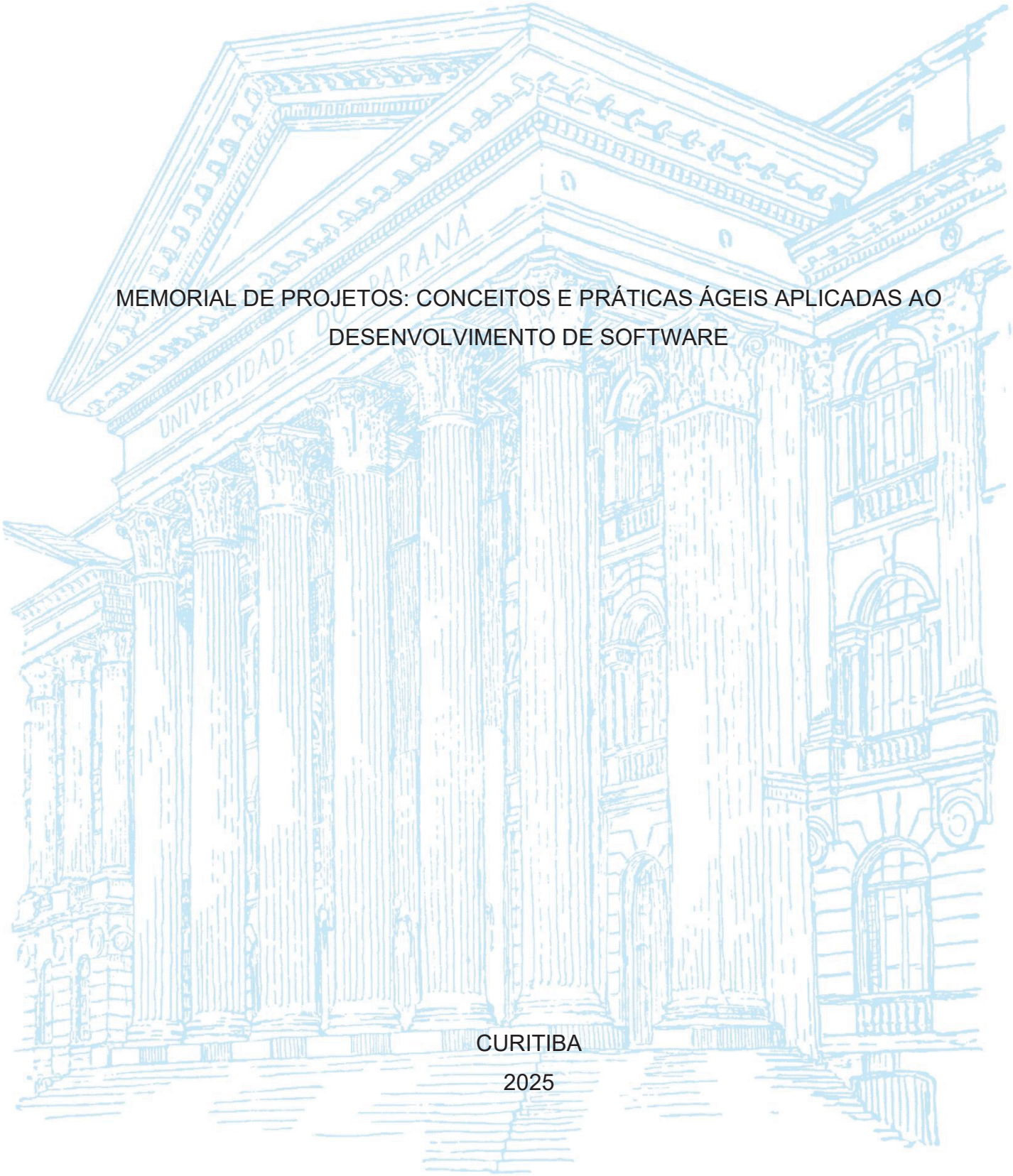
UNIVERSIDADE FEDERAL DO PARANÁ

JOSÉ PEDRO PALUDO



MEMORIAL DE PROJETOS: CONCEITOS E PRÁTICAS ÁGEIS APLICADAS AO DESENVOLVIMENTO DE SOFTWARE

CURITIBA

2025

JOSÉ PEDRO PALUDO

MEMORIAL DE PROJETOS: CONCEITOS E PRÁTICAS ÁGEIS APLICADAS AO DESENVOLVIMENTO DE SOFTWARE

Memorial de Projetos 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.

