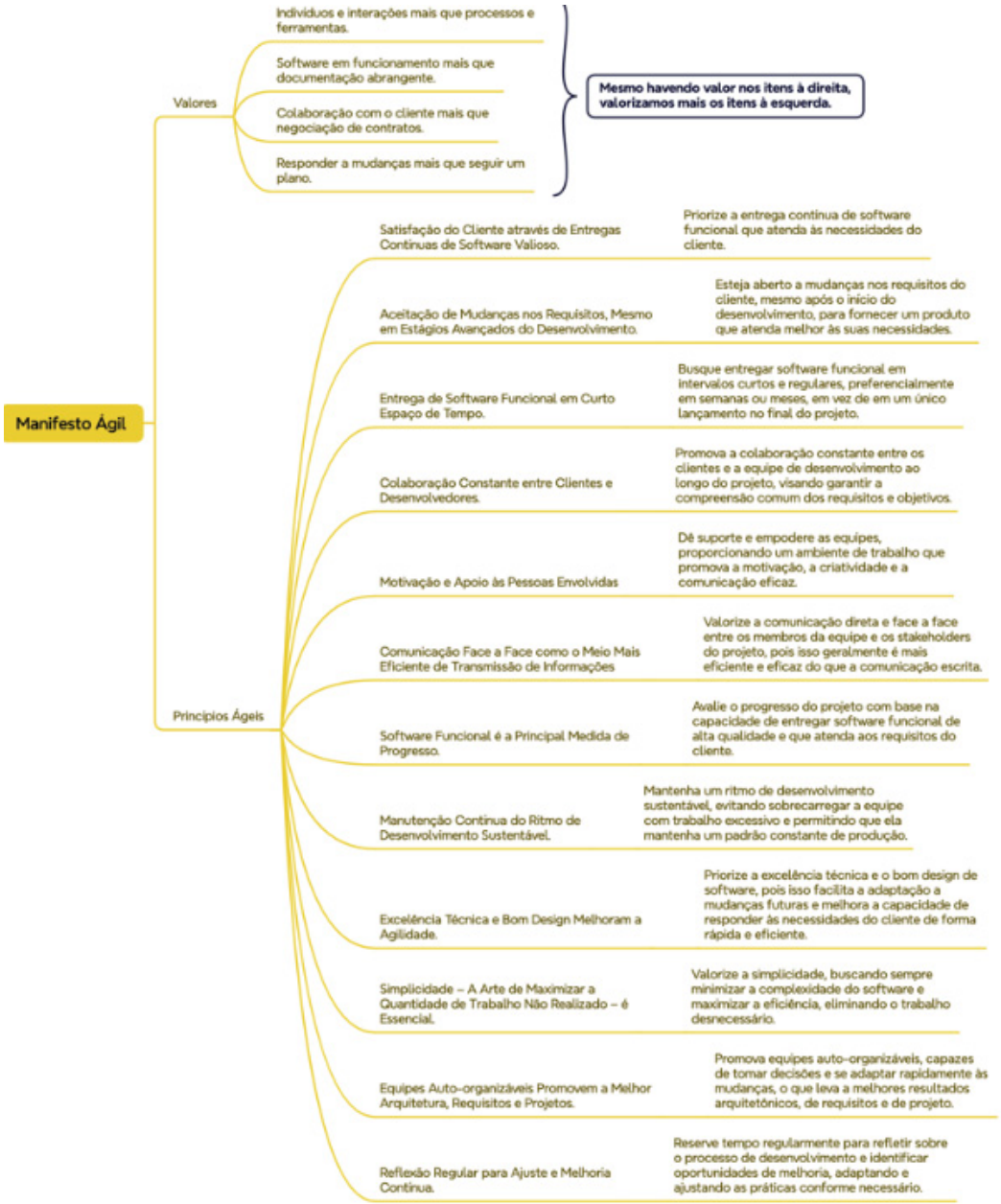
A disciplina de Métodos Ágeis para Desenvolvimento de Software trouxe os fundamentos necessários para construir os conhecimentos que sustentam as metodologias gerais na área da engenharia de software. Explorou-se os valores e princípios do Manifesto Ágil, assim como os métodos Scrum, Extreme Programming (XP), Kanban e abordagens de entrega contínua. O estudo dos papéis, artefatos e cerimônias no Scrum evidenciou a relevância da iteração, da colaboração entre equipes e da entrega incremental, consolidando o conceito de adaptação contínua em ciclos curtos.

A abordagem de entrega contínua mostrou-se parte fundamental do desenvolvimento ágil trazendo consigo um modelo de entrega frequente e confiável de incrementos de software. Todos os princípios e práticas vistos foram frequentemente abordados em outras disciplinas, como modelagem funcional e estrutural, experiência do usuário (UX), gestão de projetos, desenvolvimento web e mobile, DevOps e testes automatizados, contribuindo para a construção de uma solução integrada, iterativa e alinhada aos preceitos do desenvolvimento ágil.

Desenvolveu-se um mapa mental no qual é sintetizado todo conteúdo abordado ao decorrer da disciplina no qual pode-se ver de forma estruturada todo conhecimento obtido, selecionou-se o trecho que aborda o Manifesto Ágil que pode ser visto na Figura 1 a seguir.

2.1 ARTEFATOS DO PROJETO

FIGURA 1 - MAPA MENTAL.



FONTE: O autor (2025).

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

Ao decorrer da disciplina MAG1 foi abordado os principais aspectos necessário antes de realizar o desenvolvimento de um software que são o levantamento de requisitos, criação de casos de uso de nível 1 e 2, histórias de Usuário, prototipação, doomain-driven design e outro aspectos ágeis de modelagem.

As abordagens citadas acima foram implementadas em um projeto de sistema para gestão de condomínio que permitiu a estruturação de forma lógica das funcionalidades esperadas da aplicação. Durante a modelagem funcional elaborou-se diagramas de caso de uso nível 1 e 2 Figura 2 e 3 os quais representa m os principais atores e suas iterações com o sistema, com destaque para funcionalidades como registro de unidades habitacionais , cadastro de moradores, geração de boletos e controle de inadimplência. Realizou-se a descrição de histórias de usuário em conjunto com a criação de desenhos de telas com seus critérios de aceitação e regra de negócio Figura 4 , que serviram para criação do backlog do produto.

Na sequência, iniciou-se os estudos da disciplina MAG2 voltada para a modelagem estrutural que aprofundou a representação técnica do sistema através de diagramas UML, introduziu-se os fundamentos básicos Objetos, Classes, Relacionamentos, Encapsulamento, Herança, Polimorfismo.

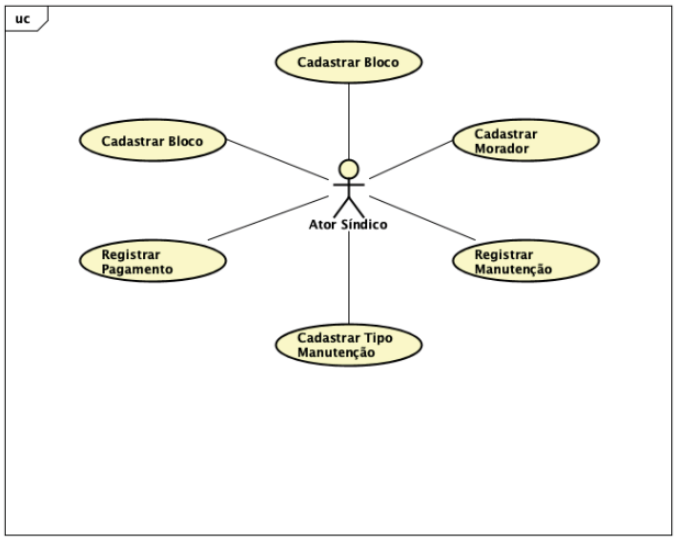
Dando continuidade ao projeto para gestão de um condomínio aplicando os conhecimentos obtidos desenvolveu-se diagramas de classes Figura 5, descrevendo entidades como "Unidade", "Morador", "Síndico" e "Boleto", com suas respectivas propriedades e relacionamentos. Além disso, foram utilizados diagramas de sequência Figura 6 para ilustrar fluxos de interação em funcionalidades específicas, como o processo de emissão de boletos e atualização de status de pagamento. Essas representações estruturadas facilitaram a transição para as etapas de implementação, orientando tanto a modelagem do banco de dados quanto a lógica de programação.

A integração entre os modelos funcionais e estruturais foi essencial para garantir coerência entre o que foi planejado e o que foi implementado. Os artefatos produzidos nessas disciplinas contribuíram para a construção de um sistema bem

definido, modular e alinhado às práticas ágeis, favorecendo entregas incrementais, validação contínua e adaptação ao longo do desenvolvimento do projeto.

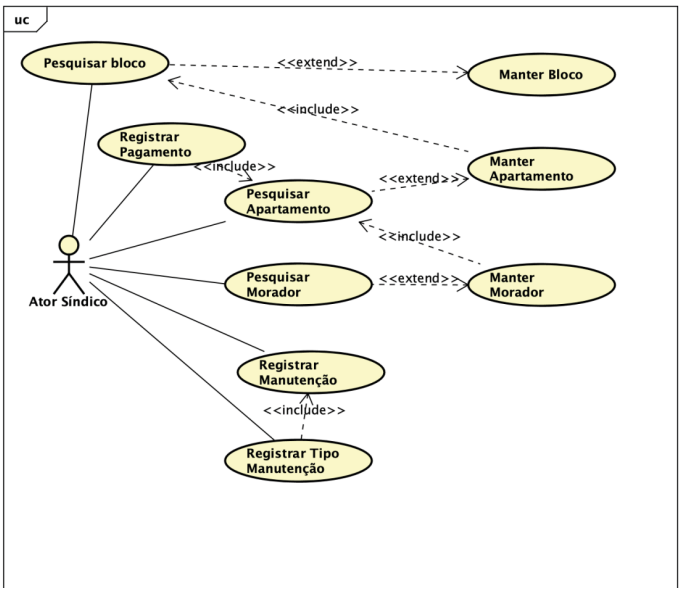
3.1 ARTEFATOS DO PROJETO

FIGURA 2 - DIAGRAMA DE CASO DE USO NÍVEL 1.



