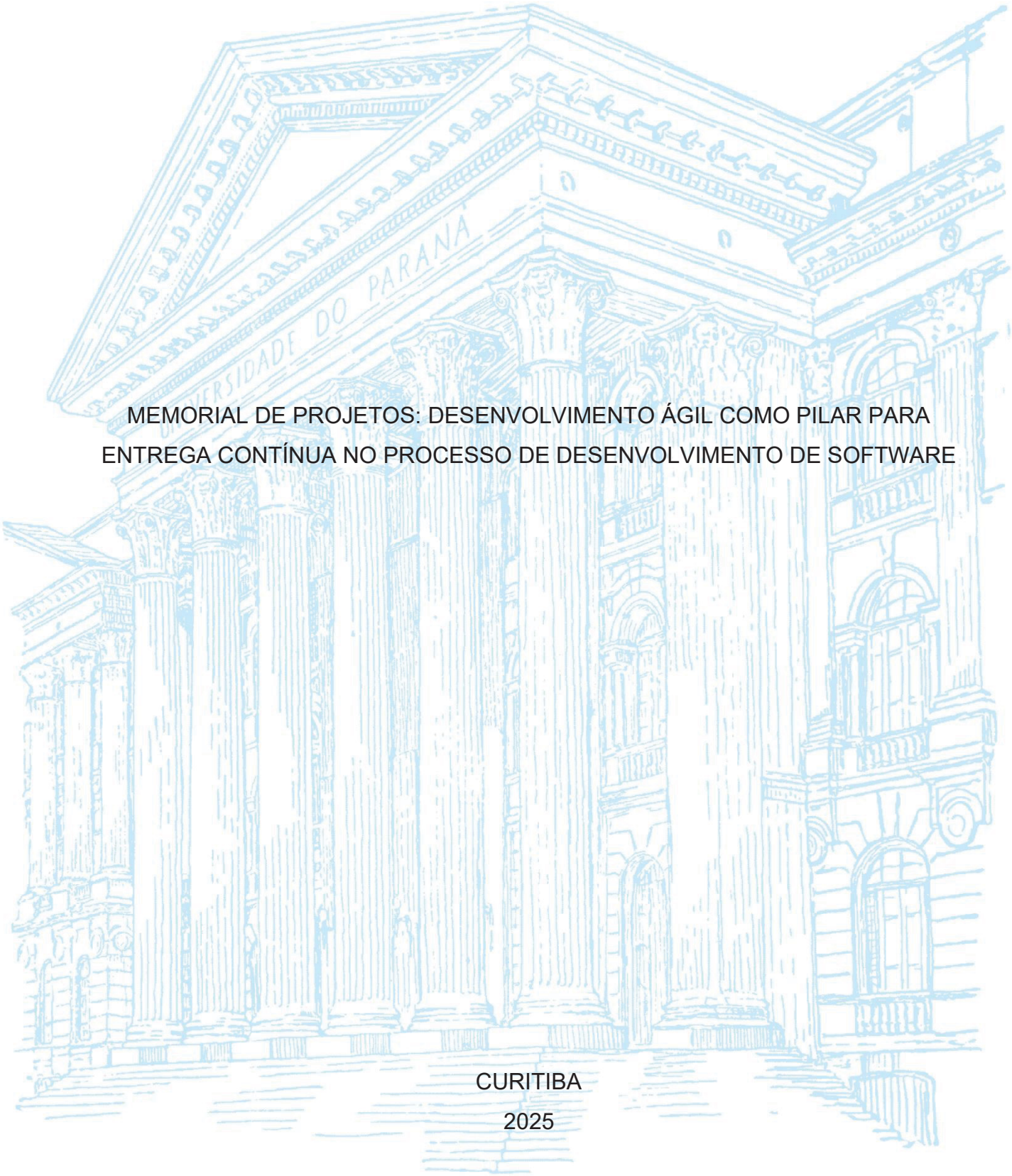
UNIVERSIDADE FEDERAL DO PARANÁ

MARCOS FELIPE CARVALHO



MEMORIAL DE PROJETOS: DESENVOLVIMENTO ÁGIL COMO PILAR PARA ENTREGA CONTÍNUA NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

CURITIBA

2025

MARCOS FELIPE CARVALHO

MEMORIAL DE PROJETOS: DESENVOLVIMENTO ÁGIL COMO PILAR PARA ENTREGA CONTÍNUA NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

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

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Desenvolvimento Ágil de Software da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **MARCOS FELIPE CARVALHO**, intitulada: **MEMORIAL DE PROJETOS: DESENVOLVIMENTO ÁGIL COMO PILAR PARA ENTREGA CONTÍNUA NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa.

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

Curitiba, 07 de Novembro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Rua Doutor Alcides Vieira Arco-verde - Curitiba - Paraná - Brasil
CEP 81520-260 - Tel: (41) 98874-0472 - E-mail: jaimewo@ufpr.br

RESUMO

O presente memorial tem como objetivo apresentar a integração conceitual e prática dos projetos desenvolvidos ao longo do curso de Pós-Graduação em Desenvolvimento Ágil de Software da Universidade Federal do Paraná. O trabalho reúne as atividades realizadas nas diversas disciplinas, evidenciando como os princípios e valores ágeis orientaram tanto o processo de aprendizado quanto a aplicação prática na construção de soluções de *software*. Por meio dos projetos desenvolvidos — que envolveram modelagem, programação, testes automatizados, DevOps, experiência do usuário e desenvolvimento *mobile* — foi possível compreender de forma integrada o ciclo de vida de um *software*, desde a concepção até a entrega contínua de valor ao usuário. O memorial demonstra a relevância da colaboração, da organização de equipes e da adoção de práticas iterativas e incrementais como pilares essenciais para a eficiência e a qualidade das entregas. Dessa forma, o documento consolida o aprendizado adquirido durante o curso e reforça o papel do desenvolvimento ágil como base para a criação de soluções tecnológicas sustentáveis, evolutivas e centradas no cliente.

Palavras-chave: práticas ágeis; ciclo de vida; entrega contínua; automação; integração de sistemas.

ABSTRACT

This memorial aims to present the conceptual and practical integration of the projects developed throughout the Postgraduate Course in Agile Software Development at the Federal University of Paraná. The work brings together the activities carried out in the various disciplines, showing how agile principles and values guided both the learning process and the practical application in building software solutions. Through the developed projects—which involved modeling, programming, automated testing, DevOps, UX, and mobile development—it was possible to comprehensively understand the software life cycle, from conception to the continuous delivery of value to the user. The memorial demonstrates the relevance of collaboration, team organization, and the adoption of iterative and incremental practices as essential pillars for the efficiency and quality of deliveries. Thus, the document consolidates the learning acquired during the course and reinforces the role of agile development as a foundation for creating sustainable, evolutionary, and customer-centric technological solutions.

Keywords: agile practices; lifecycle; continuous delivery; automation; systems integration.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 PARECER TÉCNICO

O presente parecer técnico tem como objetivo integrar, de forma conceitual e prática, os projetos desenvolvidos ao longo do curso de Pós-Graduação em Desenvolvimento Ágil de Software da Universidade Federal do Paraná. Este documento consolida os artefatos produzidos nas disciplinas, analisando-os sob a ótica dos princípios, valores e práticas que sustentam o desenvolvimento ágil, conforme definido por Beck (2001) no Manifesto Ágil. Assim, busca-se demonstrar a evolução do aprendizado, o alinhamento entre teoria e prática e a contribuição de cada disciplina para a formação de uma visão sistêmica sobre o ciclo de vida de um projeto de *software*.

O curso possibilitou compreender de forma integrada todas as etapas do processo de desenvolvimento de *software*, desde a modelagem e concepção da solução até a implantação e manutenção contínua. Disciplinas como Métodos Ágeis de Desenvolvimento de *Software* e Gerenciamento Ágil de Projetos forneceram a base metodológica necessária para compreender a importância da adaptação, colaboração e foco na entrega de valor ao cliente. Já Modelagem Ágil de *Software* e Introdução à Programação possibilitaram aplicar os conceitos teóricos em práticas concretas de análise e implementação, consolidando o entendimento sobre requisitos, abstração e orientação a objetos.

As disciplinas Banco de Dados, Aspectos Ágeis de Programação e Desenvolvimento *Web* reforçaram a importância da qualidade técnica e da clareza de código como pilares da agilidade. O uso de práticas como *Test-Driven Development, Clean Code* e Refatoração Contínua (Pressman; Maxim, 2021), favoreceu a construção de soluções robustas e de fácil manutenção. Já em *User Experience* no Desenvolvimento Ágil de Software, compreendeu-se que a entrega de valor vai além do código, envolvendo também a experiência do usuário como elemento central do processo de desenvolvimento.

Complementando o ciclo, as disciplinas Infraestrutura para Desenvolvimento e Implantação de Software e Testes Automatizados apresentaram a visão moderna de Integração e Entrega Contínua (RedHat, 2025), consolidando a importância da automação, do controle de versões e da cultura de *feedback* para garantir qualidade e velocidade nas entregas. Essas práticas demonstraram que o desenvolvimento ágil

é sustentado por um ecossistema técnico que une desenvolvimento, operações e testes em um fluxo contínuo de melhoria.

De forma geral, o Memorial de Projetos evidencia a evolução do aluno ao longo do curso, tanto no domínio técnico quanto na adoção da mentalidade ágil. As atividades realizadas permitiram desenvolver competências de colaboração, comunicação e adaptação, fundamentais para atuar em equipes multidisciplinares e em contextos dinâmicos. O conjunto de projetos produzidos demonstra não apenas a aplicação prática dos conceitos aprendidos, mas também a consolidação de uma postura profissional voltada à melhoria contínua e à entrega de valor.

Assim, este parecer técnico reflete o amadurecimento obtido durante o curso, no qual teoria e prática se complementam para formar um profissional capaz de compreender o desenvolvimento de *software* de forma holística, integrada e ágil. O aprendizado adquirido evidencia que o sucesso em projetos de *software* não depende apenas de ferramentas ou metodologias, mas da capacidade de colaborar, aprender e adaptar-se continuamente às necessidades do cliente e às transformações tecnológicas.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

Durante o desenvolvimento deste projeto, foi possível compreender as diferentes formas de desenvolver e, principalmente, de gerenciar o processo de desenvolvimento de *software*. Por meio do mapa mental elaborado, visualizou-se esse processo de ponta a ponta: desde o entendimento da solução até a sua implementação e liberação para os usuários. Segundo Sommerville (2011), esse fluxo pode ser norteado por quatro etapas fundamentais: especificação, projeto e implementação, validação do software e evolução.

Modelos de processo de *software* são representações simplificadas de um processo a partir de uma perspectiva específica (Sommerville, 2011). As formas de condução e gestão adotadas definem como a equipe realiza e reporta as entregas ao cliente. Durante o curso, foram abordadas duas abordagens principais de condução de projetos: Tradicional e Ágil.

Os métodos tradicionais seguem um processo estruturado e bem definido, no qual o cliente tem uma visão clara de todas as etapas, incluindo o cronograma. Nesse modelo, o cliente passa por um processo completo de entendimento do projeto e, ao final, recebe o *software* como um pacote fechado. Essa abordagem contempla diferentes modelos, como: Modelo Cascata, Prototipação, Espiral e Incremental.

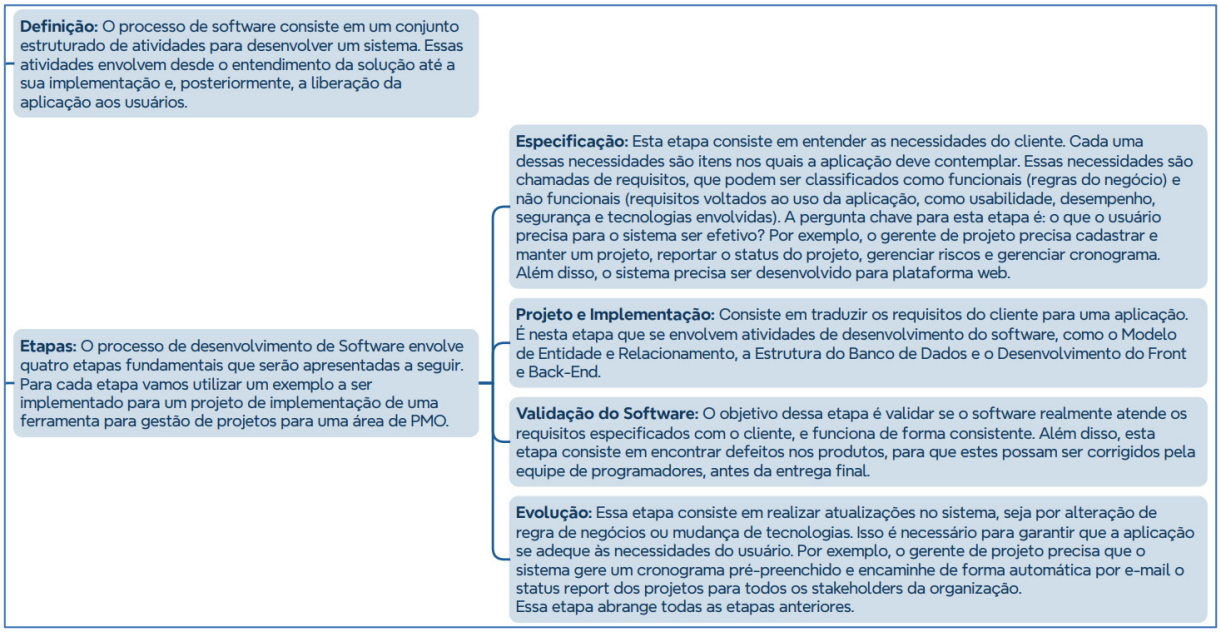
Em contrapartida, o modelo ágil prioriza a entrega contínua de valor ao cliente. Essa abordagem é fundamentada em quatro princípios (Beck *et al*., 2001): Indivíduos e interações mais que processos e ferramentas; *Software* em funcionamento mais que documentação abrangente; Colaboração com o cliente mais que negociação de contratos; responder a mudanças mais que seguir um plano.