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

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 **JOSÉ PEDRO PALUDO**, intitulada: **MEMORIAL DE PROJETOS: CONCEITOS E PRÁTICAS ÁGEIS APLICADAS AO 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, 31 de Outubro de 2025.

RAFAELA MANTOVANI FONTANA
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

Este documento é constituído por um memorial de projetos desenvolvidos nas disciplinas do curso de Pós-Graduação em Desenvolvimento Ágil de Software. Nele, demonstramos como as disciplinas do curso se integram entre si e como são capazes de evidenciar, em seus projetos, a aplicação prática dos conceitos e princípios que fundamentam o desenvolvimento ágil de software. A proposta de cada disciplina é apresentada, os artefatos entregues são descritos e analisados, e os conceitos ágeis são relacionados de modo a evidenciar como são implementados nas diferentes etapas do processo de desenvolvimento. O memorial aborda tanto os aspectos teóricos quanto os práticos das metodologias ágeis, contemplando temas como modelagem ágil de software, gestão de sprints, elaboração de histórias de usuário, uso de quadros Kanban, Programação Orientada a Objetos, Desenvolvimento Orientado a Testes, práticas de *Clean Code* e integração e entrega contínuas. A integração entre as disciplinas permite compreender o ciclo completo de desenvolvimento ágil, desde a concepção e planejamento até a implementação, testes e entrega contínua do software. Assim, o trabalho reforça a importância de uma abordagem interdisciplinar e colaborativa, centrada na entrega de valor ao cliente e na adaptação constante às mudanças.

Palavras-chave: desenvolvimento de software; metodologias ágeis; gestão de projetos de software; Scrum; Kanban.

ABSTRACT

This document consists of a report on projects developed in the Agile Software Development Postgraduate program. It demonstrates how the course's disciplines integrate with each other and how they demonstrate, in their projects, the practical application of the concepts and principles underlying agile software development. The proposal for each discipline is presented, the delivered artifacts are described and analyzed, and agile concepts are related to demonstrate how they are implemented in the different stages of the development process. The report addresses both the theoretical and practical aspects of agile methodologies, covering topics such as agile software modeling, sprint management, user story development, Kanban board use, Object-Oriented Programming, Test-Driven Development, Clean Code practices, and continuous integration and delivery. The integration between disciplines allows for an understanding of the complete agile development cycle, from conception and planning to implementation, testing, and continuous software delivery. Thus, the work reinforces the importance of an interdisciplinary and collaborative approach, focused on delivering value to the customer and constantly adapting to changes.

Keywords: software development; agile methodologies; software project management; Scrum; Kanban.

SUMÁRIO

1 PARECER TÉCNICO

Considerada a “palavra da moda” por Jacobson (2002, apud Pressman, 2016, p.68), a agilidade deixou de ser um conceito de vanguarda para se consolidar como uma prática amplamente difundida na indústria de TI (Tecnologia da Informação), especialmente no desenvolvimento de software, destacando-se, sobretudo, pela capacidade de adaptar-se de forma eficaz às mudanças. Nesse sentido, o próprio Manifesto Ágil (Beck, 2001) salienta essa prioridade ao estabelecer, como seu quarto valor, a importância de responder a mudanças acima de seguir rigidamente um plano.

Agilidade é hoje um “movimento” (Pressman, 2016, p. 67), o qual surgiu de necessidades e desafios reais da indústria de evoluir e se adaptar. Os métodos ágeis surgem, portanto, como resposta às limitações dos métodos tradicionais de desenvolvimento de software. Esta mudança de paradigma não se restringe a questões técnicas de programação ou a gerenciamento de projetos, mas envolve uma transformação cultural no modo como equipes abordam o desenvolvimento e como se relacionam entre si e com clientes e stakeholders.

Com efeito, o uso de metodologias ágeis no desenvolvimento de software vai além da simples entrega de código funcional, mas promove, também, uma abordagem integrada que visa garantir a qualidade em todas as etapas do ciclo de vida do produto (Laporte; April, 2018).