FONTE: O autor (2025).

FIGURA 3 - DIAGRAMA DE CASO DE USO NÍVEL 2.



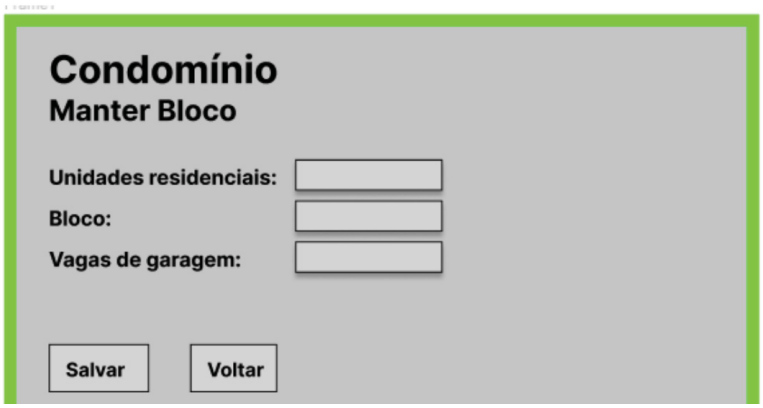
FONTE: O autor (2025).

FIGURA 4 - HISTÓRIA DE USUÁRIO,DESENHO DE TELA, CRITÉRIOS E REGRAS.

HU002 - Manter Bloco

SENDO	Síndico
QUERO	Manter os dados dos blocos
PARA	que os dados dos blocos 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 salvar os dados do bloco
3) Deve permitir apenas números inteiros nos campos vagas de garagem e unidades residenciais
4) Deve voltar à tela anterior

CRITÉRIOS DE ACEITAÇÃO - DETALHAMENTO:

1) Deve receber o parâmetro da tela de pesquisa e configurar a tela

Dado que	Os blocos estão na tabela
Quando	Quando A tela é apresentada
Então	Então O sistema configura a tela conforme os parâmetros recebidos (R1)

REGRAS DE NEGÓCIO:

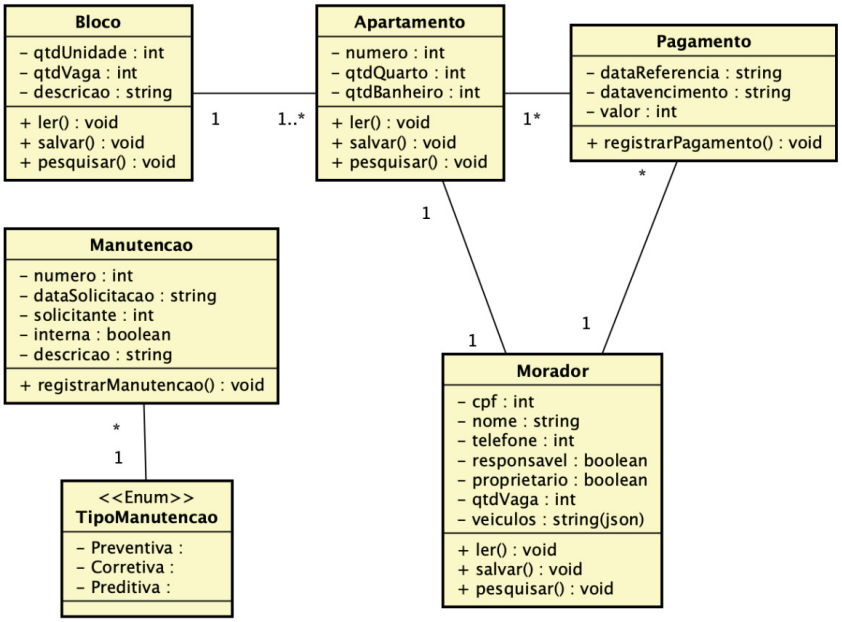
R1 – Configurar a tela conforme os seguintes parâmetros e regras

Parâmetro	**Regra**
Novo	Apresentar a tela com todos os campos vazios e habilitados. Deixar os botões Salvar e Voltar
Alterar	Apresentar a tela com todos os campos habilitados e preenchidos com os dados recebidos. Deixar os botões Salvar e Voltar
Consultar	Apresentar a tela com todos os campos desabilitados e preenchidos com os dados recebidos. Deixar somente o botão Voltar

R2 – O campo de unidades residenciais deve ser um inteiro
R3 – O campo de vagas de garagem deve ser um inteiro

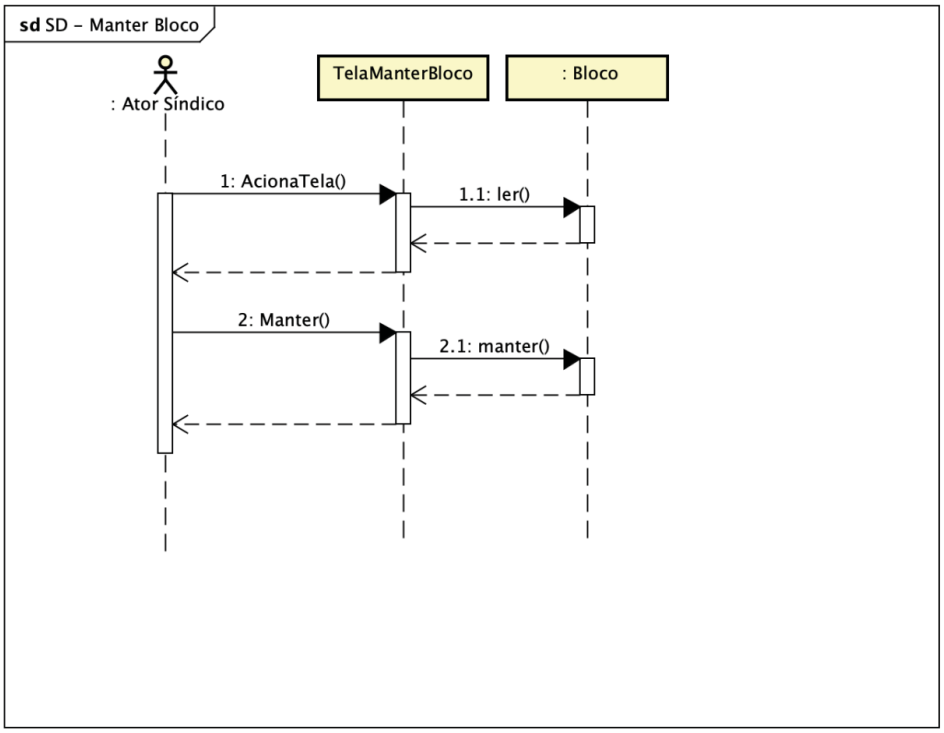
FONTE: O autor (2025).

FIGURA 5 - DIAGRAMA DE CLASSES.



FONTE: O autor (2025).

FIGURA 6 - DIAGRAMA DE SEQUÊNCIA.



FONTE: O autor (2025).

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

As disciplinas de Gerenciamento Ágil de Projetos de Software I e II abordaram os fundamentos e práticas relacionadas à condução de projetos no contexto de métodos ágeis, com ênfase na aplicação prática de frameworks como Scrum e Kanban. Durante GAP1, o foco esteve na construção de um plano de release Figura 7 para o sistema de gestão de condomínio. A atividade envolveu o levantamento, priorização e estimativa de histórias de usuário com base nas necessidades dos perfis de síndico, morador e administrador. As histórias incluíram funcionalidades como manutenção de blocos e apartamentos, registro de moradores, envio de comunicados, controle de ocorrências e visualização de estatísticas.

O plano foi estruturado em sprints quinzenais, com estimativas baseadas em horas disponíveis e velocidade de entrega, utilizando práticas como planning poker, backlog grooming e definição de critérios de aceite. Essa abordagem permitiu estabelecer um cronograma viável, respeitando as limitações de tempo e capacidade de desenvolvimento, e mantendo o foco na entrega contínua de valor ao usuário.

Em GAP2, os conceitos de fluxo de trabalho e melhoria contínua foram aprofundados por meio da simulação prática com o Game Kanban Figura 8, que evidenciou os impactos da limitação de trabalho em progresso (WIP), da visualização de gargalos e da importância do ciclo de feedback rápido. A dinâmica reforçou a importância da colaboração da equipe, da autogestão e da adaptação contínua como pilares da eficiência operacional.

As duas disciplinas forneceram base para o gerenciamento de escopo, tempo e entrega do projeto, conectando-se diretamente com os conteúdos de modelagem, desenvolvimento web e mobile, testes e DevOps. O planejamento das entregas orientou a organização do trabalho ao longo do curso, assegurando o alinhamento com os princípios ágeis de transparência, inspeção e adaptação. Como resultado, foi possível manter um fluxo de desenvolvimento sustentável e orientado à entrega incremental de funcionalidades priorizadas.

4.1 ARTEFATOS DO PROJETO

FIGURA 7 - PLANO DE REALEASE.