Existem diversas metodologias ágeis, como *Scrum, Extreme Programming* e Entrega Contínua, entre outras.

Por fim, é importante destacar que não existe um modelo único ou universalmente correto. A escolha entre as abordagens Ágil ou Tradicional dependerá das necessidades específicas do cliente, da complexidade do projeto e da forma como se deseja conduzir o desenvolvimento.

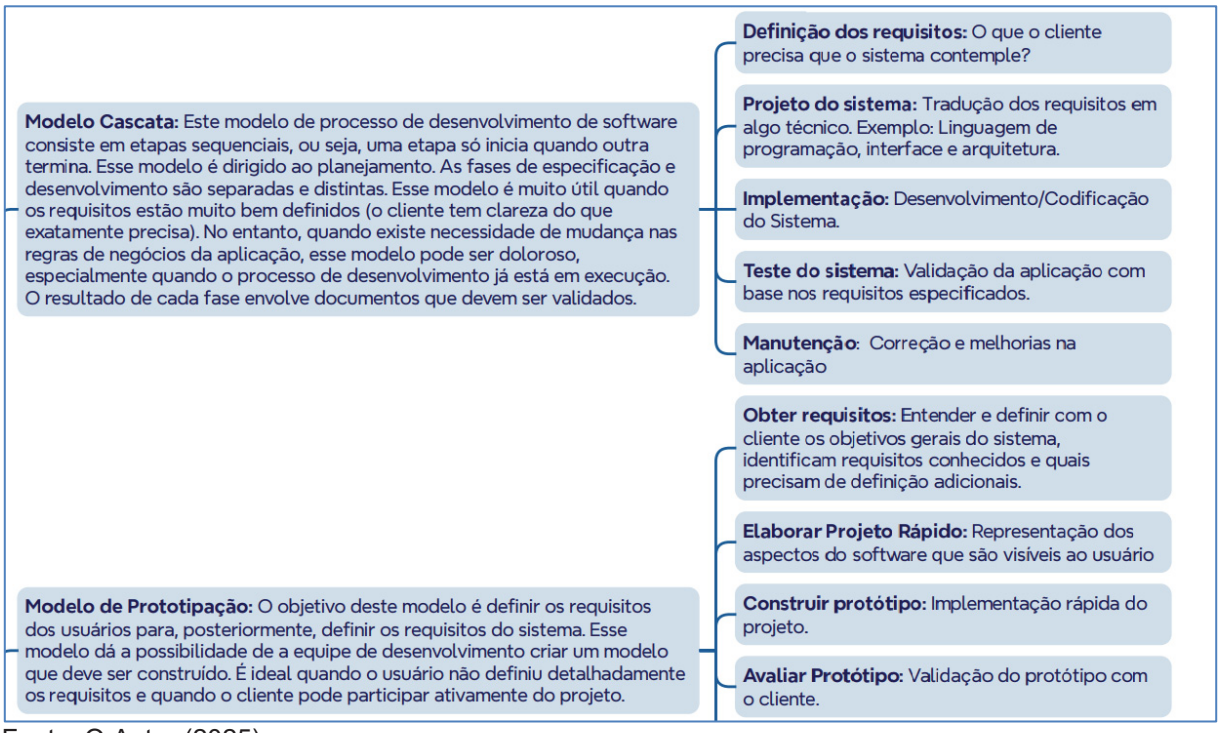
2.1 ARTEFATOS DO PROJETO

Figura 1 - Processo de Desenvolvimento de *Software*



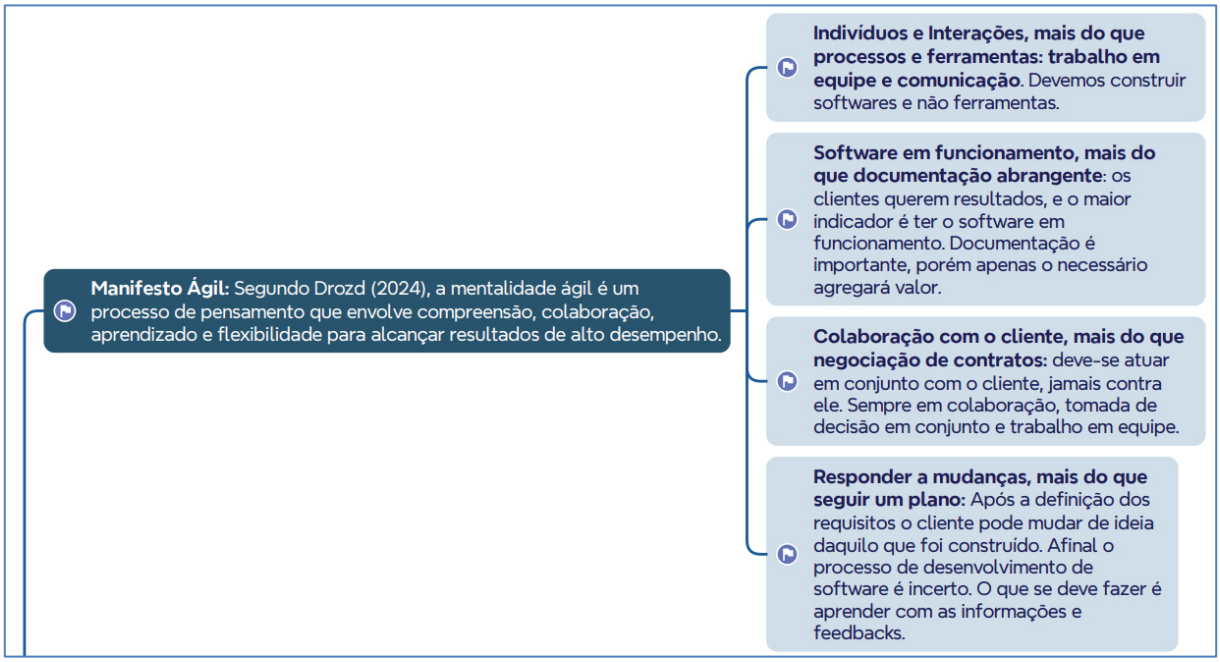
Fonte: O Autor (2025)

Figura 2 - Modelos de Desenvolvimento de Software



Fonte: O Autor (2025)

Figura 3 - Manifesto Ágil



Fonte: O Autor (2025)

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

A documentação em projetos de *software* conduzidos por métodos ágeis não significa ausência de registros. Pelo contrário, Segundo (Beck *et al.*, 2001), o *software* deve estar funcionando, é mais importante do que documentação abrangente implica que se deve documentar apenas o necessário, com foco na utilidade e na contribuição direta para o funcionamento do sistema.

Segundo Mota (2015), um *software* de boa qualidade só pode ser obtido se estiver em conformidade com os requisitos estabelecidos. Assim, para o sistema de gestão de condomínios, buscamos extrair e explicitar os requisitos, reunindo o máximo de informações para compreender o problema. A partir disso, passamos para a etapa de Análise do Projeto de *Software*, com a construção dos modelos.

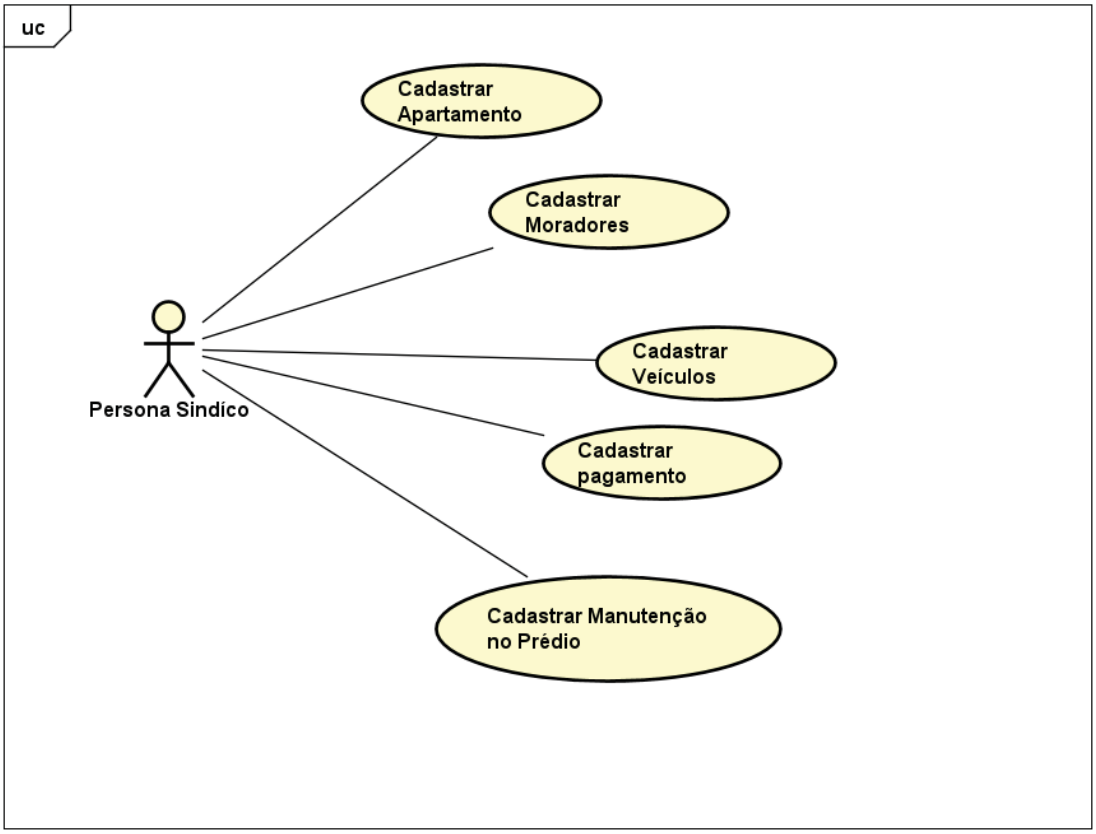
De acordo com Rumbaugh *et al.* (2006, p.1), um modelo é uma abstração de algo, cujo propósito é permitir compreendê-lo antes de sua construção. Utilizamos a UML (*Unified Modeling Language*), notação voltada à especificação, visualização e documentação de sistemas baseados em análise orientada a objetos.

No estudo de caso do Sistema de Condomínio, foram empregados os seguintes artefatos: Diagrama de Casos de Uso, Histórias de Usuário, Diagrama de Classes e Diagrama de Sequência. Esses elementos possuem importância significativa no contexto do Desenvolvimento Ágil: o Diagrama de Casos de Uso facilita a compreensão da interação do usuário com o sistema; as Histórias de Usuário representam, de forma clara e objetiva, as expectativas e comportamentos esperados das funcionalidades; o Diagrama de Classes mapeia a estrutura lógica do sistema; e o Diagrama de Sequência detalha os eventos e operações acionadas em cada interação.

Este estudo reforça que, no desenvolvimento ágil, a documentação não deve ser negligenciada. Pelo contrário, ela deve ser construída de forma colaborativa entre a equipe de desenvolvimento e o cliente a cada incremento, com o objetivo de melhorar a qualidade do software e acelerar o processo de entrega.