O Curso de Pós-Graduação em Desenvolvimento Ágil de Software busca, portanto, capacitar profissionais para dominarem essas práticas e fazerem parte desse movimento que hoje, e cada vez mais, é padrão na indústria. Essa capacitação se dá de diversas formas através das disciplinas do curso.

O curso é composto por disciplinas conceituais, como MADS (Modelagem Ágil de Software), que trata de questões conceituais de metodologias ágeis, e GAP 1 e 2 (Gerenciamento Ágil de Projetos 1 e 2), que abordam esses conceitos na área de gestão de projetos; por disciplinas que aplicam estes conceitos de forma prática, como em MAG 1 e 2 (Modelagem de Software 1 e 2), que tratam da modelagem ágil de software e propõem ao aluno o desenvolvimento de modelos de software que aplicam os conceitos teóricos das disciplinas anteriores; por disciplinas que introduzem práticas de desenvolvimento ágil na codificação, como AAP (Aspectos Ágeis de Programação) e INTRO (Introdução à Programação); por disciplinas que avançam nesses conhecimentos ágeis práticos em aplicações modernas da área de

desenvolvimento, como WEB 1 e 2 (Desenvolvimento Web 1 e 2) e MOB 1 e 2 (Desenvolvimento Mobile 1 e 2); por disciplinas de caráter técnico que aplicam conceitos de modelagem de software e gestão ágil de projetos, como BD (Banco de Dados) e UX (UX no Desenvolvimento Ágil de Software); e por disciplinas que tratam de agilidade em áreas avançadas de desenvolvimento, como INFRA (Infraestrutura para Desenvolvimento e Implantação de software (DevOps)) e TEST (Testes Automatizados).

A seguir, demonstra-se como de fato estas disciplinas capacitam o aluno com os seus projetos, os quais exigem do aluno a aplicação prática dos conceitos ágeis de desenvolvimento de software de forma concreta.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

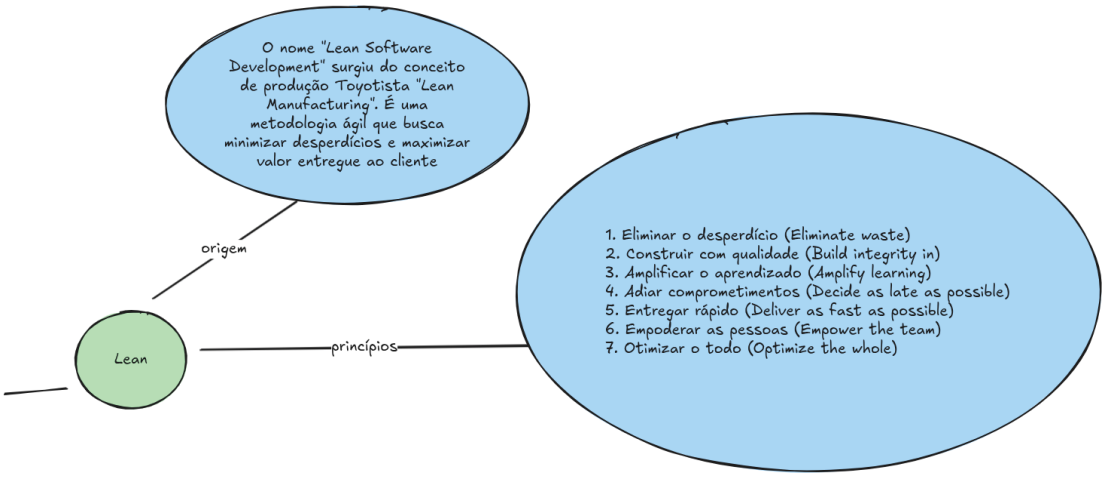
O projeto da disciplina MADS (Métodos Ágeis para Desenvolvimento de Software) consistiu no desenvolvimento de um mapa mental contendo os principais tópicos, conceitos e práticas que envolvem o tema desenvolvimento ágil. Dentre esses tópicos, estão: *Scrum* (descrito na FIGURA 3), *Kanban* (mostrado na FIGURA 2), *Extreme Programming* (ilustrado na FIGURA 4), *Lean Software Development* (apresentado na FIGURA 1), Entrega Contínua de Software, Processo de Software, entre outros.