Cálculo da Velocidade:

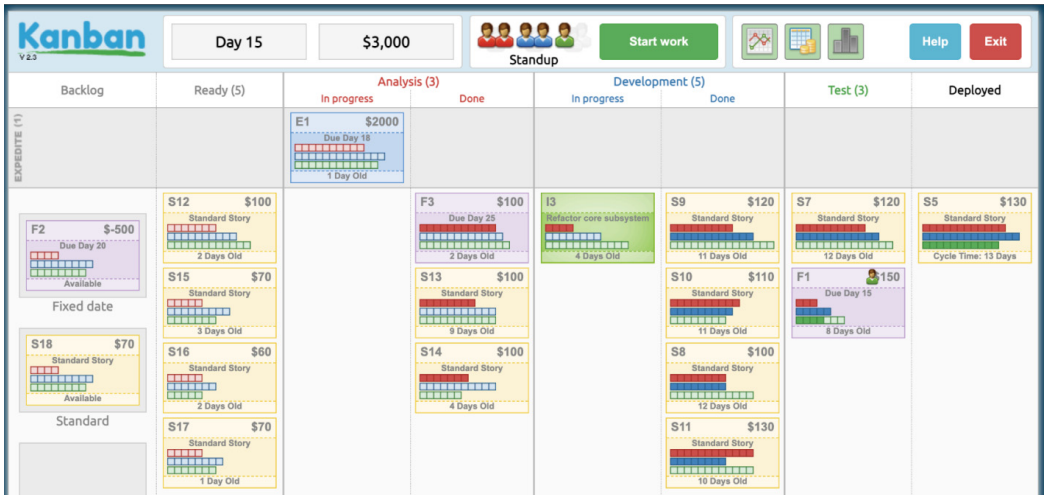
Horas disponíveis por dia:4H		Tamanho da Sprint: 2 semanas(10 dias úteis)	
Horas disponíveis por Sprint:40 (Horas disponíveis por dia * Tamanho da Sprint) = 4x10 = 40 H		Velocidade:5 (Horas disponíveis por Sprint / 8H) = 40/8 = 5	

Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4
Data Início:06/05/2024	Data Início:20/05/2024	Data Início:03/06/2024	Data Início:17/06/2024
Data Fim:17/05/2024	Data Fim:31/05/2024	Data Fim:14/06/2024	Data Fim:28/06/2024
<HU002 - Manter Apartamento> **SENDO** Síndico **QUERO** Manter os dados de um apartamento no sistema **PARA** Registrar informações detalhadas sobre o novo apartamento ESTIMATIVA (2)	<HU006 - Adicionar Morador> **SENDO** Síndico **QUERO** Adicionar um novo morador a um bloco específico **PARA** Registrar informações pessoais e de contato do morador ESTIMATIVA (2)	<HU009 - Visualizar Calendário de Reservas> **SENDO** Morador ou Síndico **QUERO** Visualizar um calendário de reservas das áreas comuns do condomínio **PARA** Verificar disponibilidade e planejar reservas futuras ESTIMATIVA (1)	<HU013 - Enviar Comunicado> **SENDO** Síndico ou Administrador **QUERO** Enviar um comunicado para todos os moradores do condomínio **PARA** Informar sobre eventos, reuniões ou outras informações relevantes ESTIMATIVA (1)

FONTE: O autor (2025).

FIGURA 8 - GAME KANBAN.



FONTE: O autor (2025).

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação forneceu os fundamentos da lógica computacional e das estruturas básicas de programação, necessária para a formação de qualquer profissional da área de desenvolvimento de software. Utilizando a linguagem Java e a IDE NetBeans como ambiente de desenvolvimento, a proposta pedagógica enfatizou a prática da escrita de algoritmos, o uso de estruturas condicionais e de repetição, além da manipulação de dados em banco relacional e da aplicação de testes unitários com JUnit .

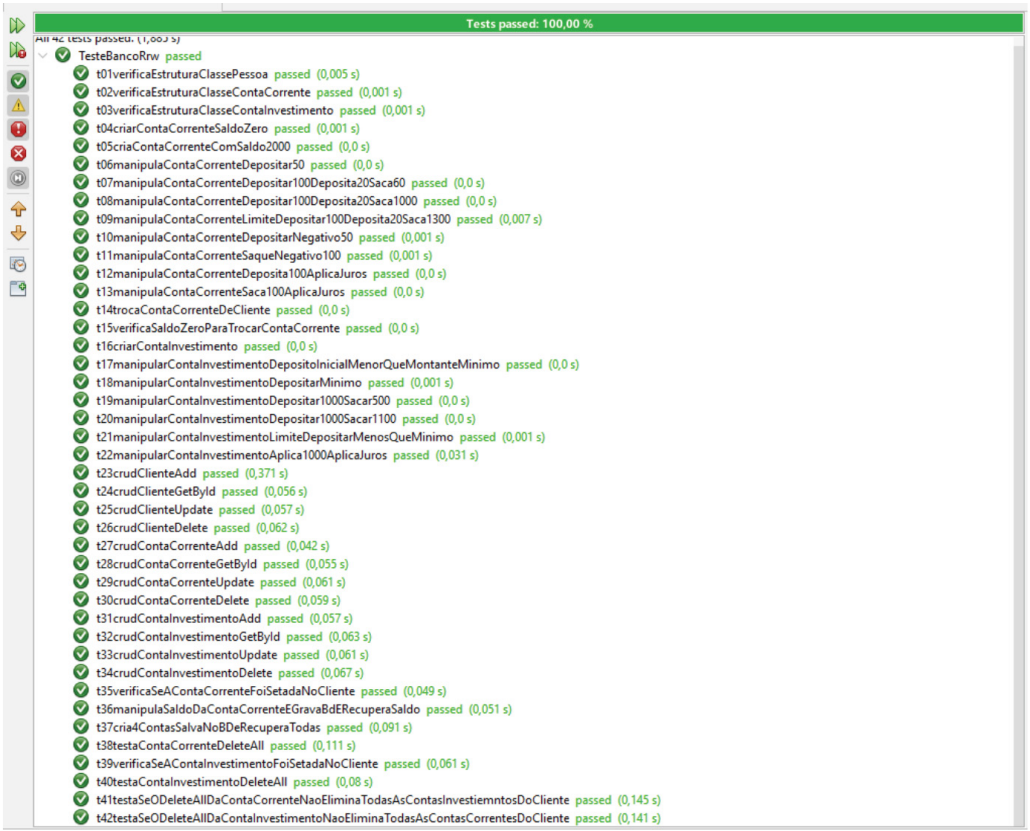
Foi proposto como atividade avaliativa o desenvolvimento do backend de um sistema bancário simplificado, com suporte às operações de cadastro de clientes, criação de contas correntes e contas investimento, movimentações de crédito e débito, com persistência em banco de dados MySQL. O projeto incluía um diagrama de classes e um conjunto de 42 casos de teste automatizados previamente definidos. A meta era fazer com que pelo menos 40 desses testes fossem aprovados simultaneamente Figura 9, utilizando o paradigma orientado a objetos e os princípios de desenvolvimento orientado por testes (TDD).

A realização deste projeto permitiu a aplicação dos conteúdos teóricos em um cenário prático, simulando o desenvolvimento real de um sistema com validações automatizadas. A prática de TDD contribuiu diretamente para a compreensão da importância dos testes desde as primeiras fases do ciclo de vida do software, conceito fundamental nas disciplinas posteriores de Aspectos Ágeis de Programação, Testes Automatizados e DevOps.

Além disso, o raciocínio lógico e a capacidade de decompor o problema em funções e classes reutilizáveis proporcionaram uma base sólida para o desenvolvimento de soluções mais complexas nas disciplinas de desenvolvimento Web e Mobile, bem como para a modelagem funcional e estrutural. A disciplina, portanto, cumpriu papel estratégico ao introduzir os fundamentos que sustentam a lógica de desenvolvimento iterativo, incremental e validado, conforme preconizado pelas metodologias ágeis.

5.1 ARTEFATOS DO PROJETO

FIGURA 9 - TESTES 100% APROVADOS.



FONTE: O autor (2025).

6 DISCIPLINA: BD – BANCO DE DADOS

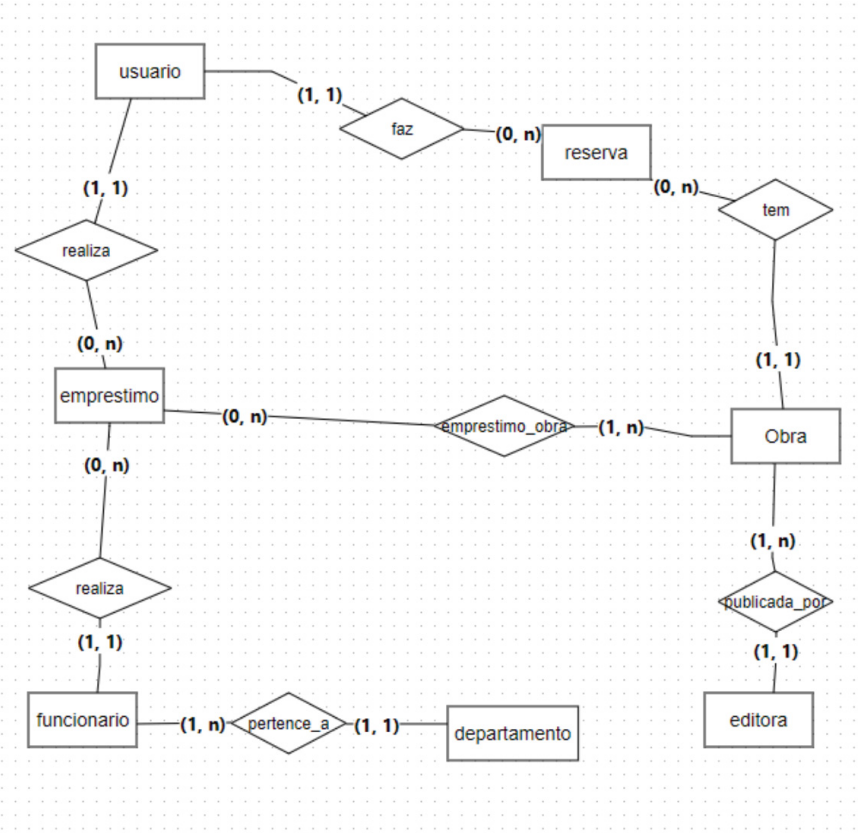
A disciplina de Banco de Dados teve como foco a compreensão e aplicação das etapas fundamentais da modelagem e implementação de sistemas de persistência de dados, essenciais para qualquer processo de desenvolvimento ágil de software. Os conteúdos abordados incluíram a modelagem conceitual e lógica, utilizando o modelo Entidade-Relacionamento, bem como a implementação de esquemas relacionais via SQL, com ênfase em boas práticas de normalização, integridade referencial e estruturação eficiente dos dados.