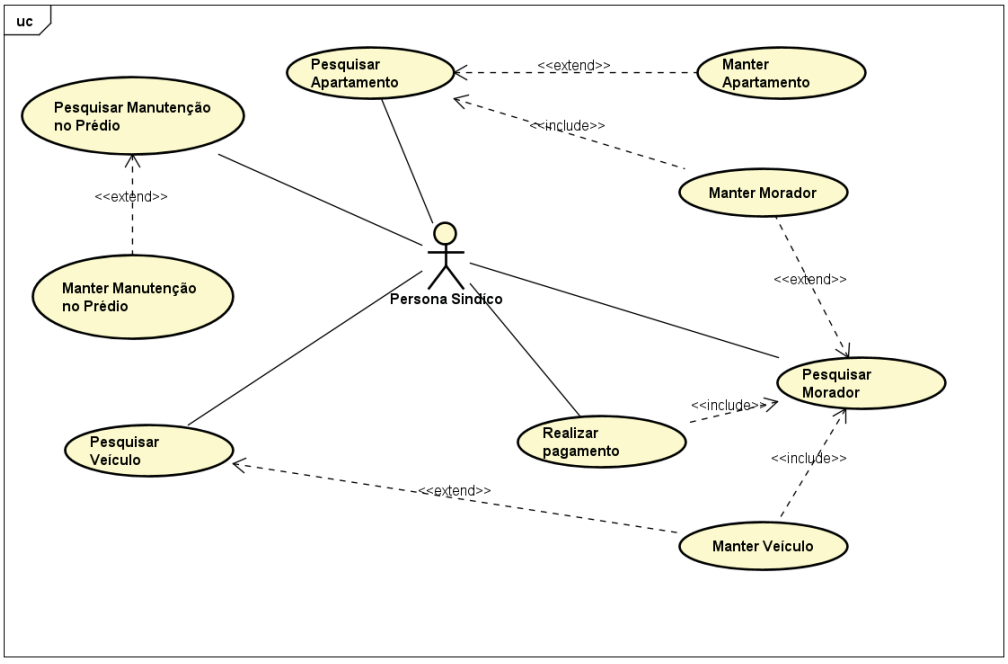
3.1 ARTEFATOS DO PROJETO

Figura 4 - Diagrama de Caso de Uso Sindíco – Nível 1



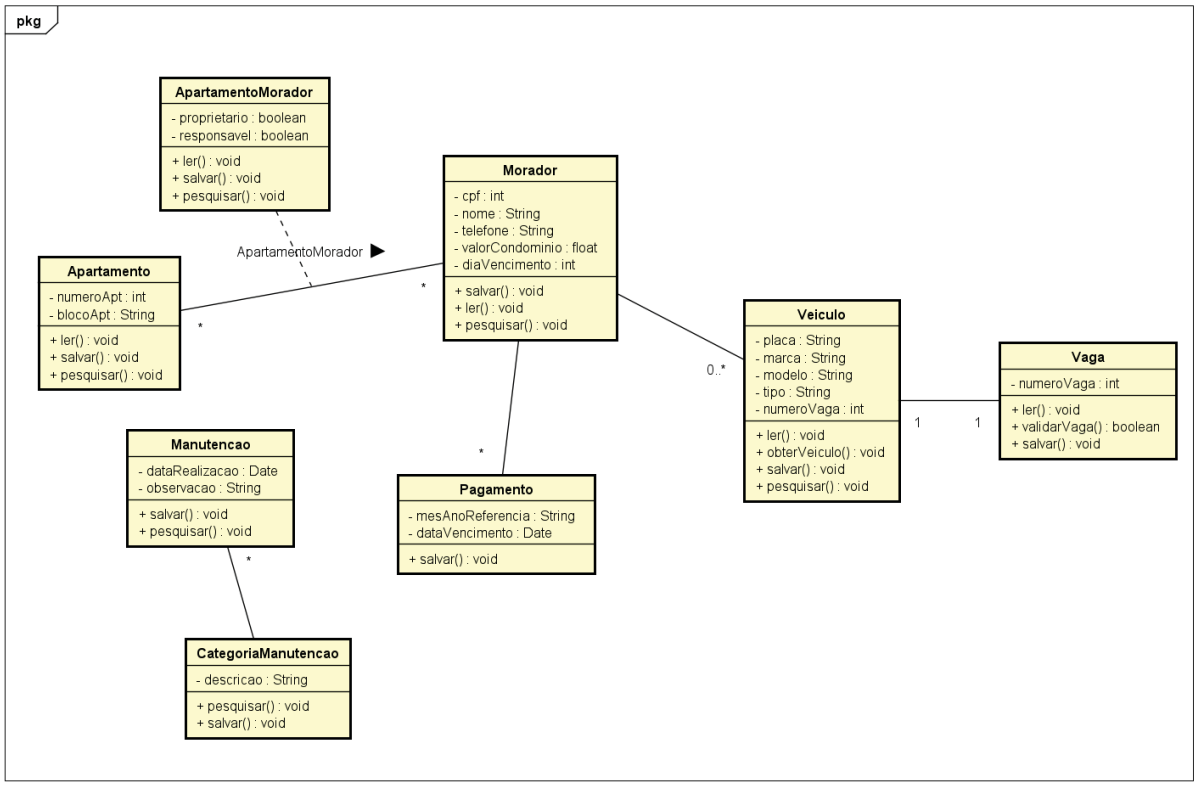
Fonte: O Autor (2025)

Figura 5 - Diagrama de Caso de Uso Sindíco – Nível 2



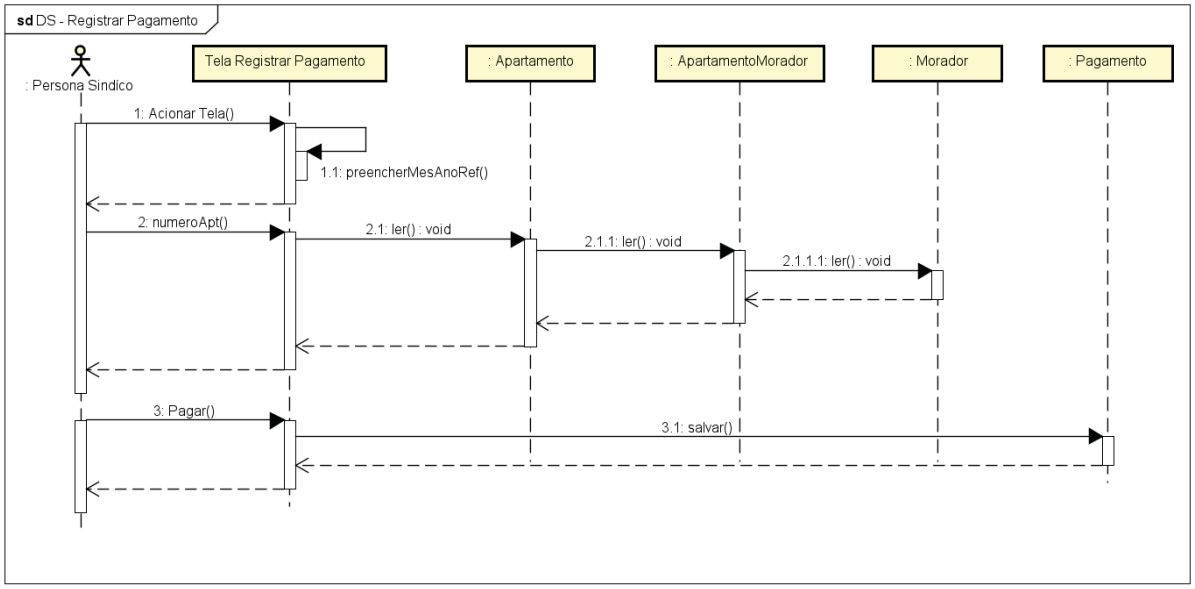
Fonte: O Autor (2025)

Figura 6 - Diagrama de Classe - Gestão de Condomínio



Fonte: O Autor (2025)

Figura 7 - Diagrama de Sequência - Gestão de Condomínio



Fonte: O Autor (2025)

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

Um projeto é um empreendimento temporário, com início e fim definidos, criado para gerar um produto, serviço ou resultado único e exclusivo (PMI, 2017). Para isso, utilizou-se o Gerenciamento de Projetos, que, de acordo com PMI (2017, p.10), é a aplicação de conhecimento, habilidades, ferramentas e técnicas a uma ampla gama de atividades para atender aos requisitos de um determinado projeto.

Durante a disciplina de GAP 1, foi desenvolvido um plano de *release* para o projeto Gestão de Atividades e Base de Indicadores, uma aplicação voltada à área de Processos e Melhoria Contínua do Sistema FIEP. Seu objetivo é apoiar o controle de demandas, projetos e atividades da equipe, além de oferecer relatórios que dão visibilidade aos gestores sobre as iniciativas em andamento e o desempenho do time. De acordo com Sommerville (2011), o plano de *release* é essencial no desenvolvimento ágil, pois permite planejar e prever a entrega de histórias de usuário com base na velocidade da equipe, na capacidade de trabalho e nas prioridades do produto.

Já na disciplina de Gerenciamento de Projetos Ágeis 2, utilizou-se a ferramenta *Kanban Board Game* (2025), que, de forma intuitiva, pode-se compreender de forma prática do sistema visual de gestão do fluxo de trabalho. Aprendeu-se sobre a importância de monitorar o *Work in Progress* (WIP), ou seja, os itens em desenvolvimento, e como limitar o WIP contribui para identificar gargalos e manter um fluxo estável (Freitas, 2025)

Além disso, a disciplina de Modelagem de Software foi fundamental para definir os requisitos e as atividades necessárias ao atendimento das necessidades do cliente. Por fim, nas disciplinas de Gerenciamento de Projetos 1 e 2, expõem-se ferramentas e técnicas que garantem a entrega do *software* dentro do prazo, com qualidade e conforme o orçamento previsto, reforçando a integração entre abstração e condução do projeto de software no desenvolvimento ágil.

4.1 ARTEFATOS DO PROJETO

Quadro 1 - Cálculo de Velocidade

Horas disponíveis por dia:	4 horas	**Tamanho da Sprint:**	3 semanas (15 dias)
Horas disponíveis por Sprint:	120 horas	**Velocidade:**	7 pontos por Sprint

Fonte: O Autor (2025)

Quadro 2 - Plano de Release

Iteração/Sprint 1	**Iteração/Sprint 2**	**Iteração/Sprint 3**	**Iteração/Sprint 4**
Data Início: 01/04/2024	**Data Início: 22/04/2024**	**Data Início: 13/05/2024**	**Data Início: 03/06/2024**
Data Fim: 19/04/2024	**Data Fim: 10/05/2024**	**Data Fim: 31/05/2024**	**Data Fim: 21/06/2024**
HU001 SENDO Gestor de Processos e Melhoria Contínua QUERO manter demandas e projetos PARA que os dados fiquem atualizados ESTIMATIVA (1 ponto)	HU008 SENDO Gestor de Processos e Melhoria Contínua QUERO uma métrica de quantidade horas realizadas pela minha equipe PARA acompanhar a alocação ESTIMATIVA (1 ponto)	HU013 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO uma métrica de quantidade de demandas por status PARA verificar os projetos no prazo, com tendência de atraso e atrasadas ESTIMATIVA (2 pontos)	HU017 SENDO Gestor de Processos e Melhoria Contínua QUERO acessar o capacity de cada analista no software GABI, PARA que possa entender melhor a distribuição de trabalho e alocar recursos de forma eficiente. ESTIMATIVA (3 pontos)

HU002 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO pesquisar demandas PARA realizar manutenção nos dados ESTIMATIVA (1 ponto)	HU009 SENDO Gestor de Processos e Melhoria Contínua QUERO um indicador PARA acompanhar horas previstas e realizadas das demandas visando analisar o desvio do planejamento ESTIMATIVA (2 pontos)	HU014 SENDO Gestor de Processos e Melhoria Contínua QUERO uma métrica quantidade de demandas ao longo do tempo PARA analisar os períodos que houve ociosidade e sobrecarga da equipe ESTIMATIVA (1 ponto)	HU018 SENDO Gestor de Processos e Melhoria Contínua QUERO exportar o status report semanal dos projetos em powerpoint PARA automatizar e diminuir o tempo de dedicação da equipe nessa tarefa. ESTIMATIVA (3 pontos)
HU003 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO pesquisar atividades das demandas PARA manter os dados atualizados ESTIMATIVA (1 ponto)	HU010 SENDO usuário do Gestão de Atividades e Base de Indicadores QUERO de visualizar a data de última atualização da demanda PARA identificar as demandas movimentadas ESTIMATIVA (1 ponto)	HU015 SENDO Gerente Executivo QUERO de encaminhar para os meus subordinados demandas através do aplicativo PARA organizar o fluxo de demandas. ESTIMATIVA (1 ponto)	

HU004 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO manter as atividades PARA realizar manutenção nos dados ESTIMATIVA (1 PONTO)	HU011 SENDO Gestor de Processos e Melhoria Contínua QUERO de definir responsáveis para os pontos de atenção PARA mostrar em qual alçada de aprovação se encontram as pendências das demandas. ESTIMATIVA (1 ponto)	HU016 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO um repositório de documentos para os anexos inseridos nas atividades PARA consultar os materiais ESTIMATIVA (3 pontos)	
HU005 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO apontar horas nas atividades PARA registrar o tempo que dediquei na demanda ESTIMATIVA (1 PONTO)	HU012 SENDO Membro de Equipe de Processos e Melhoria Contínua QUERO ser notificado quando um projeto está prestes a começar PARA organizar minhas atividades ESTIMATIVA (2 pontos)		
HU006 Sendo Membro de Equipe de			

Processos e Melhoria Contínua QUERO manter os pontos de atenção das demandas PARA realizar manutenção nos dados e identificar possíveis ameaças nos objetivos dos projetos ESTIMATIVA (1 PONTO)			

Fonte: O Autor (2025)

Figura 8 - Kanban Board Game Execução de Sprints

Fonte: O Autor (2025)

Figura 9 - Kanboard Game Resultado

Fonte: O Autor (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

Durante a disciplina de Introdução à Programação executou-se os fundamentos do desenvolvimento de *software*. Embora o repasse de conteúdos em disciplinas especificas, a programação dessa disciplina mostrou de forma intuitiva e lúdica os conceitos de Linguagem de Programação, Tipo de Dados, Orientação a Objetos, TDD e Banco de Dados.