A disciplina envolveu a comparação de metodologias clássicas com metodologias modernas, permitindo ao aluno ganhar mais conhecimento sobre a evolução dos métodos de desenvolvimento de software ao longo do tempo, contextualizando conceitos potencialmente abstratos. Assim, ao realizar o trabalho, o mapa mental deixa de ser apenas um ajuntamento de conceitos isolados, mas uma sintetização de tópicos já trabalhados.

Esta disciplina se mostra, portanto, como pilar do curso, pois trata dos conceitos fundamentais das metodologias ágeis que serão aprofundadas e aplicadas nas disciplinas subsequentes. Assim, o mapa mental confeccionado torna-se uma ferramenta visual de organização e memorização desses conceitos e servindo como um panorama geral das metodologias ágeis e de todo o curso.

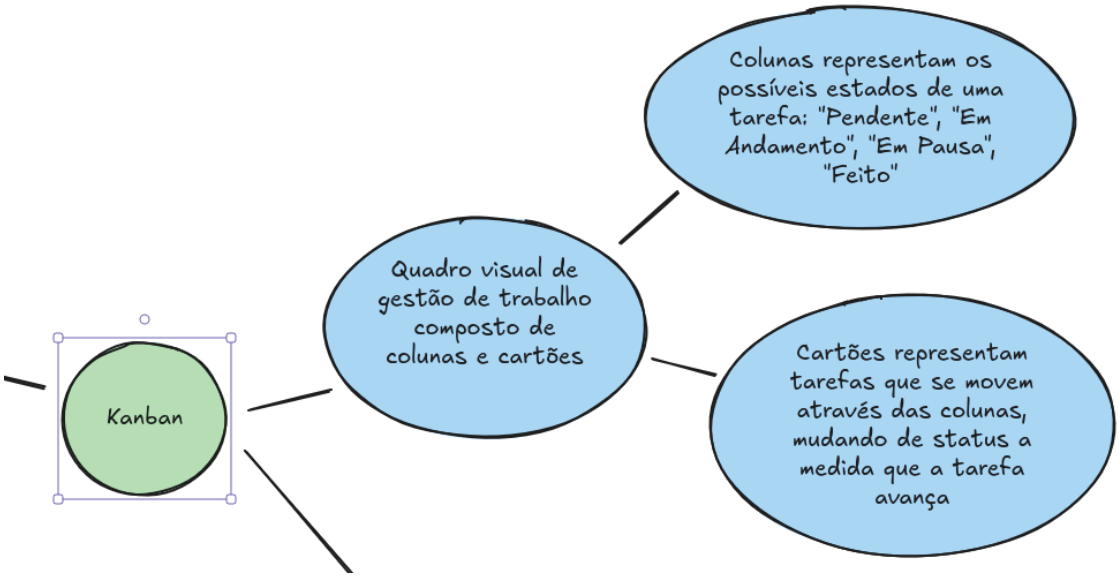
2.1 ARTEFATOS DO PROJETO

FIGURA 1 – TRECHO DO MAPA MENTAL SOBRE LEAN SOFTWARE DEVELOPMENT



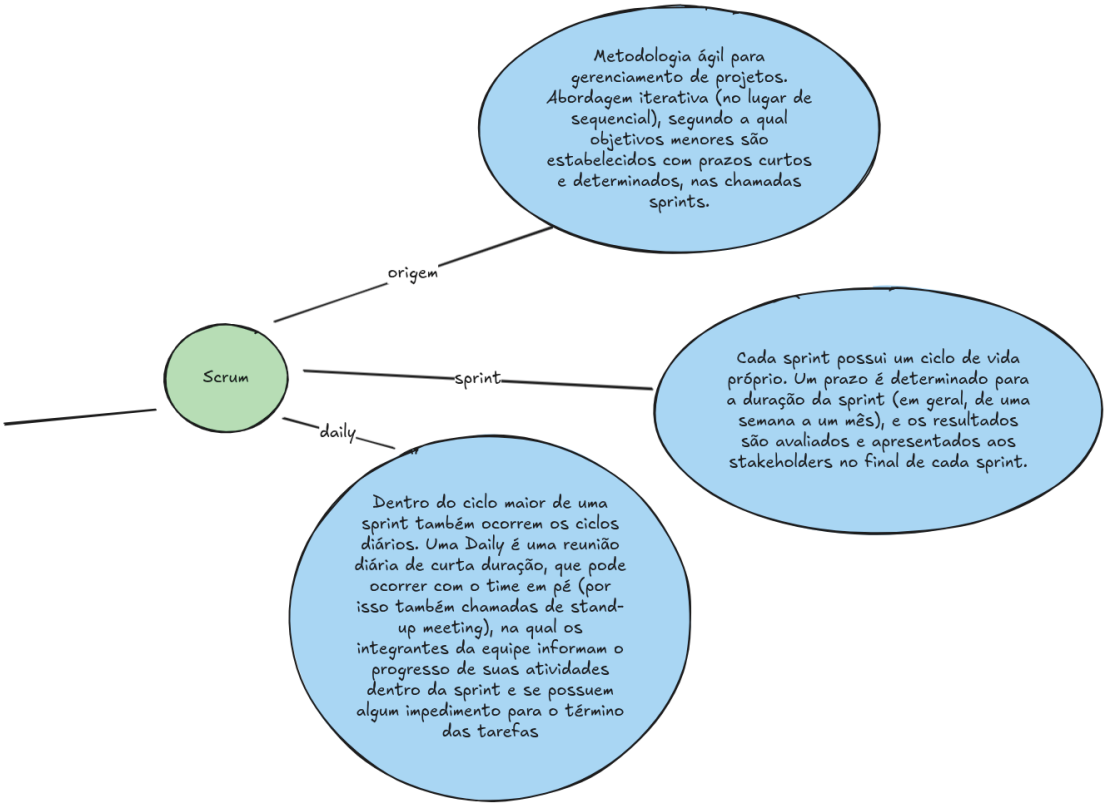
FONTE: O autor (2025)

FIGURA 2 – TRECHO DO MAPA MENTAL SOBRE KANBAN



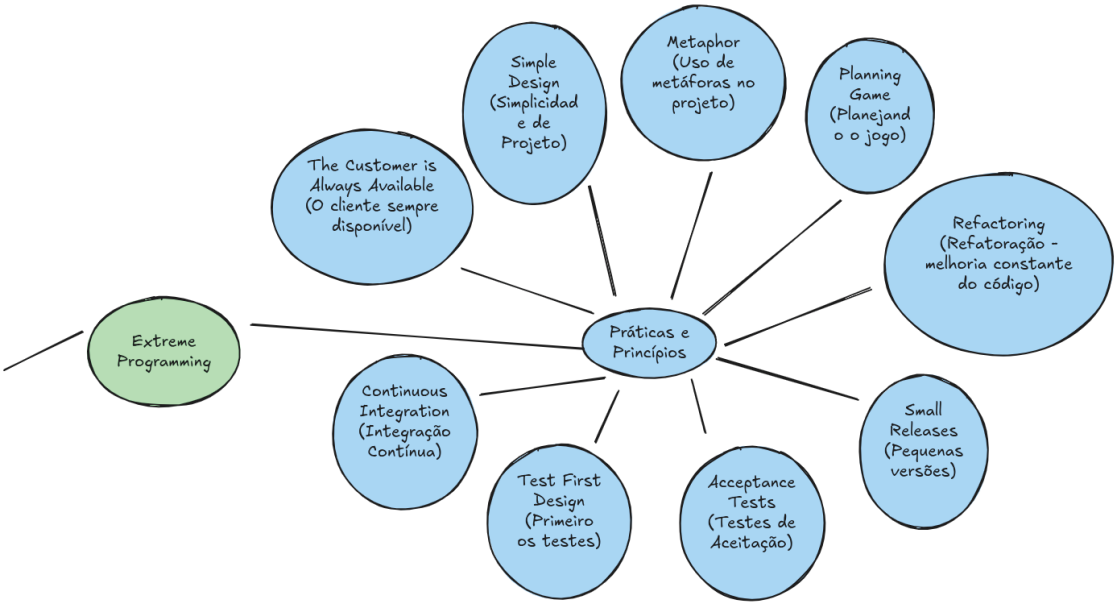
FONTE: O autor (2025)

FIGURA 3 – TRECHO DO MAPA MENTAL SOBRE SCRUM



FONTE: O autor (2025)

FIGURA 4 – TRECHO DO MAPA MENTAL SOBRE SCRUM



FONTE: O autor (2025)

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

O projeto da disciplina MAG 1 – Modelagem Ágil de Software - Visão Funcional, consistiu na modelagem de um sistema de gestão de condomínio, e dois artefatos deveriam ser entregues. O primeiro artefato seria composto por diagramas de caso de uso de nível 1 e 2, e o segundo, por histórias de usuário.