O trabalho desenvolvido na disciplina foi dividido em duas partes. A primeira parte consistiu na modelagem conceitual e lógica de um sistema de controle de biblioteca Figura 10 e 11, o que proporcionou uma base sólida para a compreensão das estruturas fundamentais de dados, relacionamentos e abstrações aplicadas a cenários reais. A segunda parte, de natureza prática, permitiu a escolha de um tema livre — no caso, o controle de estoque de joias —, onde foram exploradas relações 1:1, 1:N e N:N entre tabelas como Produto, Categoria, Fornecedor, Entrada e Imagem. Essa atividade envolveu a elaboração de um modelo lógico Figura 12, criação de tabelas, definições de chaves primárias e estrangeiras, além da implementação de inserções reais de dados, visualizações com *views* SQL e manipulação de dados com atualizações e joins.

A disciplina teve papel transversal no curso ao oferecer a base técnica para o armazenamento e organização de informações utilizadas nos demais projetos das disciplinas de programação, UX, DevOps e métodos ágeis. A clareza na definição do modelo de dados foi fundamental para a construção do sistema integrado entregue no projeto final do curso, assegurando que as regras de negócio fossem refletidas corretamente na camada de persistência. Assim, o domínio das técnicas de modelagem e uso de SQL contribuiu significativamente para a fluidez, consistência e escalabilidade das aplicações desenvolvidas sob a abordagem ágil.

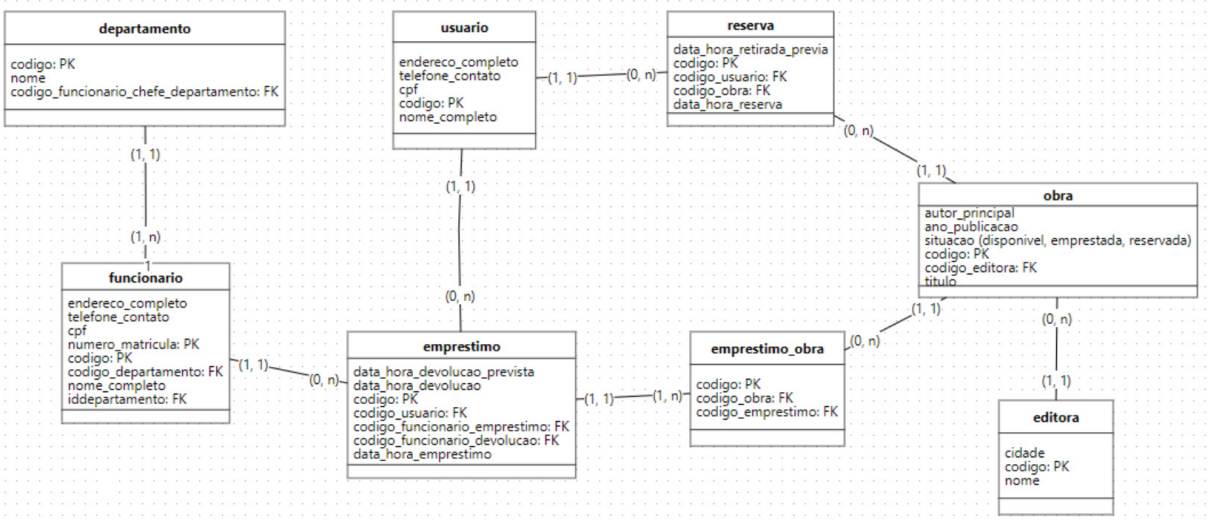
6.1 ARTEFATOS DO PROJETO

FIGURA 10 - MODELO ENTIDADE-RELACIONAMENTO CONCEITUAL.



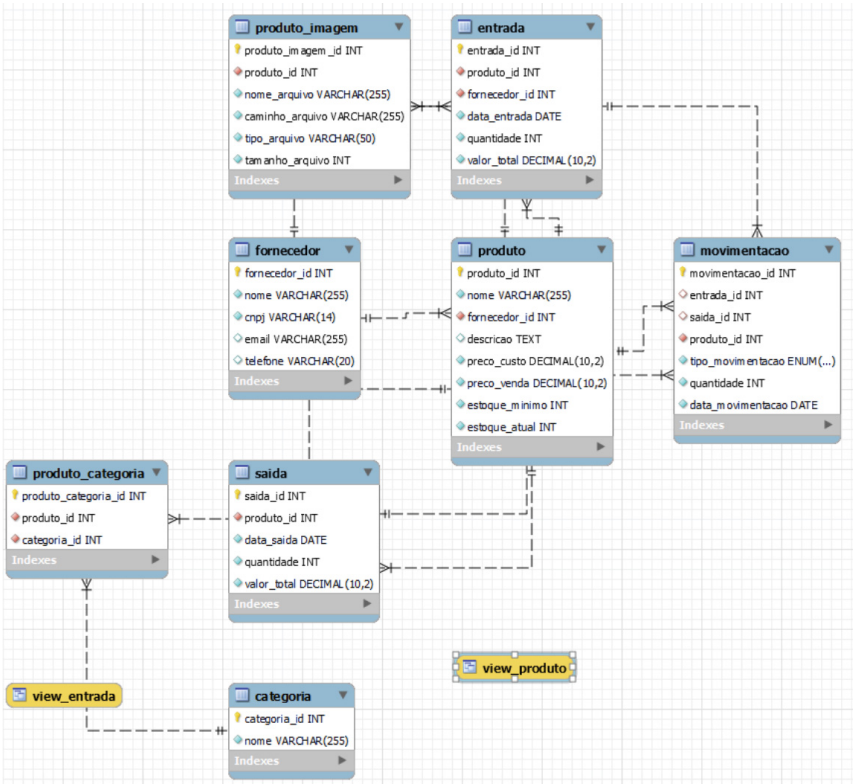
FONTE: O autor (2025).

FIGURA 11 - MODELO LÓGICO - DIAGRAMA DE ENTIDADE RELACIONAMENTO.



FONTE: O autor (2025).

FIGURA 12 - MODELO LÓGICO.



FONTE: O autor (2025).

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como foco central a aplicação de boas práticas de codificação associadas aos princípios do desenvolvimento ágil. Os conteúdos abordaram temas como refatoração, testes automatizados, versionamento com Git, princípios SOLID, programação orientada a objetos, TDD (Test-Driven Development), além de padrões de projeto e métricas de qualidade de software. Esses conceitos sustentam a manutenção de código limpo, reutilizável, testável e adaptável, características fundamentais em ciclos iterativos e incrementais.

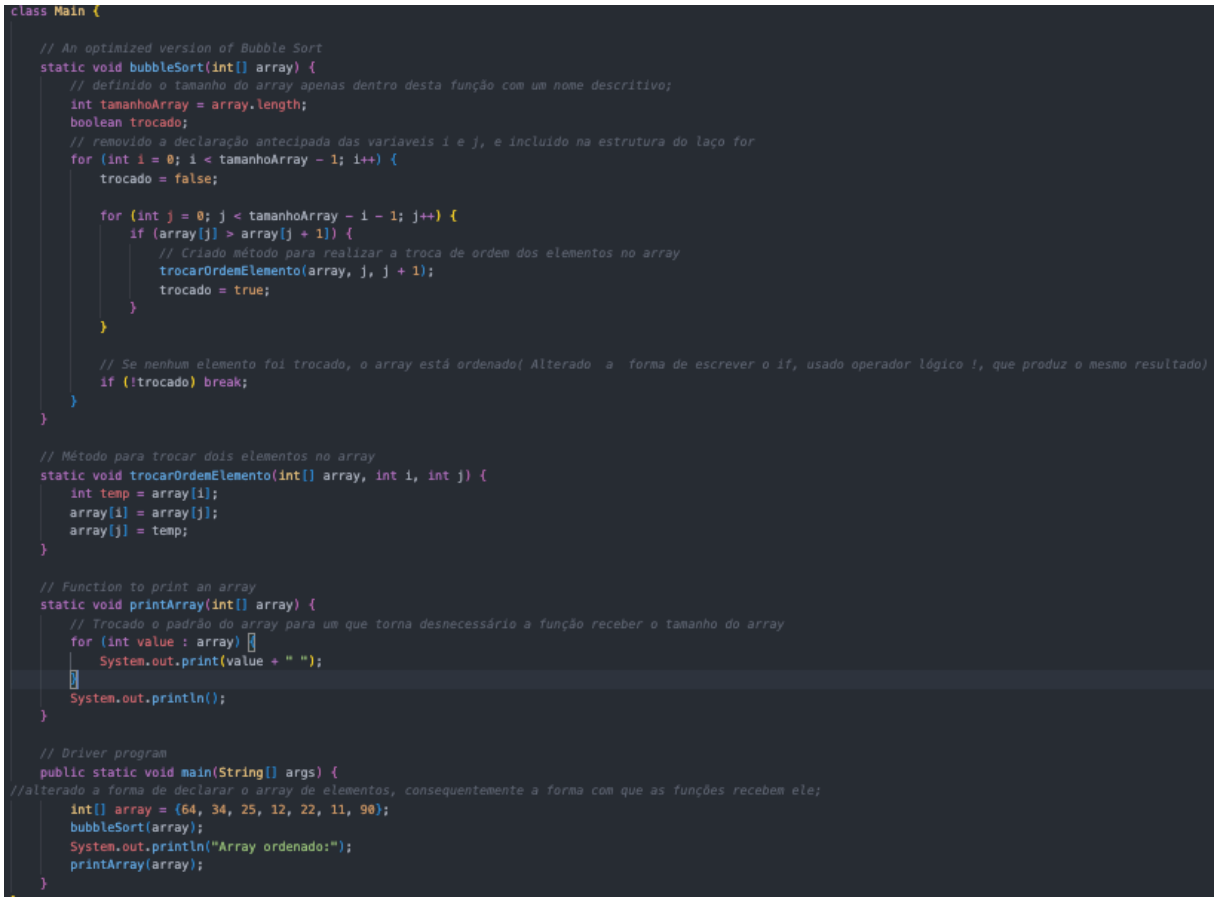
O projeto proposto na disciplina consistiu na aplicação dos princípios de *clean code* sobre um algoritmo de ordenação (Bubble Sort) Figura 13, com a exigência de, no mínimo, três alterações significativas no código original. Foram implementadas melhorias como: substituição de nomes genéricos de variáveis por nomes semânticos, extração de trechos de código repetidos em métodos específicos, e remoção de comentários desnecessários a partir da melhoria da legibilidade da lógica. Essas mudanças evidenciaram a importância da clareza, legibilidade e simplicidade no código-fonte — pilares que favorecem a comunicação entre os membros da equipe, a facilidade de manutenção e a evolução contínua do software.