Segundo Gotardo (2015), a Linguagem de Programação é um conjunto de instruções sintáticas e semânticas para dizer os computadores quais tarefas e passos e precisa executar. Sendo assim, como atividade principal, desenvolveu-se um sistema de conta corrente utilizando a linguagem Java, conhecida por sua robustez, segurança e portabilidade. A estruturação do código com base no paradigma de orientação a objetos nos permitiu aplicar princípios como encapsulamento, herança e reutilização.

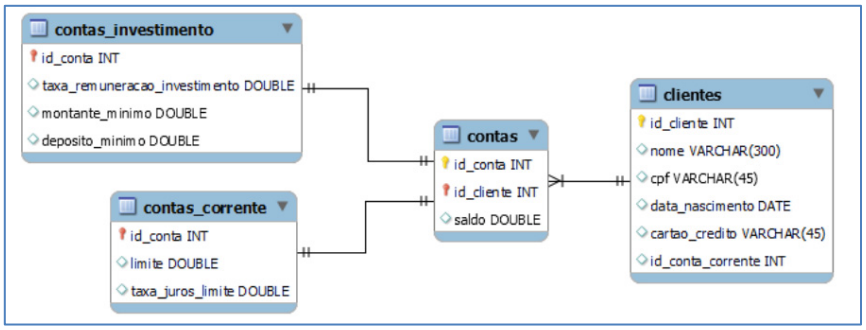
A aplicação de TDD (Beck, 2003), por meio do framework JUnit (JUnit, 2025), nos proporcionou contato direto com uma das práticas mais valorizadas nas metodologias ágeis, favorecendo entregas contínuas e com qualidade garantida. Além disso, a integração com banco de dados (Spring, 2025) permitiu o armazenamento e manipulação das informações, simulando cenários reais de desenvolvimento.

Esse projeto se conectou de maneira prática com as disciplinas MAG1 e MAG2 (Modelagem Ágil), ao estruturar classes e fluxos baseados em requisitos; e GAP1 e GAP2 (Gerenciamento Ágil), ao vivenciar, ainda que de forma inicial, a organização e o planejamento de tarefas com foco em entregas parciais e funcionais.

Dessa forma, a disciplina não apenas introduziu os aspectos técnicos da programação, mas também estabeleceu um alicerce para a atuação em times ágeis, integrando teoria e prática de forma coerente com os princípios do desenvolvimento ágil de *software*.

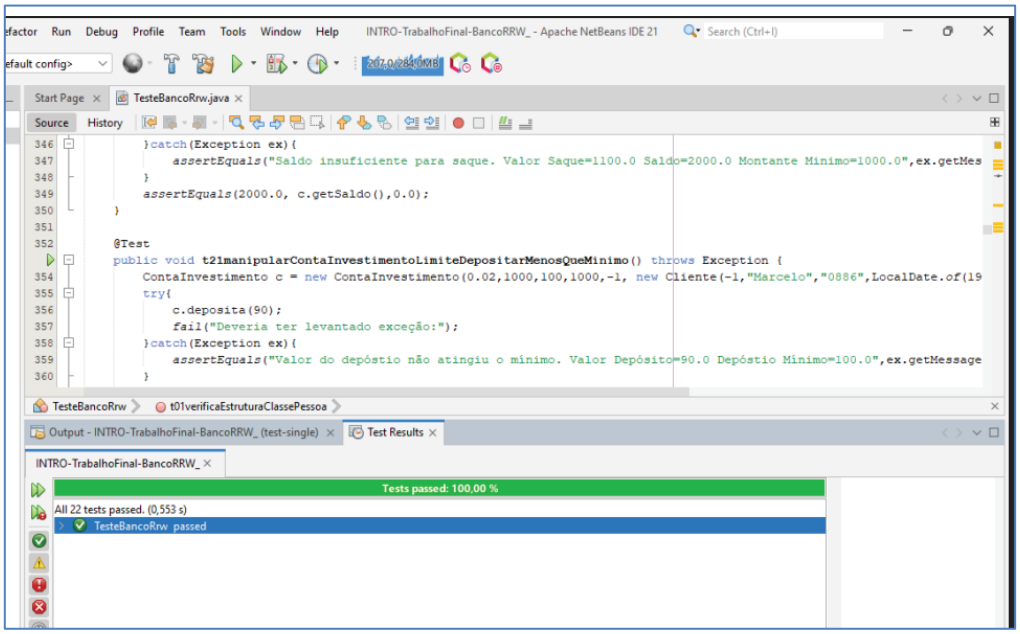
5.1 ARTEFATOS DO PROJETO

Figura 10 - Entidade Relacionamento - Conta Bancária



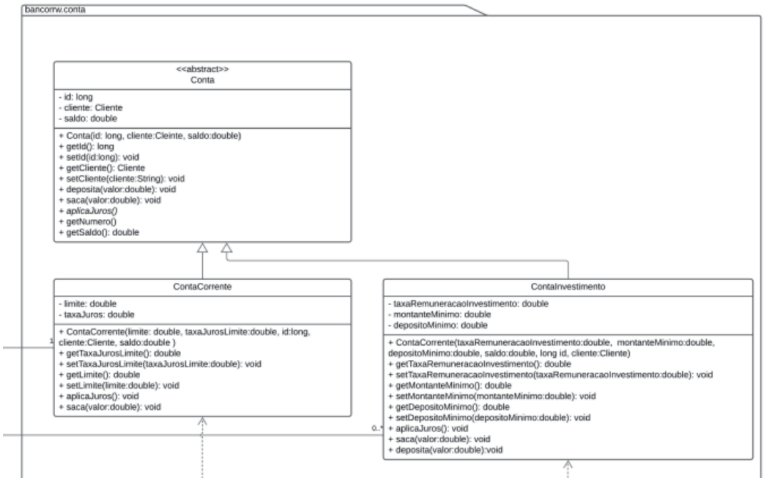
Fonte: O Autor (2025)

Figura 11 - Execução de Testes utilizando TDD



Fonte: O Autor (2025)

Figura 12 - Diagrama de Classes Conta Bancária



Fonte: O Autor (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

Segundo Chen (1976, p. 10), o Modelo Entidade-Relacionamento (MER) permite a representação conceitual de sistemas de informação, abstraindo regras de negócio e identificando restrições sobre os dados. A disciplina de Banco de Dados teve como objetivo introduzir os conceitos fundamentais de modelagem e gerenciamento de dados, essenciais para qualquer projeto de software. Com base no MER e nos modelos lógico e físico de dados (Elmasri; Navathe, 2016, p. 45-60), foi desenvolvido como estudo de caso um banco de dados voltado para o gerenciamento do controle e aplicação de vacinas em colaboradores das indústrias do Paraná. A empresa XPTO foi contratada pelo Governo do Paraná para aplicar vacinas nos colaboradores das indústrias do estado.

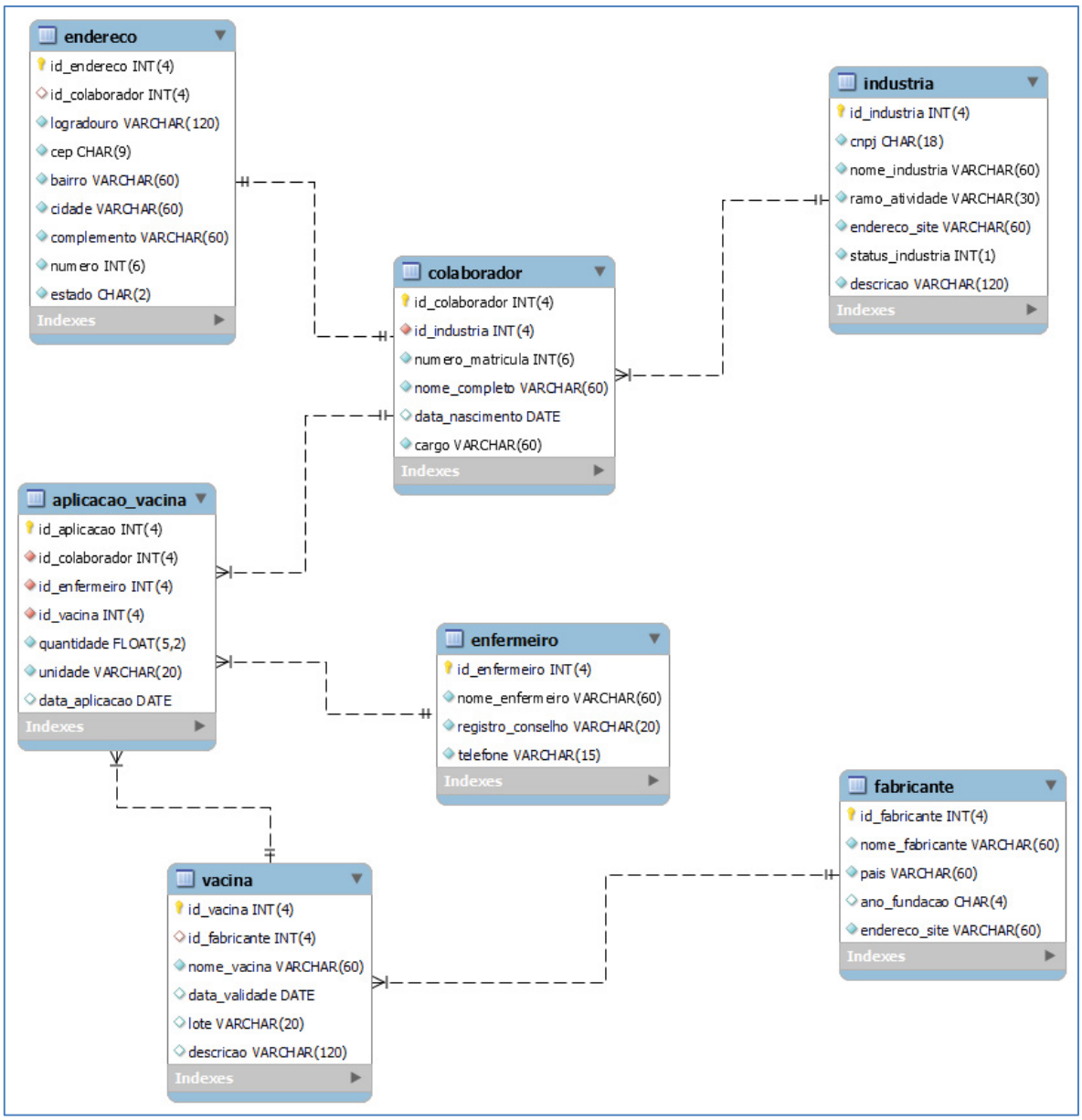
A modelagem conceitual (Elmasri; Navathe, 2016), representada pelo Modelo Entidade Relacionamento, foi utilizada para abstrair as regras de negócio do sistema de vacinas e entender restrições e modelo lúdico onde os dados devem estar estruturados. Em seguida, a modelagem lógica (Elmasri; Navathe, 2016) serviu para obter essas abstrações em estruturas mais detalhadas, como tipos de dados e relacionamentos com cardinalidades, preparando o banco para ser implementado em um Sistema Gerenciados de Banco de Dados, ou SGBD (SGBD, 2025). Por fim, a modelagem física (Elmasri; Navathe, 2016) representou a implementação concreta no MySQL (MYSQL, 2025). É uma descrição de um banco de dados no nível de abstração visto pelo usuário, e nesse momento utilizamos a linguagem SQL (*Structured Query Language*) (W3Schools, 2025), especialmente suas sublinguagens DDL (*Data Definition Language*) e DML (*Data Manipulation Language*) (Elmasri; Navathe, 2016), para criar tabelas e relacionamentos e para realizar a manipulação dos dados. O projeto incluiu a criação de diversas entidades, como Indústrias, Colaboradores, Vacinas e Enfermeiros.