Para o primeiro artefato, o diagrama de nível 1 (representado na FIGURA 5) deveria ser mais alto nível, focado no negócio, nos casos de uso de modo geral e nos atores; enquanto o de nível 2 deveria ter mais ênfase no sistema, contendo os fluxos da aplicação através das telas. Para o segundo artefato, as histórias de usuário deveriam seguir o padrão “COMO... , QUERO... , PARA...”, mostrando a ênfase no usuário e no valor que a aplicação entrega a ele. A FIGURA 6 apresenta um exemplo de história de usuário.

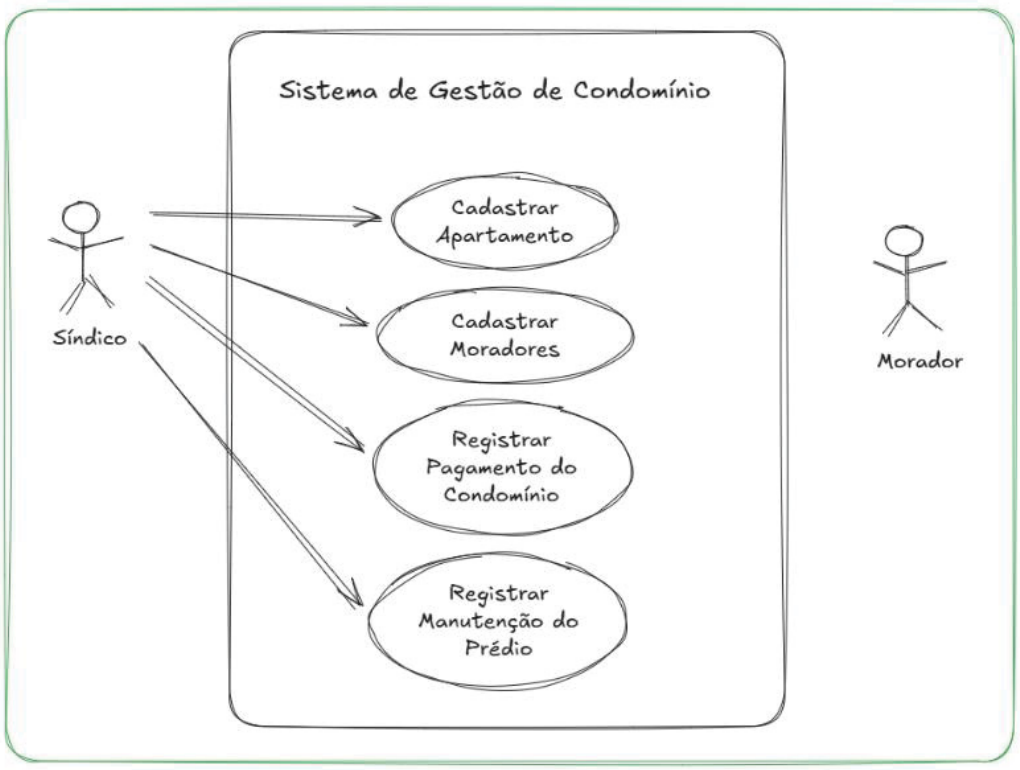
O projeto da disciplina MAG 2 – Modelagem Ágil de Software - Visão Estrutural, por sua vez, servia como uma extensão à primeira disciplina. Dois artefatos deveriam ser entregues, também tendo em vista o mesmo sistema de gestão de condomínio. O primeiro artefato seria composto pelo diagrama de classe, enquanto o segundo artefato seria composto pelo diagrama de sequência das histórias de usuário.