A disciplina se conectou de maneira direta com outras unidades curriculares, como **I**ntrodução à Programação, ao reforçar os fundamentos da codificação estruturada e orientada a objetos, e com Testes Automatizados, ao antecipar práticas de validação e cobertura de código. Além disso, os princípios estudados foram empregados nas fases de construção dos sistemas desenvolvidos nas disciplinas de desenvolvimento web, mobile e DevOps, contribuindo para entregas mais confiáveis e sustentáveis ao longo das sprints planejadas nas disciplinas de gerenciamento ágil.

Com isso, Aspectos Ágeis de Programação consolidou conhecimentos que atravessam todo o ciclo de desenvolvimento, reforçando a importância da qualidade técnica como parte integrante da agilidade no processo de entrega de software.

7.1 ARTEFATOS DO PROJETO

FIGURA 13 -CÓDIGO BUBBLE SORTE COM CLEAN CODE APLICADO.



```java
class Main {

    // An optimized version of Bubble Sort
    static void bubbleSort(int[] array) {
        // definido o tamanho do array apenas dentro desta função com um nome descritivo;
        int tamanhoArray = array.length;
        boolean trocado;
        // removido a declaração antecipada das variaveis i e j, e incluido na estrutura do laço for
        for (int i = 0; i < tamanhoArray - 1; i++) {
            trocado = false;

            for (int j = 0; j < tamanhoArray - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    // Criado método para realizar a troca de ordem dos elementos no array
                    trocarOrdemElemento(array, j, j + 1);
                    trocado = true;
                }
            }

            // Se nenhum elemento foi trocado, o array está ordenado( Alterado  a  forma de escrever o if, usado operador lógico !, que produz o mesmo resultado)
            if (!trocado) break;
        }
    }

    // Método para trocar dois elementos no array
    static void trocarOrdemElemento(int[] array, int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    // Function to print an array
    static void printArray(int[] array) {
        // Trocado o padrão do array para um que torna desnecessário a função receber o tamanho do array
        for (int value : array) {
            System.out.print(value + " ");
        }
        System.out.println();
    }

    // Driver program
    public static void main(String[] args) {
//alterado a forma de declarar o array de elementos, consequentemente a forma com que as funções recebem ele;
        int[] array = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(array);
        System.out.println("Array ordenado:");
        printArray(array);
    }
```

FONTE: O autor (2025).

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

As disciplinas de Desenvolvimento Web 1 e 2 foram essenciais para consolidar a aplicação prática de conceitos de desenvolvimento front-end e back-end em projetos web, com ênfase na construção de sistemas completos baseados em frameworks modernos. Em WEB1, o foco esteve na criação de aplicações front-end utilizando Angular, explorando conceitos como componentes, serviços, roteamento e manipulação de dados com persistência via Local Storage. O trabalho final consistiu na implementação de dois CRUDs — de alunos e cursos — com telas de listagem Figura 14, inserção Figura 15, edição e exclusão, estruturadas com Material Design ou Bootstrap.

Em WEB2, a complexidade do projeto foi ampliada com a inclusão do back-end em Java, utilizando o framework Spring Boot com persistência em banco de dados PostgreSQL. Além dos CRUDs de alunos e cursos, foi desenvolvido um terceiro CRUD para matrículas, com relacionamento entre entidades e uso de comboboxes para seleção. A arquitetura passou a adotar uma separação mais clara entre camadas, incluindo APIs REST para comunicação entre frontend e backend.

Esses projetos proporcionaram vivência prática em construção de sistemas web reais, com atenção à usabilidade, organização de código e integração entre tecnologias. A estruturação em Angular e Spring Boot favoreceu a modularização e a escalabilidade da aplicação, alinhando-se às práticas recomendadas em ambientes ágeis. A execução em equipe também reforçou aspectos de colaboração, divisão de tarefas e integração contínua, como observado nas disciplinas de gerenciamento ágil de projetos.

A experiência obtida em WEB1 e WEB2 conectou-se diretamente com outras disciplinas do curso, como Banco de Dados (pela modelagem e persistência), UX (pela interface e fluxo do usuário), DevOps (pela organização do projeto e boas práticas de entrega), além de Testes Automatizados e Aspectos Ágeis de Programação, que reforçaram a importância da qualidade do código e validação das funcionalidades. Assim, essas disciplinas contribuíram para o desenvolvimento de um sistema funcional completo, refletindo os princípios do desenvolvimento ágil de software.

8.1 ARTEFATOS DO PROJETO

FIGURA 14 - LISTAGEM DE ALUNO.

FONTE: O autor (2025).

FIGURA 15 - CADASTRO DE ALUNO.

FONTE: O autor (2025).

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

A disciplina de UX no Desenvolvimento Ágil de Software teve como foco a integração da experiência do usuário ao processo iterativo de desenvolvimento, destacando a importância de projetar soluções centradas nas necessidades reais dos usuários. O conteúdo abordou princípios de usabilidade, design de interação, arquitetura da informação, testes com usuários e prototipação, alinhando-se aos valores ágeis de entrega contínua de valor, validação com usuários e adaptação rápida com base em feedback.

Como projeto final, foi desenvolvido o protótipo de um site para um consultório médico, com o objetivo de facilitar o agendamento de consultas e otimizar a comunicação com os pacientes. A solução propôs uma estrutura clara com acesso a campanhas de saúde, carteira de vacinação e resultados de exames, integrando funcionalidades visíveis apenas para usuários autenticados. A justificativa do projeto se baseou na necessidade de melhorar a gestão e a transparência dos serviços prestados, ao mesmo tempo em que se ampliava a captação de pacientes.

Foram elaboradas cinco telas funcionais: página inicial Figura 16, campanhas Figura 17, carteira de vacinação Figura 18, resultados de exames Figura 19 e agendamento de consultas Figura 20. As escolhas visuais priorizaram cores suaves e acolhedoras (azul e verde), tipografia legível (sans-serif) e um layout responsivo com navegação intuitiva, otimizando o uso tanto em dispositivos móveis quanto em desktops.

A validação com uma usuária real permitiu identificar pontos fortes e oportunidades de melhoria. O feedback reforçou a clareza da navegação e da visualização de exames, mas também indicou sugestões como a inclusão de alertas para novos resultados e uma seção de perguntas frequentes, demonstrando a eficácia da coleta de dados empáticos para o aprimoramento da solução.

A disciplina se integrou de forma direta com áreas como modelagem funcional, desenvolvimento web e gerenciamento de projetos, uma vez que os protótipos gerados serviram de base para implementação e planejamento técnico. Assim, UX contribuiu significativamente para a entrega de software orientado ao valor e centrado no usuário, alinhando-se aos princípios fundamentais do desenvolvimento ágil.