Essa disciplina teve bastante integração com as disciplinas de MAG1 e MAG2, principalmente em relação ao diagrama de classes, pois o diagrama de entidade-relacionamento, embora apresente algumas diferenças sutis em sua aplicação, possui, em tese, conceitos semelhantes. A principal diferença está no relacionamento muitos-para-muitos, que é pouco usual no banco de dados; nesse caso, utiliza-se uma tabela intermediária.

Além disso, essa disciplina teve ligação com a disciplina de Introdução à Programação, nos conceitos relacionados à Orientação a Objetos, especialmente no que diz respeito à abstração, que consiste em extrair elementos do mundo real e traduzi-los para um modelo de software — neste caso, um modelo para armazenamento de dados.

6.1 ARTEFATOS DO PROJETO

Figura 13 - Modelo de Entidade Relacionamento



Fonte: O Autor (2025)

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

Segundo Pressman (2021), o desenvolvimento de projetos de *software*, considera-se quatro macroetapas, conforme abordado na disciplina de Métodos Ágeis de Desenvolvimento de *Software*: Especificação, Projeto e Implementação, Validação do *Software* e Evolução. Ao se tratar das três últimas etapas, é necessário pensar no programa como uma estrutura que seja fácil de desenvolver enquanto o *software* está em produção, mas também de manter e evoluir após sua implantação.

A disciplina de Aspectos Ágeis teve como objetivo compreender temas como Código Limpo e Boas Práticas, Práticas Ágeis no Código, Colaboração e Métodos de Programação. O estudo de caso dessa disciplina consistiu em alterar o algoritmo *Bubble Sort,* aplicando três modificações para deixá-lo mais próximo dos princípios de clean code. Para isso, foram utilizados alguns princípios, como:

- Nomes significativos: Alteração das variáveis *temp* para *posicaoTemporaria* e *n* para *tamanhoElementos*, visando revelar a intenção de uso.
- Remoção de comentários desnecessários: Foram excluídos comentários óbvios, como o da função *print array*, substituindo-os por nomes de funções mais intuitivos.
- Formatação: Ajuste na indentação e separação de blocos, além de modificação de condicionais *if* para uma única linha, quando apropriado.

Segundo Azevedo (2023), o código limpo melhora a escalabilidade e flexibilidade do software, facilitando a implementação de novas funcionalidades e a adaptação a requisitos em constante mudança. Além disso, McConnell (2004) reafirma que, sistemas desenvolvidos com boas práticas são mais fáceis de corrigir, melhorar e adaptar às novas demandas, pois possuem menos dependências ocultas e seguem padrões consistentes.

A disciplina também retomou conceitos vistos em matérias de modelagem, como o *Gherkin* (Cucumber, 2025), que antes era utilizado para critérios de aceite em testes de alto nível e, nesta abordagem, foi aplicado diretamente na programação via código. Além disso, reforçou princípios do Manifesto Ágil por meio de práticas como *Pair Programming* (Agile, 2025) *e Mob Programming* (Mob Programming, 2025), destacando a importância da colaboração e comunicação, já que a equipe atua no mesmo código, promovendo excelência técnica, simplicidade e autonomia.

7.1 ARTEFATOS DO PROJETO

```
// Implementação otimizada em Java do Bubble sort
// Código extraído de https://www.geeksforgeeks.org/bubble-sort/

class BubbleSort {

    // An optimized version of Bubble Sort
    static void bubbleSort(int arr[], int tamanhoVetor)
    {
        int i, j, temp;
        boolean trocado = false;

        for (i = 0; i < tamanhoVetor - 1; i++) {
            for (j = 0; j < tamanhoVetor - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    trocado = true;
                }
            }

            if (!trocado) break;
        }

    }

    static void exibirValoresOrdenados(int arr[], int tamanhoVetor)
    {
        int i;

        for (i = 0; i < tamanhoVetor; i++)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

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

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

Durante as disciplinas de Desenvolvimento *Web* 1 e 2, utilizou-se o *framework* Angular (Angular, 2025), uma das principais tecnologias para a construção de aplicações de página única (SPA) (Pressman; Maxim, 2021), baseado em *Typescript* (Typescript, 2025), um superconjunto do Javascript. No primeiro módulo, o projeto consistiu na implementação de uma aplicação de cadastro, leitura, edição e exclusão de dados utilizando como entidades Alunos e Cursos. Essa experiência possibilitou explorar recursos essenciais do Angular, como rotas, modelos, serviços, módulos e componentes, que juntos estruturam uma aplicação organizada e escalável.

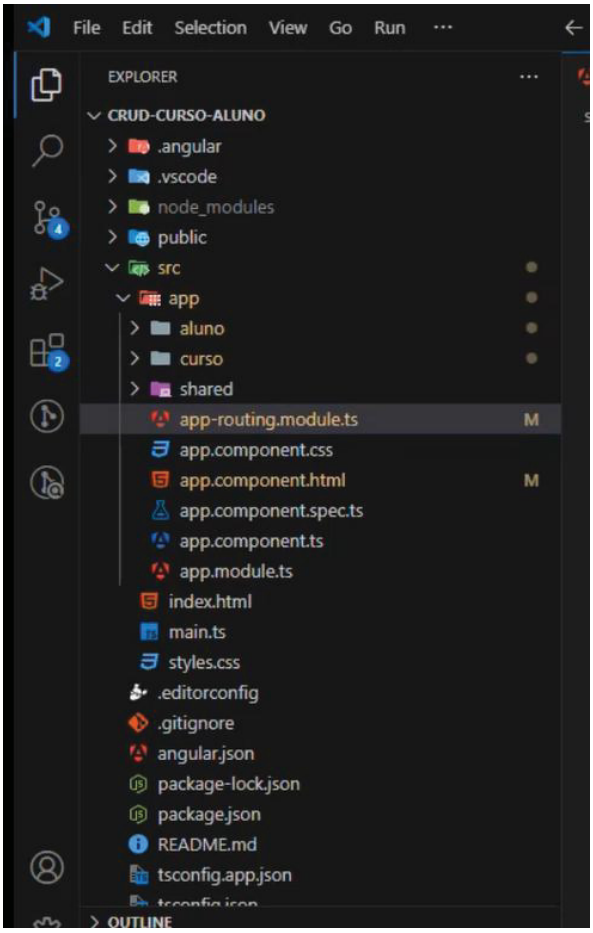
No segundo módulo, o estudo de caso foi aprofundado com a integração ao back-end, utilizando banco de dados e o Spring Boot (Spring, 2025) para a criação de microserviços. Essa etapa ampliou a complexidade do projeto, permitindo compreender a importância da comunicação entre *front-end* e *back-end* e reforçando conceitos de persistência de dados.

A disciplina se integrou diretamente com Introdução à Programação, que forneceu as bases lógicas e de orientação a objetos; com Banco de Dados, essencial para estruturar e manipular as informações; e com Aspectos Ágeis de Programação, que orientou práticas como clareza de código, simplicidade e refatoração. Essa integração reforçou o aprendizado prático e mostrou a relevância do desenvolvimento ágil, que favorece a entrega contínua, flexibilidade e qualidade no software desenvolvido.

Assim, o estudo de Desenvolvimento *Web* foi fundamental para consolidar o aprendizado do curso, aplicando teoria e prática em um ambiente alinhado às necessidades atuais do mercado.

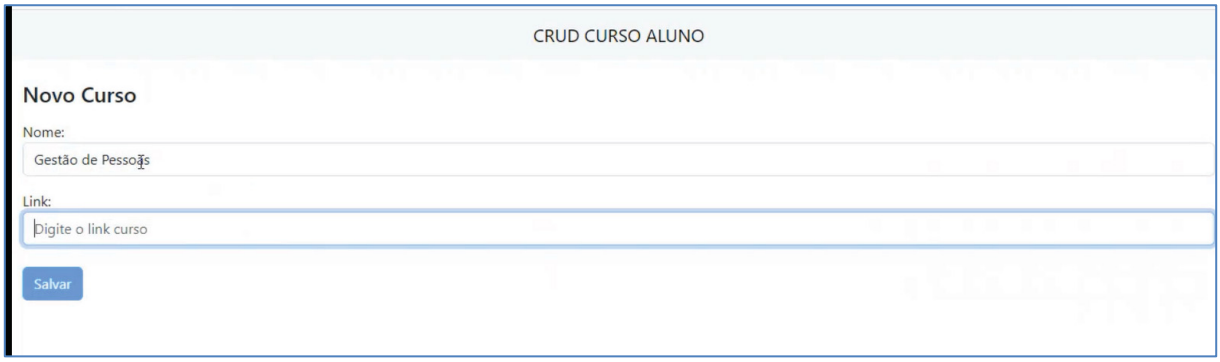
8.1 ARTEFATOS DO PROJETO

Figura 14 - Inclusão de Aluno



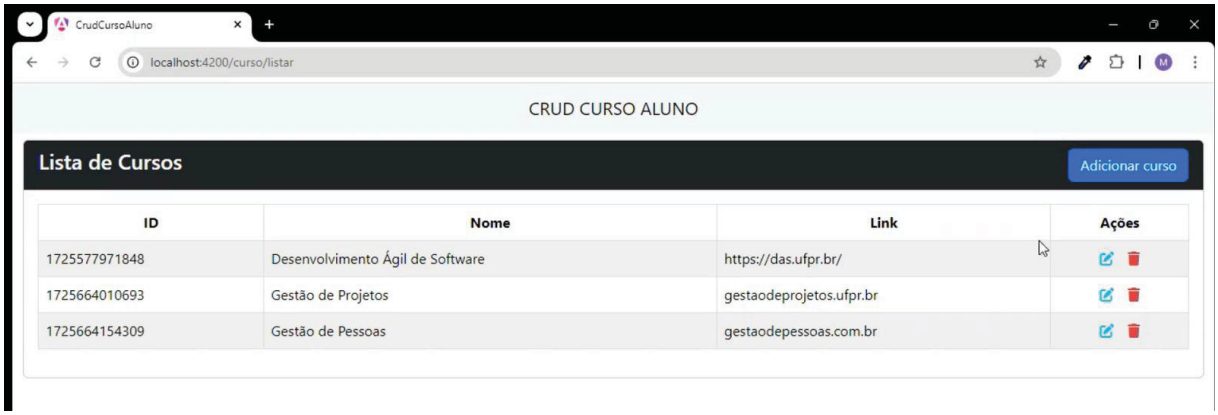
Fonte: O Autor(2025)

Figura 15 - Formulário CRUD Aluno



Fonte: O Autor (2025)

Figura 16 - Listagem de Aluno



Fonte: O Autor(2025)

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

Na disciplina de UX no Design de Software, aplicou-se um estudo voltado ao desenvolvimento de interfaces que priorizam usabilidade, acessibilidade, desempenho e estética, com foco na experiência do usuário. Para isso, utilizou-se como estudo de caso a iniciativa do aplicativo Direciona, que pode ser desenvolvido para *tablets* e *desktops*, com o objetivo de auxiliar líderes a identificar e acompanhar áreas de interesse relacionadas a projetos e demandas do Sistema Fiep. A plataforma proporciona uma experiência intuitiva e personalizada, permitindo que os usuários se mantenham informados e engajados.

O projeto envolveu a criação de pelo menos cinco telas, detalhando a função de cada uma e explicando as escolhas de design. Foram utilizadas fontes como *Segoe* (Microsoft, 2025), pela simplicidade e objetividade na leitura, e Montserrat (Google, 2025), na tela inicial e descritivos, conferindo elegância. As cores foram aplicadas estrategicamente: Ciano como cor primária para transmitir profissionalismo, laranja-amarelado e Verde-grama para destacar ações do usuário, e branco-acinzentado para indicar a tela atual.

Além do *design*, foi realizada a validação com pelo menos um usuário, explicando a função principal do produto, as opções de interação e navegação, e coletando *feedback* para melhorias. Essa prática reforça os princípios do desenvolvimento ágil, permitindo ajustes rápidos e contínuos na interface, melhorando a satisfação do usuário e a eficiência do sistema.

O estudo de UX se integrou diretamente com disciplinas como Desenvolvimento *Web*, que forneceu suporte técnico para a implementação das telas; e Introdução à Programação, que fundamentou a lógica de funcionamento da aplicação. Assim, a disciplina de UX no Design de *Software* foi essencial para consolidar habilidades de planejamento, colaboração e entrega iterativa, alinhadas às práticas ágeis de desenvolvimento de *software*.

9.1 ARTEFATOS DO PROJETO

Figura 17 - Feed aplicativo Direciona

Fonte: O Autor(2025)

Figura 18 - Nova solicitação aplicativo Direciona

Fonte: O Autor(2025)

Figura 19 - Formulário de nova solicitação

Fonte: O Autor(2025)

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

As disciplinas de Desenvolvimento Mobile 1 e 2 tiveram como foco o estudo e a aplicação dos fundamentos essenciais para a criação de aplicativos móveis, um segmento que atualmente representa uma grande demanda de soluções tecnológicas. O conteúdo abordado permitiu compreender o ecossistema *mobile* e as especificidades do desenvolvimento para dispositivos Android, desde a configuração do ambiente até a integração com serviços externos.

Na primeira e segunda parte da disciplina, foi apresentada uma visão geral sobre o ambiente Android (Android, 2025), destacando o uso do Android Studio como principal Ambiente de Desenvolvimento Integrado (IDE). Essa etapa introduziu conceitos importantes, como estrutura de projetos, componentização, ciclo de vida de atividades (*Activities*), manipulação de eventos, *views* e navegação entre telas.

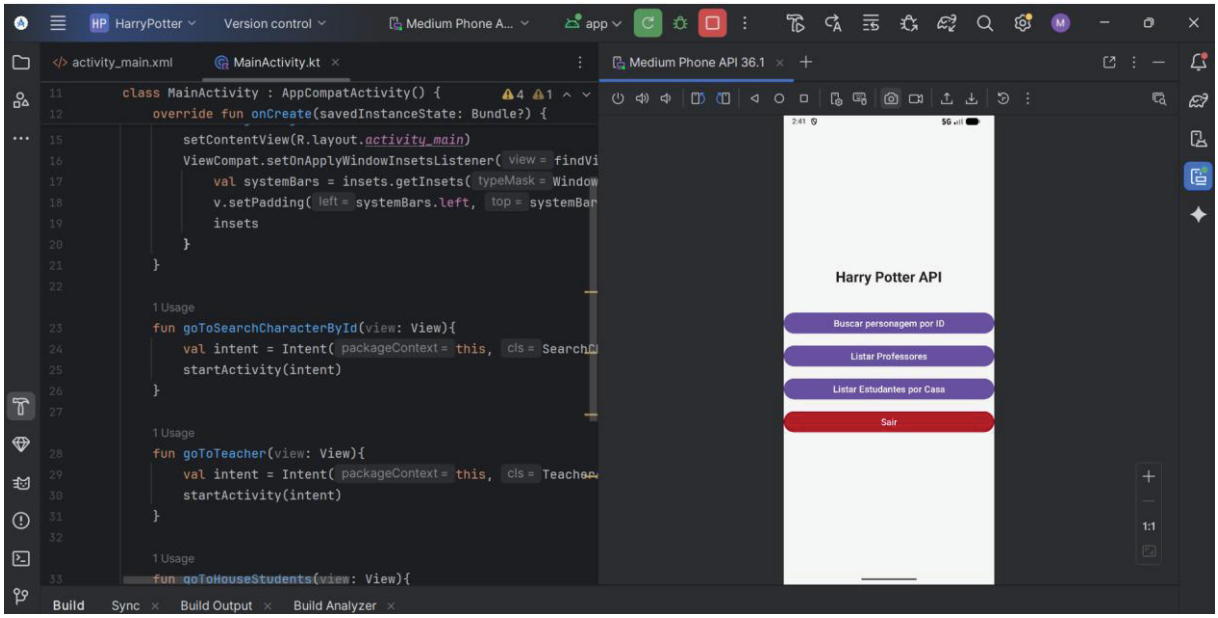
Como projeto prático, foi desenvolvido um aplicativo que consome dados da API pública HP-API (HP-API, 2025), voltada ao universo de Harry Potter. O objetivo foi aplicar conhecimentos sobre requisições HTTP, corrotinas em *Kotlin*, tratamento de respostas JSON e consumo de web services *RESTful* (Restful, 2025). O aplicativo apresenta uma tela principal (*dashboard*) que permite ao usuário executar diferentes ações: listar um personagem específico por ID, listar professores da escola, listar estudantes de uma casa e encerrar a aplicação. Essas funcionalidades demonstram a integração entre *front-end* e *back-end*, bem como a aplicação de práticas ágeis no desenvolvimento incremental do *software*.