9.1 ARTEFATOS DO PROJETO

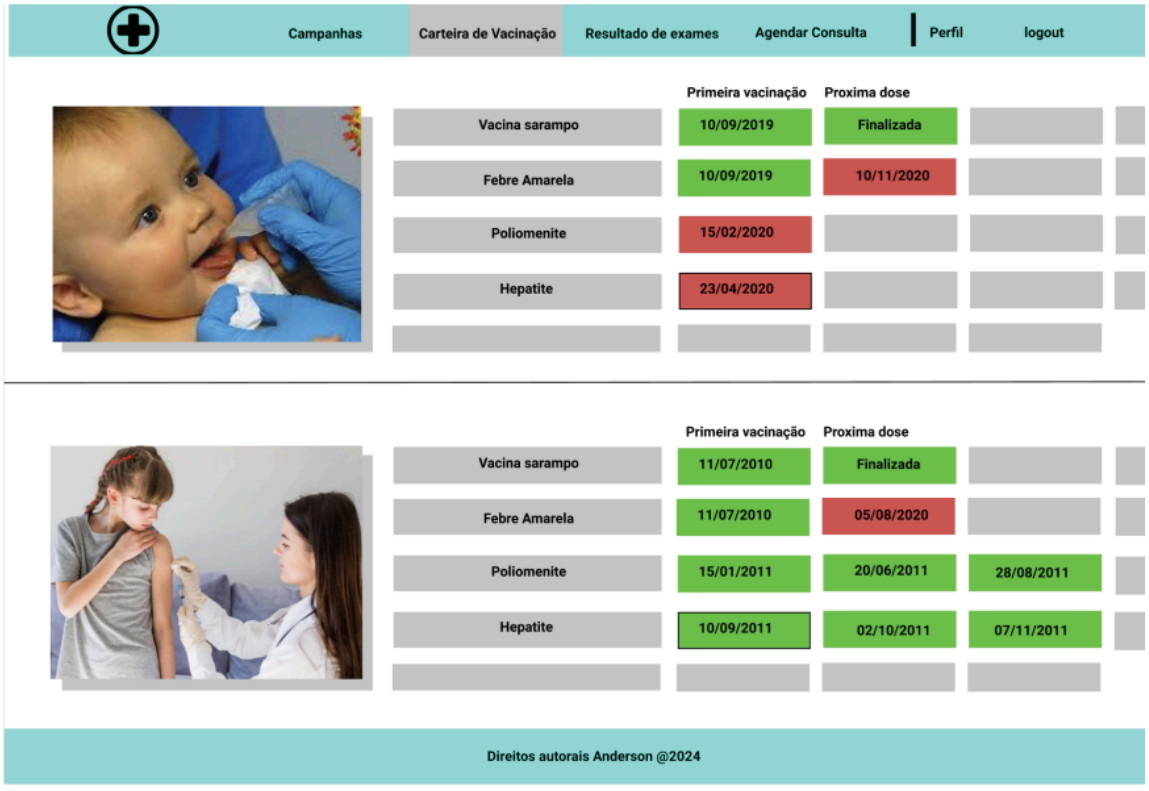
FIGURA 16 - PÁGINA INICIAL.

FONTE: O autor (2025).

FIGURA 17 - CAMPANHAS.

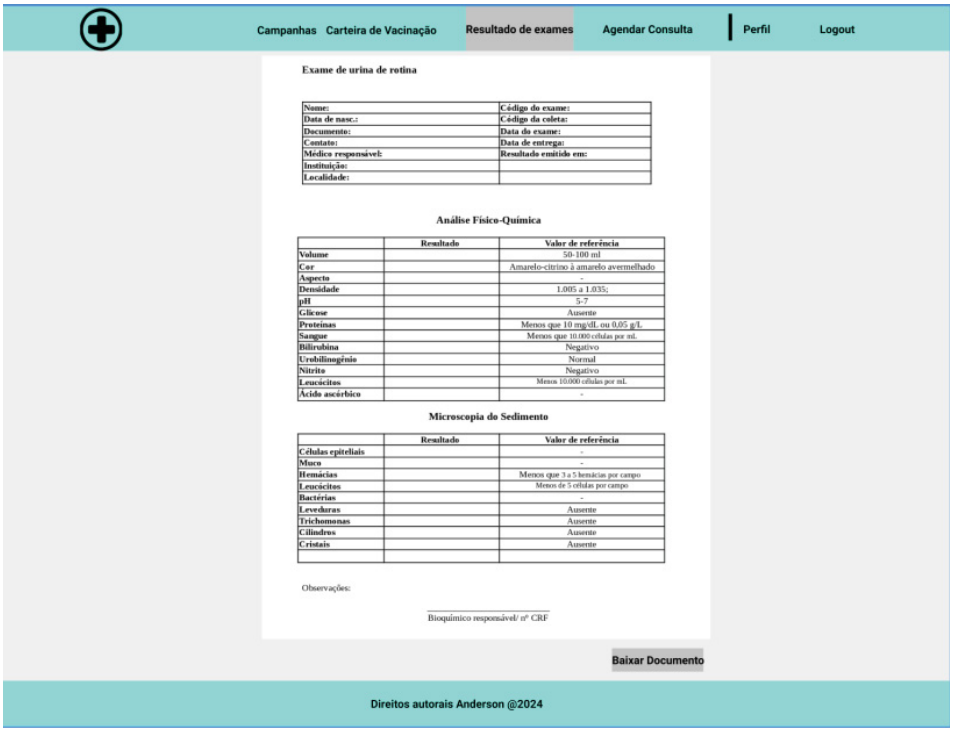
FONTE: O autor (2025).

FIGURA 18 - CARTEIRA DE VACINAÇÃO.



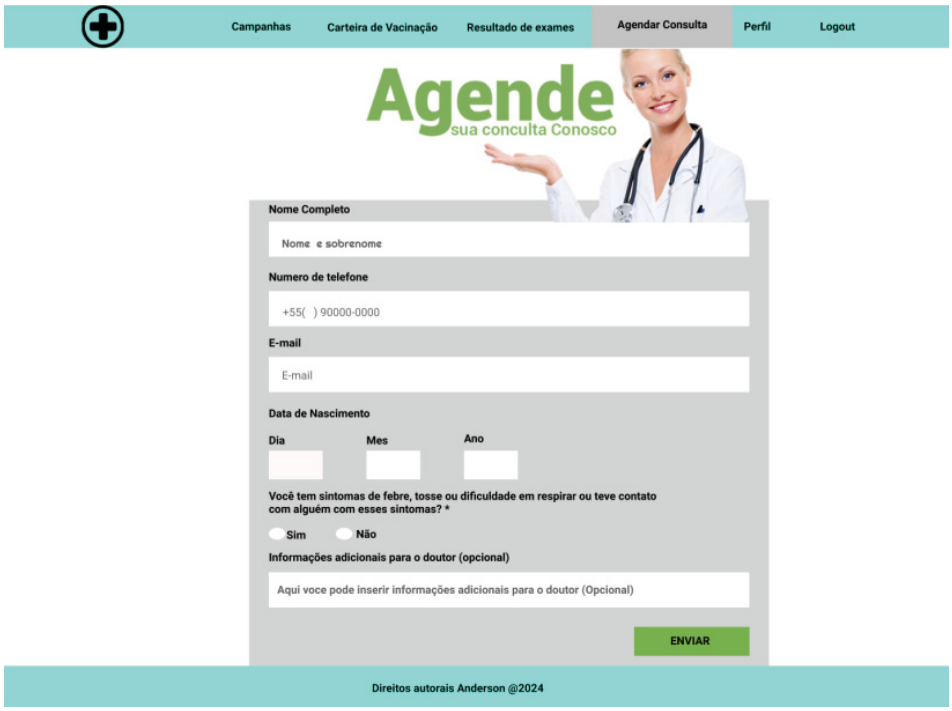
FONTE: O autor (2025).

FIGURA 19 - RESULTADO DE EXAMES.



FONTE: O autor (2025).

FIGURA 20 - AGENDAR CONSULTA.



FONTE: O autor (2025).

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

As disciplinas de Desenvolvimento Mobile 1 e 2 (MOB1 e MOB2) tiveram como principal objetivo capacitar os alunos a projetar, desenvolver e testar aplicações móveis nativas utilizando a linguagem Kotlin e a plataforma Android. No contexto do desenvolvimento ágil de software, essas disciplinas mostraram-se essenciais ao promoverem a aplicação prática de conceitos como prototipação rápida, iteração contínua e entrega incremental de valor por meio de funcionalidades bem definidas.

Em MOB1, o projeto desenvolvido foi o FinApp Figura 21, um aplicativo de controle financeiro que permitia cadastrar e visualizar movimentações de crédito e débito, simulando a dinâmica de um extrato bancário simplificado. Essa atividade foi crucial para fixar os conceitos de layouts responsivos, RecyclerView, persistência com SharedPreferences, navegação entre telas (Intent) e boas práticas de organização do código. A construção do FinApp incentivou decisões de design orientadas à experiência do usuário (UX), alinhando-se aos aprendizados da disciplina de UX e também à modelagem de dados tratada na disciplina de Banco de Dados.

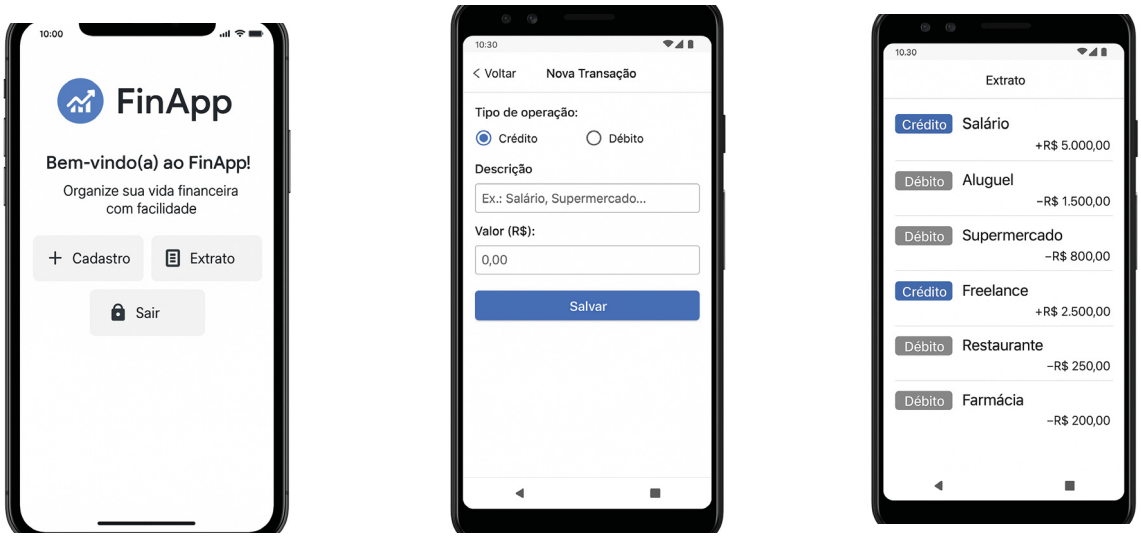
Já em MOB2, o projeto teve como base a HP-API, uma API pública com dados sobre personagens do universo Harry Potter Figura 22. O desafio foi desenvolver um aplicativo que realizasse requisições HTTP assíncronas usando corrotinas, consumindo endpoints para listar professores, alunos de casas específicas e buscar personagens por ID. Esse projeto aprofundou o uso de bibliotecas como Retrofit para chamadas HTTP e reforçou a importância da separação de responsabilidades entre camadas da aplicação, contribuindo com a compreensão de arquiteturas como MVC/MVVM — prática fundamental para integração com backends, como os abordados nas disciplinas de Web1, Web2 e Banco de Dados.