Durante a execução do projeto, compreendeu-se, na prática, como os princípios da agilidade se aplicam ao desenvolvimento *mobile*: entregas incrementais de funcionalidades, *feedback* contínuo e melhoria constante.

A disciplina teve forte integração com outras áreas do curso, como Introdução à Programação, Banco de Dados, Desenvolvimento *Web* e Aspectos Ágeis de Programação. Essa integração se evidenciou na aplicação de conceitos de orientação a objetos (Rumbaugh; Jacobson; Booch, 2006), manipulação e consumo de dados, boas práticas de design de código e integração contínua (Red, 2025), demonstrando que o desenvolvimento mobile é uma extensão natural do ciclo ágil de *software*, consolidando a visão de que a agilidade também se manifesta no ambiente *mobile*, promovendo entregas rápidas, iterativas e de alto valor para o usuário.

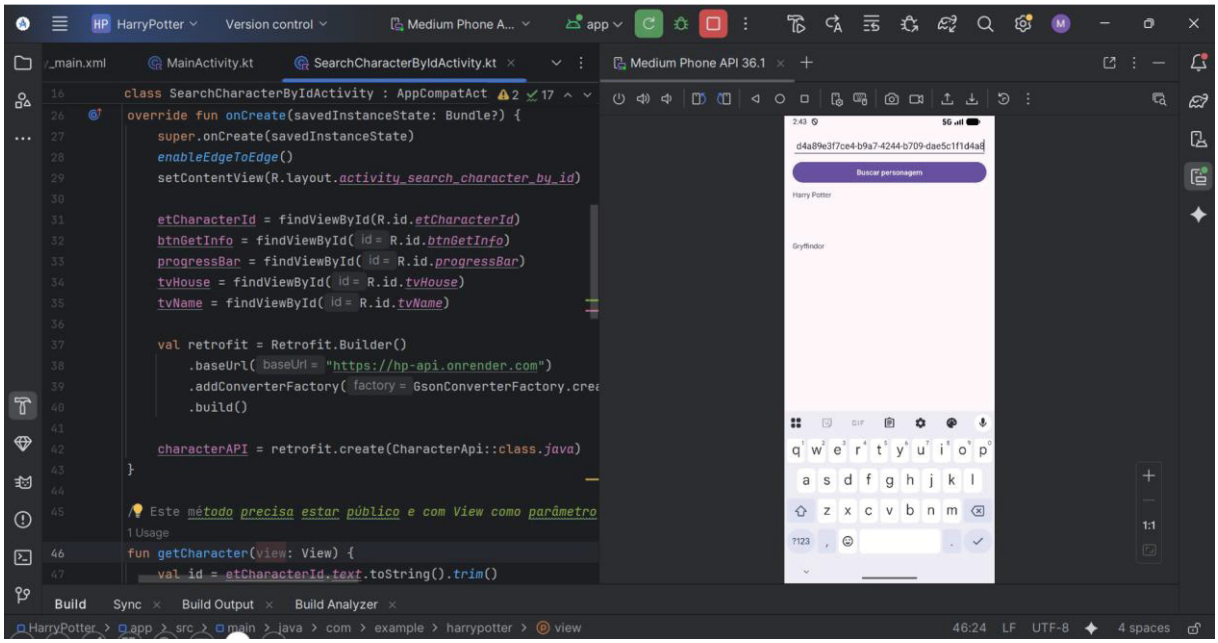
10.1 ARTEFATOS DO PROJETO

Figura 20 - Tela principal estudo de caso Mobile



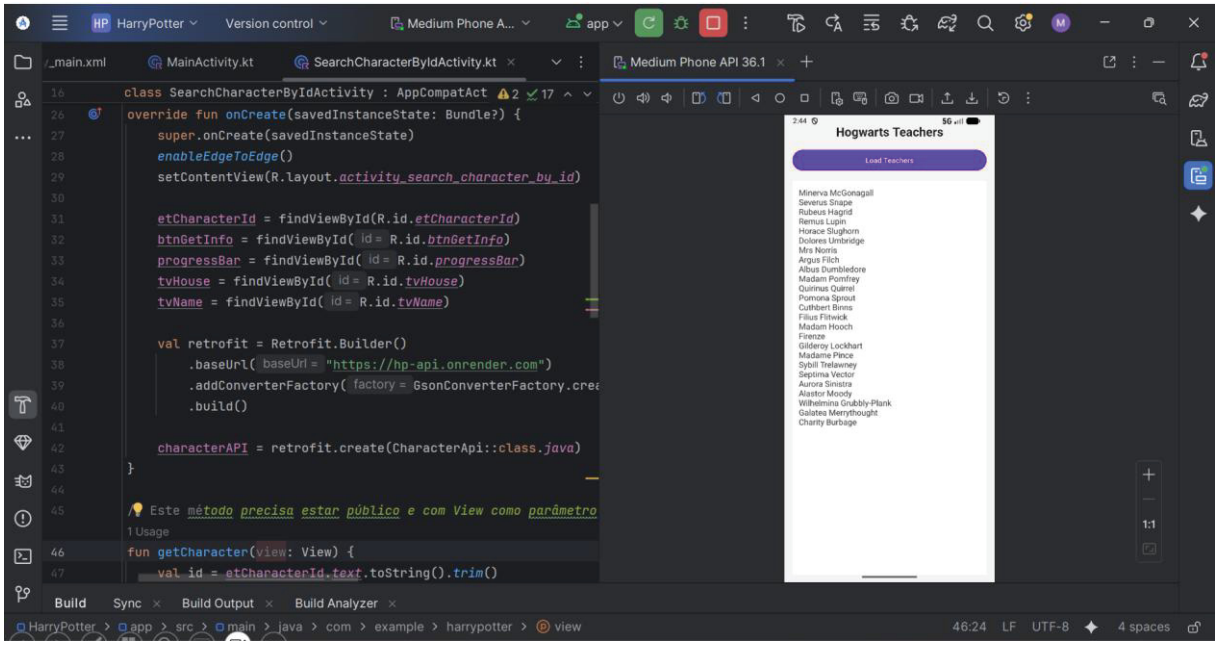
Fonte: O Autor(2025)

Figura 21 - Buscar personagem



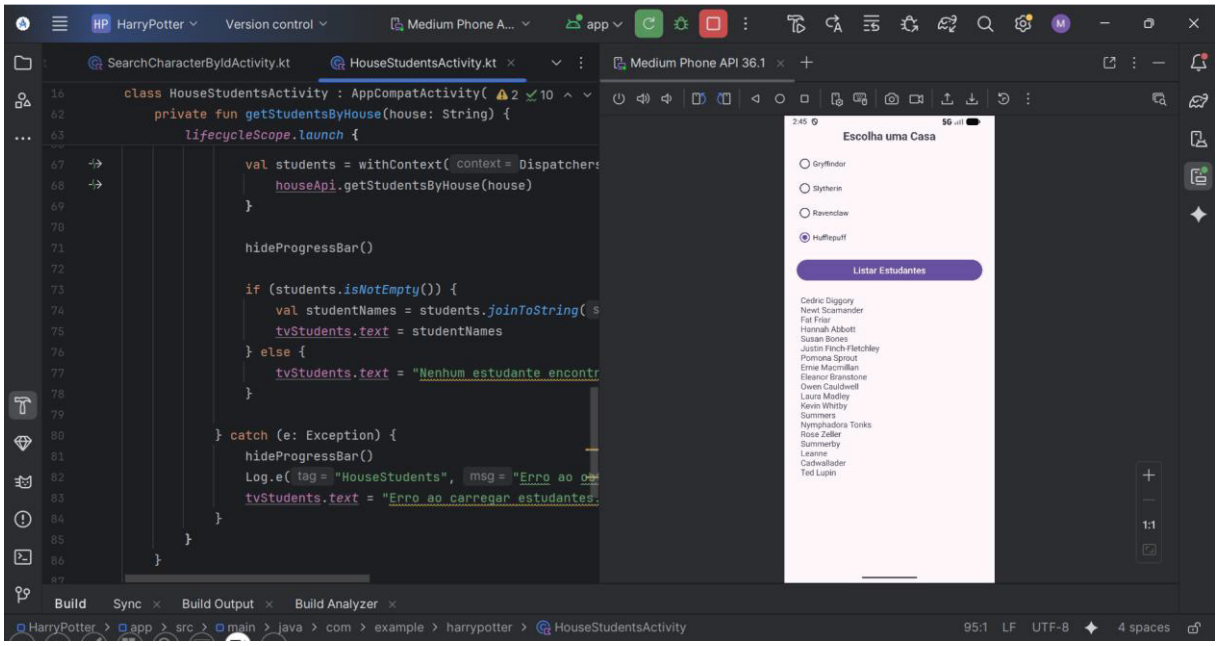
Fonte: O Autor(2025)

Figura 22 - Listar personagem



Fonte: O Autor(2025)

Figura 23 - Seleção casa para exibir personagens



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 (*DevOps*) se mostrou um pilar essencial para a compreensão de um ciclo de vida de software moderno e eficiente. O projeto prático realizado foi concebido para aplicar, de forma estrutural, os princípios e as práticas do movimento *DevOps*. A disciplina enfatiza o estudo e a aplicação de técnicas ágeis para o gerenciamento do ciclo de vida de software (Pressman; Maxim, 2021), cobrindo desde o controle de versões até a entrega e implantação de software.

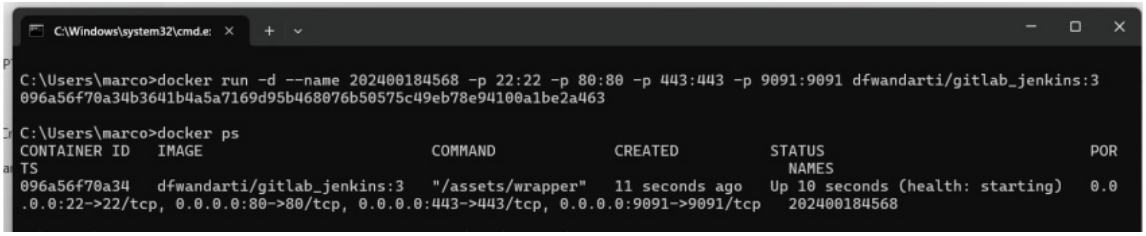
DevOps (Red, 2025) busca encurtar a distância entre as equipes de desenvolvimento e operações para entregar valor ao cliente de forma mais frequente e segura. Um dos contrastes mais significativos é a diferença de tempo: enquanto o ciclo de vida de software tradicional (Pressman; Maxim, 2021) pode levar meses ou até anos, o DevOps busca a entrega de software em dias.

A disciplina de Infraestrutura não atua isoladamente. Pelo contrário, ela se integra de forma coesa com outros temas do curso, principalmente nas disciplinas de desenvolvimento web, mobile e banco de dados. O uso do *Git* (Git, 2025) para controle de versão é a base para qualquer pipeline de Integração e Entrega Contínua (CI/CD) (Red, 2025). No trabalho, utilizou-se o Docker (Docker, 2025) para criar imagens imutáveis e padronizadas, garantindo que a aplicação funcionasse de maneira consistente em diferentes ambientes. A orquestração desses contêineres foi abordada com o uso do Kubernetes (Kubernetes, 2025), a ferramenta mais popular para gerenciar a implantação e a escala de aplicações.

Um ponto de aprendizado crucial, e que demonstra a importância da cultura de feedback, veio diretamente da análise do trabalho. O comentário de que *não é recomendável usar Git dentro do container* é uma lição prática sobre a aplicação correta das ferramentas de DevOps. Ele ressalta que, embora a tecnologia ofereça flexibilidade, a adesão a certas práticas é fundamental para garantir a segurança e a eficácia do sistema. Esse comentário serve para transformar descobertas em melhorias.

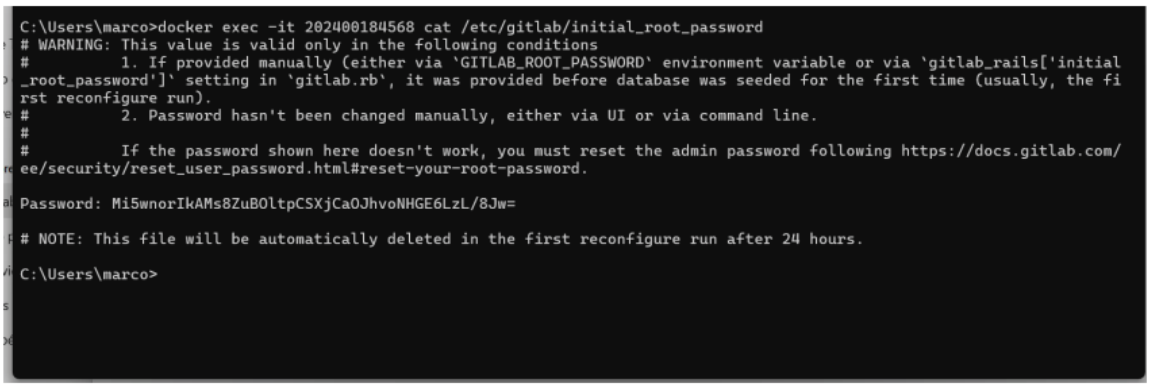
11.1 ARTEFATOS DO PROJETO

Figura 24 - Baixando e rodando o container



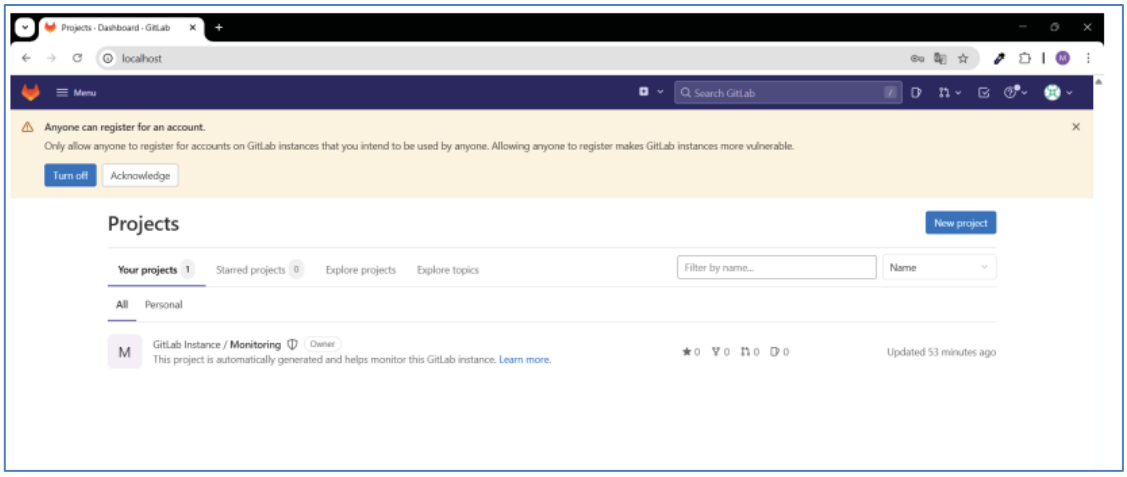
Fonte: O Autor(2025)

Figura 25 - Acessando senha do GitLab



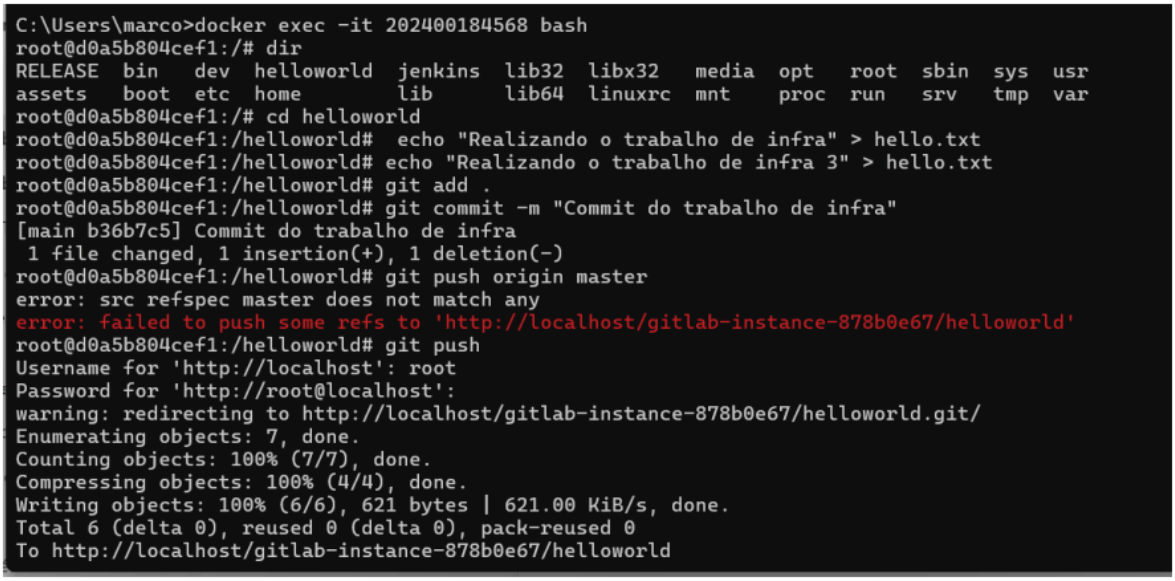
Fonte: O Autor(2025)

Figura 26 - Acessando o GitLab



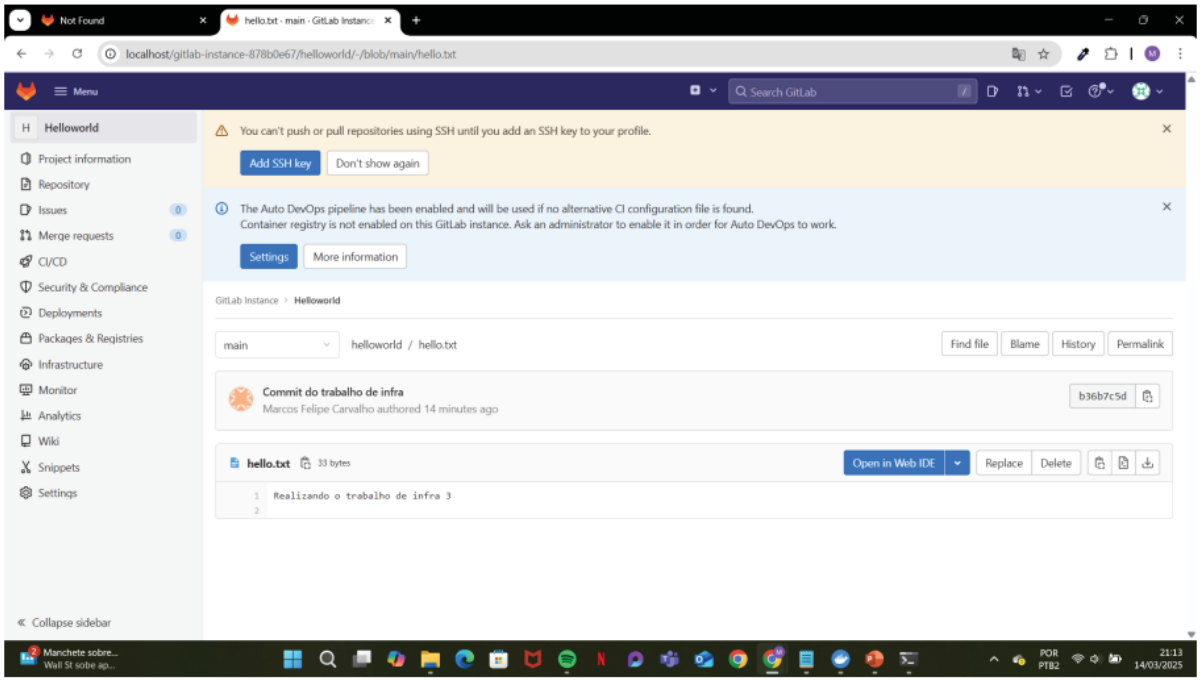
Fonte: O Autor(2025)

Figura 27 - Fazendo alterações no repositório



Fonte: O Autor(2025)

Figura 28 – Resultado



Fonte: O Autor(2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados apresentou os fundamentos e práticas essenciais para a garantia de qualidade em software, cobrindo conceitos de verificação e validação, tipos de teste (unitário, integração, sistema, aceitação e regressão), princípios de testabilidade, *mocks/stubs*, automação de testes e integração contínua (Pressman; Maxim, 2021). O conteúdo também explorou frameworks e ferramentas modernas para testes de interface e ponta a ponta, com destaque para o uso do Playwright (Playwright, 2025) e para a importância de estruturar suítes de teste de forma isolada e reprodutível.

O trabalho prático proposto consistiu em desenvolver um código que automatiza a criação de uma nota no serviço Anotepad (Anotepad, 2025). O requisito funcional é simples e permite exercitar conceitos centrais de automação: o script deve abrir o site, preencher o título com a *string* “Entrega trabalho TEST DAS 2024” e gravar no corpo da nota o nome(s) e matrícula(s) do(s) autor(es). Caso o trabalho seja em grupo, todos os nomes e matrículas devem constar no corpo da nota.

Do ponto de vista de testes automatizados, o projeto permitiu abordar e aplicar:

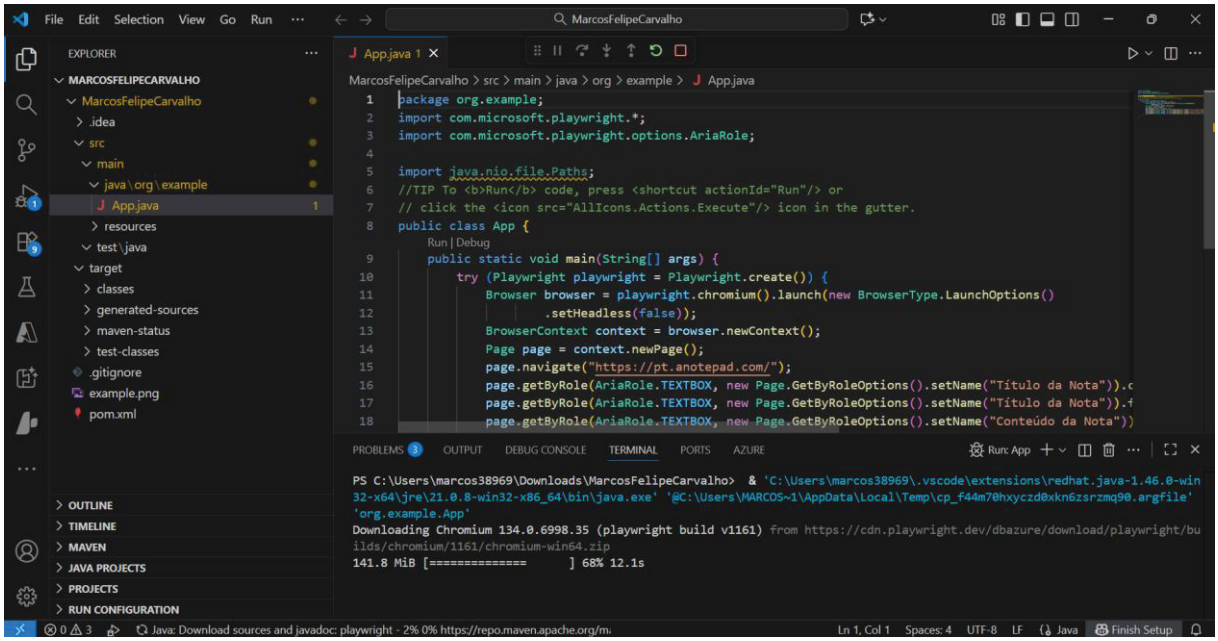
- Robustez de *scripts*: uso de esperas implícitas/automáticas e localizadores resilientes para evitar fragilidade frente a alterações na UI;
- Isolamento e repetibilidade: garantir que o teste possa ser executado várias vezes sem dependências externas (p.ex. limpar a nota anterior se necessário);
- Integração com CI: os *scripts* podem ser executados como parte de pipelines de CI para validar cenários críticos automaticamente a cada alteração.

A disciplina se integrou fortemente com outras áreas do curso: Aspectos Ágeis de Programação (práticas de código limpo e TDD) (Beck, 2003), Desenvolvimento Web (compreensão de DOM e APIs), e *DevOps*/CI (execução automatizada de suítes em pipelines) (Red, 2025). Essa integração reforça a ideia de *shift-left* (Sommerville, 2011), em que qualidade e testes são incorporados desde cedo no ciclo de desenvolvimento.

Conclusão: O trabalho demonstra que testes automatizados são ferramentas práticas para garantir entrega contínua de valor e reduzir risco introdutório de regressões.

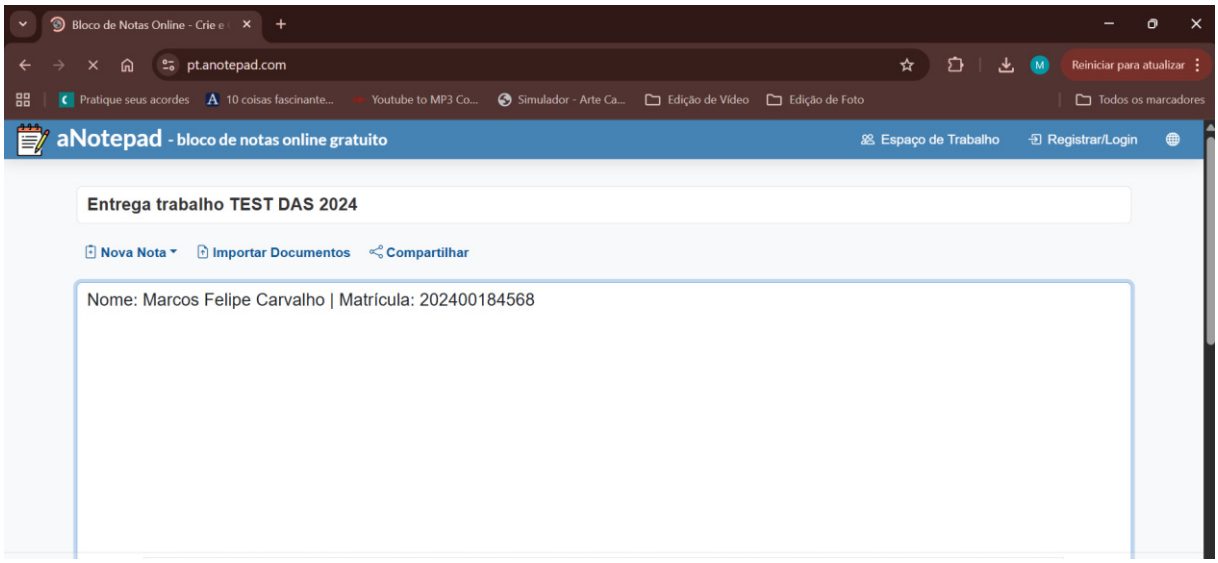
12.1 ARTEFATOS DO PROJETO

Figura 29 - Código Teste Automatizado



Fonte: O Autor(2025)

Figura 30 - Resultado do Teste



Fonte: O Autor(2025)

13 CONCLUSÃO

O presente memorial consolidou o percurso de aprendizado e prática desenvolvido ao longo do curso de Pós-Graduação em Desenvolvimento Ágil de Software. A integração entre as disciplinas permitiu compreender de forma ampla e aplicada o ciclo de vida do desenvolvimento de software, desde a modelagem e programação até os testes, a implantação e a experiência do usuário. Cada etapa estudada contribuiu para a formação de uma visão sistêmica, em que teoria e prática se complementam para sustentar a entrega contínua de valor ao cliente.