Ambos os projetos promoveram o desenvolvimento iterativo e incremental, com entregas frequentes e foco em funcionalidade. A integração entre as disciplinas foi evidente: o consumo de APIs, modelagem de dados, boas práticas de UI/UX e versionamento com GitHub formaram um ecossistema coerente de aprendizado. Assim, MOB1 e MOB2 não apenas ensinaram como construir aplicativos móveis

funcionais, mas também fortaleceram a visão sistêmica necessária para atuar em equipes ágeis multidisciplinares.

10.1 ARTEFATOS DO PROJETO

FIGURA 21 - FINAPP.



FONTE: O autor (2025).

FIGURA 22 - HARRY POTTER APP.

FONTE: O autor (2025).

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

A disciplina de Infraestrutura para Desenvolvimento e Implantação de Software teve como foco principal a aplicação dos conceitos de DevOps no contexto do desenvolvimento ágil, promovendo a integração entre desenvolvimento, entrega contínua e infraestrutura automatizada. Os conteúdos abordados trataram de virtualização, contêineres, versionamento de infraestrutura, automação de pipelines e integração contínua, pilares fundamentais para ciclos de entrega curtos, seguros e sustentáveis.

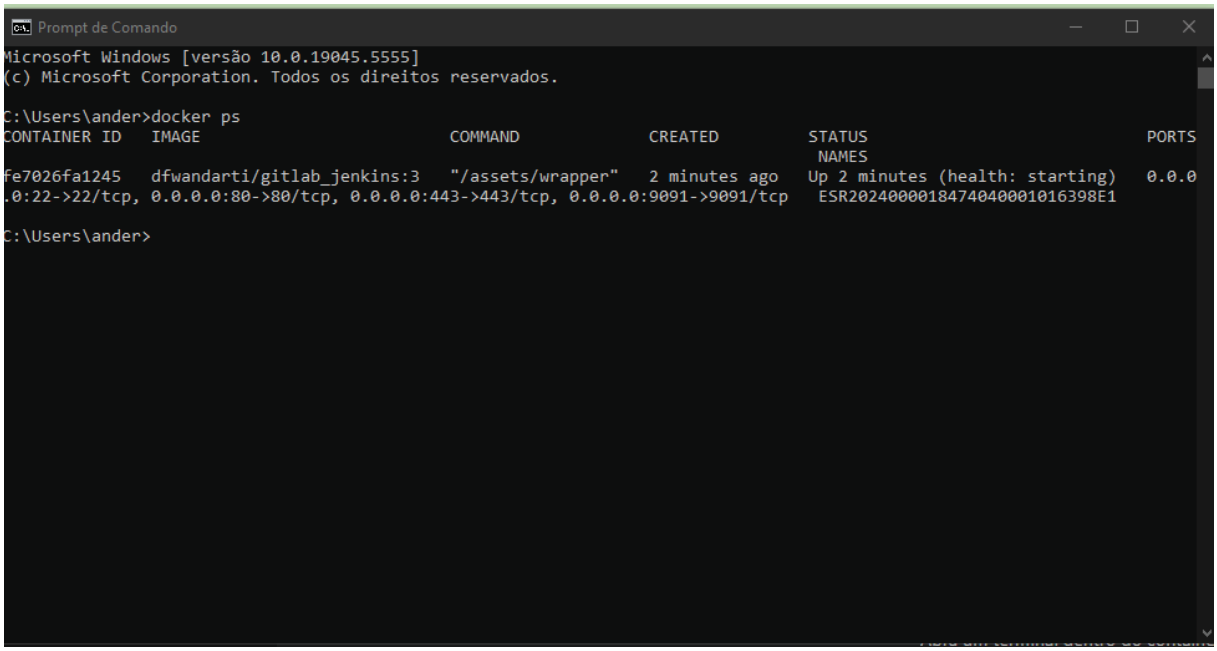
O projeto prático da disciplina consistiu na implantação de um ambiente integrado com GitLab e Jenkins a partir da imagem dfwandarti/gitlab_jenkins:3 executada via Docker. O exercício incluiu a criação de um container nomeado conforme a matrícula do aluno Figura 22, com publicação das portas 22, 80, 443 e 9091 Figura 23, além da autenticação no GitLab via root e recuperação da senha inicial dentro do container. Após o acesso, foi realizado o commit e push de um projeto no repositório, seguido da verificação do log do Git como evidência de execução correta Figura 24. Essa atividade promoveu o entendimento prático de automação e versionamento, fundamentais na cultura DevOps.

A execução desse trabalho permitiu o contato direto com ferramentas amplamente utilizadas em ambientes ágeis, como Docker, GitLab e Jenkins, reforçando práticas de CI/CD (integração e entrega contínua). Além disso, possibilitou a reflexão sobre automação de ambientes, gerenciamento de pipelines e versionamento de código, temas que impactam diretamente a confiabilidade e velocidade de entrega dos produtos desenvolvidos.

A disciplina teve integração direta com as áreas de Desenvolvimento Web, Mobile e Testes Automatizados, ao permitir que os projetos dessas disciplinas fossem executados em ambientes controlados e reprodutíveis. Também reforçou os fundamentos tratados em Banco de Dados e Programação, ao viabilizar a entrega de software funcional em múltiplos ciclos de desenvolvimento. Dessa forma, o aprendizado em DevOps se mostrou essencial para consolidar a visão de entrega contínua e colaborativa no desenvolvimento ágil de software.

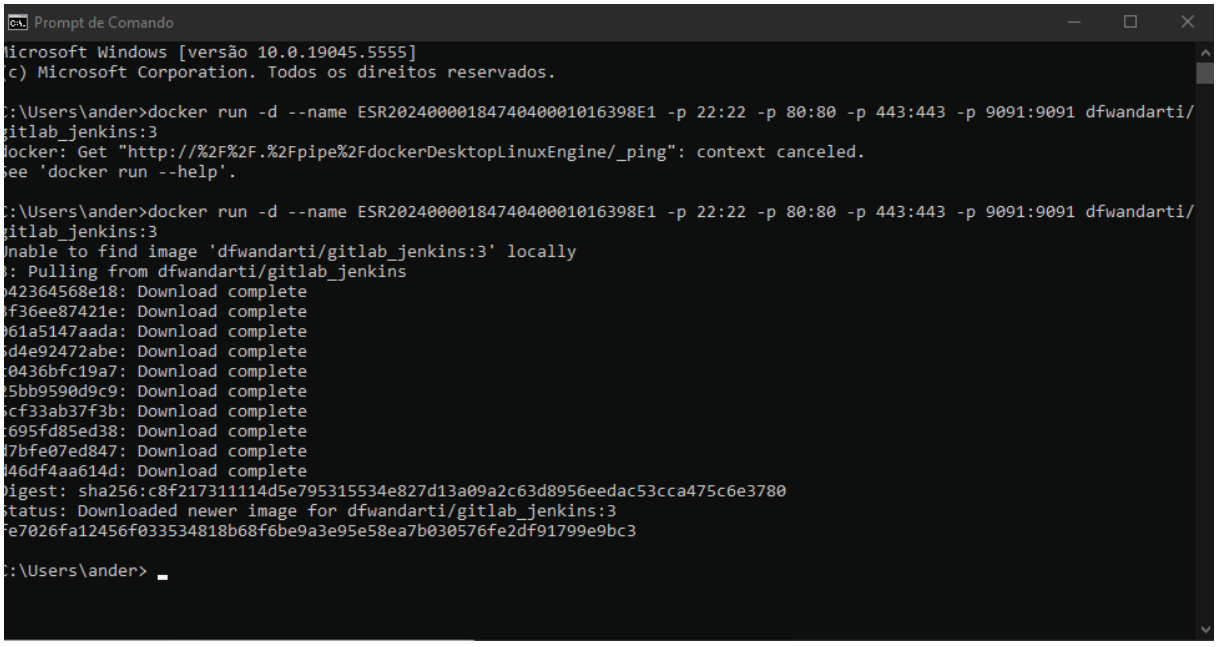
11.1 ARTEFATOS DO PROJETO

FIGURA 23 - CONTAINER CRIADO.