Ao longo do curso, foi possível observar que o desenvolvimento ágil não se limita a um conjunto de ferramentas ou metodologias, mas representa uma mudança de mentalidade, voltada à colaboração, adaptação e foco naquilo que realmente gera valor. As práticas de entregas incrementais, *feedback* contínuo e melhoria permanente mostraram-se essenciais para alcançar maior eficiência, qualidade e satisfação do usuário. As disciplinas de DevOps e Testes Automatizados reforçaram essa visão ao evidenciar como a automação e a integração contínua sustentam a agilidade e reduzem riscos operacionais. Já as disciplinas de UX, Web e Mobile destacaram a importância de conectar o desenvolvimento técnico à experiência do usuário final.

O memorial também evidenciou que o sucesso na adoção de métodos ágeis depende da maturidade das equipes, da cultura organizacional e da capacidade de adaptação frente às mudanças. Entre os principais desafios identificados estão: a resistência cultural em equipes acostumadas a modelos tradicionais; a dificuldade de alinhar a agilidade com estruturas hierárquicas rígidas; a falta de métricas claras para medir valor entregue; e a necessidade constante de capacitação para o uso de novas ferramentas e práticas. Superar esses obstáculos requer comprometimento coletivo, comunicação transparente e apoio da liderança.

Em síntese, o curso proporcionou o desenvolvimento de competências técnicas e comportamentais fundamentais para atuar em contextos dinâmicos e colaborativos. O desenvolvimento ágil, mais do que um método, se consolidou como um pilar estratégico para a entrega contínua de valor, a melhoria dos processos e a sustentabilidade das soluções tecnológicas. Esse aprendizado reflete a importância de unir pessoas, processos e tecnologia em torno de um propósito comum: entregar software de forma eficiente, adaptável e centrada no cliente.

REFERÊNCIAS

AGILE Alliance. **Pair Programming**. Disponível em: https://agilealliance.org/glossary/pair-programming/. Acesso em: 27 out. 2025.

ANDROID. **Android**. Disponível em: https://developer.android.com/. Acesso em: 27 out. 2025.

ANGULAR. **Angular**. Disponível em: https://angular.io/. Acesso em: 27 out. 2025.

ANOTEPAD. **Anotepad: bloco de notas**. Disponível em: https://pt.anotepad.com/. Acesso em: 28 out. 2025.

AZEVEDO, B. **Clean Code e SOLID na construção da aplicação Oratio: melhorando a manutenibilidade e escalabilidade**. 2023. Trabalho acadêmico (TCC) – Universidade Federal de Campina Grande, Campina Grande, 2023.

BECK, K. *et al*. **Manifesto for Agile Software Development**. 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 28 out. 2025.

BECK, K. **Test-Driven Development: by example**. Boston: Addison-Wesley, 2003.

BLACHA, M; R, James. **Modelagem e projetos baseados em objetos com UML2**. 2. ed. Rio de Janeiro: GEN LTC, 2006. 520 p.

CANDIDO, R *et al.* **Gerenciamento de projetos**. Curitiba: Aymará Educação, 2012. 120 p.

CHEN, P. **The Entity-Relationship Model—Toward a Unified View of Data**. ACM Transactions on Database Systems, v. 1, n. 1, p. 9–36, mar. 1976.

CUCUMBER. **Gherkin Reference**. Disponível em: https://cucumber.io/docs/gherkin/reference/. Acesso em: 28 out. 2025.

DOCKER. **Docker**. Disponível em: https://www.docker.com/. Acesso em: 27 out. 2025.

ELMASRI, R; NAVATHE, S. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Education do Brasil, 2016.

FREITAS, T. *et al*. **Sistema Puxado em Projetos: Controle do WIP, Alívio de Sobrecarga e Entrega Contínua de Valor. Interference: A Journal of Audio Culture**, v. 11, n. 2, p. 626-642, 2025.

GIT. **Git**. Disponível em: https://git-scm.com/. Acesso em: 27 out. 2025.

GOOGLE. **Google Fonts: Montserrat**. Disponível em: https://fonts.google.com/specimen/Montserrat. Acesso em: 28 out. 2025.

GOTARDO, R. **Linguagem de programação**. Rio de Janeiro: Seses, v. 60, 2015.

HP-API. **The Harry Potter API**. Disponível em: https://hp-api.onrender.com/. Acesso em: 28 out. 2025.

JUNIT. **JUnit 5 User Guide**. Disponível em: https://junit.org/junit5/docs/current/user-guide/. Acesso em: 28 out. 2025.

KANBAN Board Game. **Kanban Board Game**. Disponível em: https://kanbanboardgame.com/. Acesso em: 28 out. 2025.

KUBERNETES. **Kubernetes**. Disponível em: https://kubernetes.io/. Acesso em: 27 out. 2025.

MCCONNELL, S. **Complete Code: a practical handbook of software construction techniques**. 2. ed. Redmond: Microsoft Press, 2004. 914 p.

MICROSOFT. **Segoe UI font family**. Disponível em: https://learn.microsoft.com/en-us/typography/font-list/segoe-ui. Acesso em: 28 out. 2025.

MOB. **Mob Programming**. Disponível em: https://mobprogramming.org/. Acesso em: 27 out. 2025.

MOTA, F. **Introdução à UML**. 2015. Disponível em: https://fernandommota.github.io/academy/disciplines/2015/analise_projeto_software/files/04_introducaoUML.pdf. Acesso em: 27 out. 2025.

MYSQL. **MySQL**. Disponível em: https://www.mysql.com/. Acesso em: 27 out. 2025.

PLAYWRIGHT. **Playwright**. Disponível em: https://playwright.dev/. Acesso em: 27 out. 2025.

PMI. **Um guia do conhecimento em gerenciamento de projetos (Guia PMBOK®)**. 6. ed. Newtown Square: Project Management Institute, 2017.

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

RED Hat. **What is CI/CD?** Disponível em: https://www.redhat.com/en/topics/devops/what-is-ci-cd. Acesso em: 27 out. 2025.

RESTFUL. **RESTful API**. Disponível em: https://restfulapi.net/. Acesso em: 27 out. 2025.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. **Modelagem e projetos orientados a objetos com UML 2**. 2. ed. Rio de Janeiro: Elsevier, 2006.

SDLC. **Software Development Life Cycle**. Disponível em: https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/. Acesso em: 27 out. 2025.

SEGOE. **Segoe UI Font**. Disponível em: https://www.dafontfree.io/segoe-ui-font/. Acesso em: 27 out. 2025.

SGBD. **Sistemas Gerenciadores de Banco de Dados**. Disponível em: https://www.oracle.com/database/what-is-database/. Acesso em: 27 out. 2025.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Education do Brasil, 2011.

SPRING. **Spring Boot: Database Access**. Disponível em: https://spring.io/guides/gs/accessing-data-jpa/. Acesso em: 28 out. 2025.

TYPESCRIPT. **TypeScript**. Disponível em: https://www.typescriptlang.org/. Acesso em: 27 out. 2025.

W3SCHOOLS. **DDL, DML, DCL and TCL in MySQL**. Disponível em: https://www.w3schools.in/mysql/ddl-dml-dcl. Acesso em: 28 out. 2025.