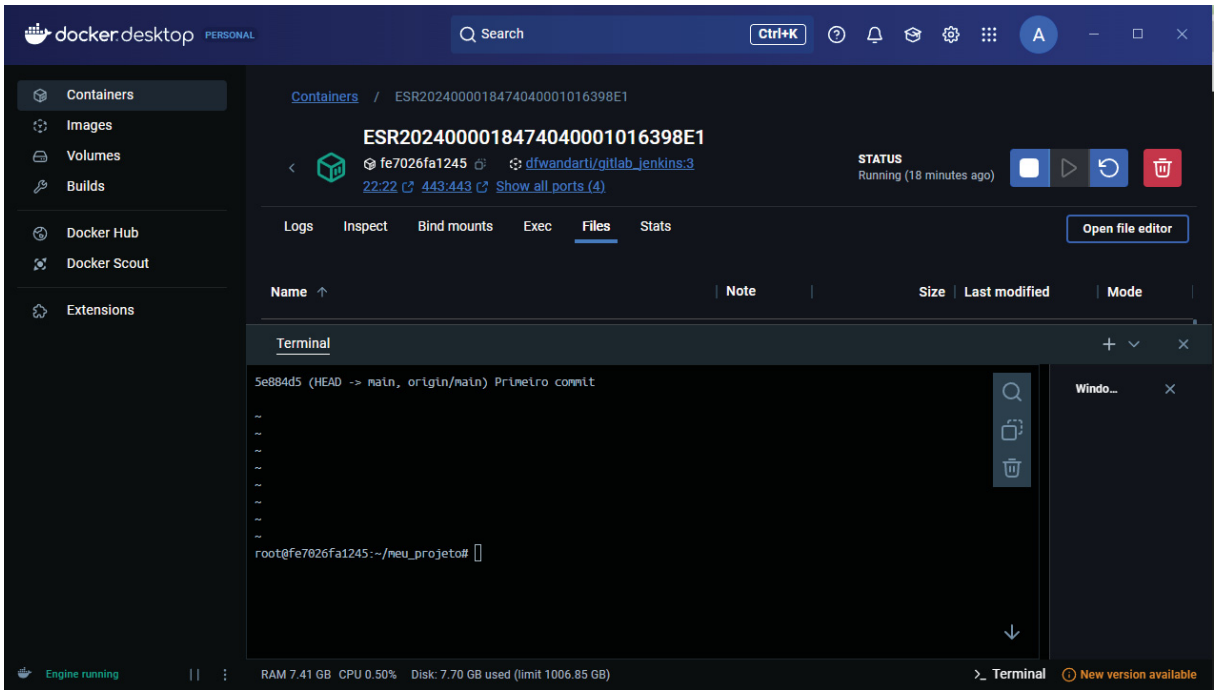
FONTE: O autor (2025).

FIGURA 24 - COMANDO PARA LIBERAR PORTAS.



FONTE: O autor (2025).

FIGURA 25 - COMMIT INICIAL.



FONTE: O autor (2025).

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados teve como objetivo central consolidar práticas de verificação e validação de software, fundamentais para garantir qualidade, confiabilidade e robustez em ciclos de desenvolvimento ágil. O conteúdo abordado introduziu os principais conceitos de testes em diferentes níveis (unitário, integração, sistema e aceitação), além de técnicas como TDD (Test-Driven Development) e BDD (Behavior-Driven Development), com foco na automação contínua do processo de testes.

O projeto da disciplina consistiu na criação de um script automatizado, utilizando a biblioteca Playwright, capaz de acessar a plataforma Anotepad, preencher automaticamente o título e o conteúdo de uma nota, e em seguida encerrar a execução. O script foi desenvolvido em JavaScript Figura 26, e demonstrou a habilidade de interagir com elementos da interface gráfica do navegador, simulando o comportamento de um usuário real. A nota criada incluiu nome e matrícula do aluno, representando a entrega automatizada de um artefato via interface web Figura 27.

A relevância desse exercício prático está na consolidação do conceito de testes de interface (E2E – end-to-end) e na aplicação de automação em processos de validação de requisitos. Em um contexto de desenvolvimento ágil, esse tipo de abordagem reduz falhas manuais, acelera a entrega e contribui diretamente para o feedback rápido e contínuo, promovendo maior confiança nas funcionalidades implementadas.

Além disso, a disciplina apresentou integração significativa com outras áreas do curso, como Programação, DevOps, Web, Mobile e UX, pois a automação de testes foi aplicada tanto no backend quanto no frontend, reforçando a importância de um pipeline de entrega completo e confiável. Assim, a disciplina de Testes Automatizados consolidou-se como uma etapa essencial na garantia da qualidade em um fluxo de desenvolvimento ágil e iterativo.

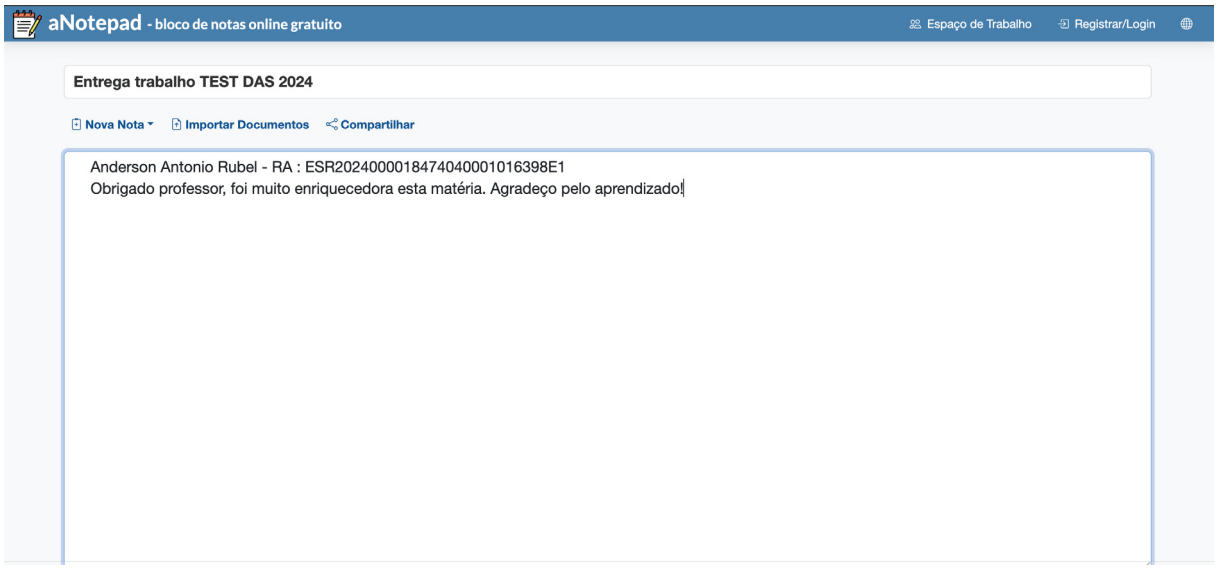
12.1 ARTEFATOS DO PROJETO

FIGURA 26 - CÓDIGO JAVASCRIPT USANDO PLAYWRIGHT.

```
const { chromium } = require("playwright");
(async () => {
  const browser = await chromium.launch({ headless: false });
  const page = await browser.newPage();

  await page.goto("https://pt.anotepad.com", {
    timeout: 60000,
    waitUntil: "domcontentloaded",
  });

  await page.waitForSelector('input[name="notetitle"]');
  const titulo = "Entrega trabalho TEST DAS 2024";
  await page.fill('input[name="notetitle"]', titulo);

  await page.waitForSelector('textarea[name="notecontent"]');
  const corpo = `
    Anderson Antonio Rubel - RA : ESR2024000018474040001016398E1
    Obrigado professor, foi muito enriquecedora esta matéria. Agradeço pelo aprendizado!
  `;
  await page.fill('textarea[name="notecontent"]', corpo);

  await page.waitForTimeout(10000);

  await browser.close();
})();
```

FONTE: O autor (2025).

FIGURA 27 -RESULTADO DA EXECUÇÃO DO CÓDIGO PLAYWRIGHT.



FONTE: O autor (2025).

13 CONCLUSÃO

O presente memorial reuniu as principais atividades desenvolvidas ao longo do curso de Especialização em Desenvolvimento Ágil de Software, apresentando a integração entre teoria e prática por meio dos projetos realizados em cada disciplina. Os conteúdos abordaram desde os fundamentos dos métodos ágeis até a entrega contínua de soluções funcionais, envolvendo áreas como modelagem, programação, experiência do usuário, testes, DevOps, gerenciamento de projetos, banco de dados e desenvolvimento para web e dispositivos móveis.

A disciplina de Métodos Ágeis serviu como base conceitual, guiando a condução dos projetos em ciclos iterativos e incrementais. As disciplinas de Gerenciamento Ágil (GAP1 e GAP2) viabilizaram o planejamento de sprints, releases e o uso de quadros visuais como o Kanban. Modelagem Funcional e Estrutural permitiram traduzir os requisitos do sistema em artefatos visuais e estruturados. Introdução à Programação e Banco de Dados forneceram as bases técnicas para codificação e manipulação de dados. Aspectos Ágeis de Programação contribuiu com práticas de clean code, promovendo legibilidade e manutenção. Desenvolvimento Web e Mobile possibilitaram a construção de aplicações com foco em responsividade, navegabilidade e consumo de APIs. UX trouxe a perspectiva do usuário ao centro do processo de design, e Infraestrutura com DevOps permitiu consolidar práticas de entrega contínua. Por fim, a disciplina de Testes Automatizados reforçou a importância da validação contínua do software desenvolvido.

Dentre os principais desafios observados para a adoção prática do desenvolvimento ágil, destacam-se: a necessidade de mudança cultural nas equipes para abraçar a colaboração e a adaptabilidade; a disciplina na execução dos rituais ágeis; a integração fluida entre áreas técnicas e de negócio; e o domínio das ferramentas necessárias para garantir a automação dos processos e a entrega contínua de valor. A execução dos projetos permitiu vivenciar essas dificuldades na prática, ao mesmo tempo em que reforçou o papel fundamental da comunicação, da iteração constante e da validação contínua no sucesso de um projeto ágil.

14 REFERÊNCIAS

ANDERSON, David J. *Kanban: successful evolutionary change for your technology business*. Sequim: Blue Hole Press, 2010.

BECK, Kent et al. *Manifesto for Agile Software Development*, 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 11 ago. 2025.

COHN, Mike. *Succeeding with agile: software development using Scrum*. Boston: Addison-Wesley, 2010.

KIM, Gene; HUMBLE, Jez; DEBOIS, Patrick; WILLIS, John. *The DevOps handbook*. Portland: IT Revolution Press, 2016.

NIELSEN, Jakob; BUDIU, Raluca. *Mobile usability*. Berkeley: New Riders, 2012.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILLE, Ian. *Engenharia de software*. 9. ed. São Paulo: Pearson Addison-Wesley, 2011.