O diagrama de classe (ilustrado na FIGURA 7) deveria conter as tabelas do banco de dados, com as entidades do sistema (apartamento, morador, vaga, etc.). O diagrama de sequência (mostrado na FIGURA 8) deveria representar as histórias de usuário da primeira disciplina, mostrando de maneira visual como os atores interagem com o sistema, como o sistema interage com o banco de dados, e como as regras e condições do sistema são geridas.

Nesta disciplina, o aluno tem a oportunidade de aplicar os conceitos ágeis vistos na disciplina MADS, pois, enquanto em MADS foram apresentados de maneira teórica os conceitos de metodologias ágeis, em MAG 1 e 2 esses conceitos devem ser aplicados na confecção dos diferentes diagramas e das histórias de usuário.

3.1 ARTEFATOS DO PROJETO

FIGURA 5 – DIAGRAMA DE CASO DE USO DE NÍVEL 1 (MAG 1)



FONTE: O autor (2025)

FIGURA 6 – EXEMPLO DE HISTÓRIA DE USUÁRIO (MAG 1)

HU03 Registrar o Pagamento do Condomínio

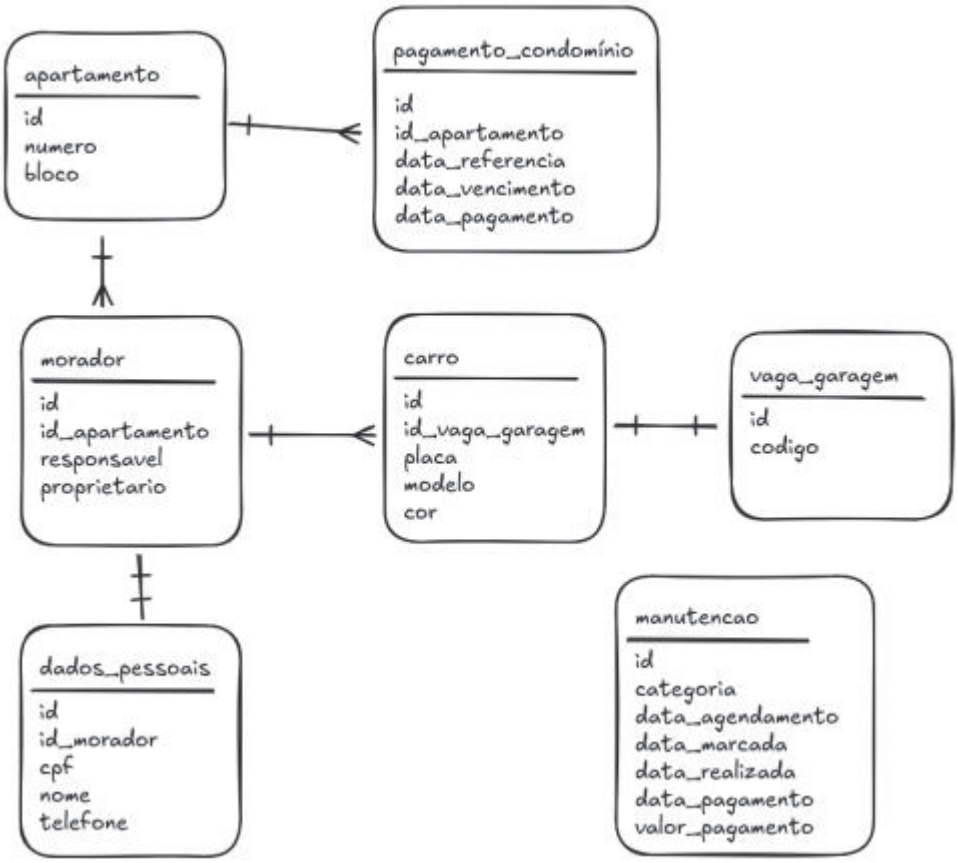
Como síndico. Quero registrar pagamentos do condomínio. Para manter as contas do condomínio em dia.

Critérios de Aceitação:

- Ao digitar o número do apartamento, mostrar o nome do responsável e o valor do condomínio.Ao digitar o número do apartamento, o sistema deve apresentar o nome do morador responsável e o valor do condomínio.
- Um pagamento só pode ser registrado se não houver valores anteriores vencidos.
- Registrar apartamento, Mês/Ano Referência, Data Vencimento, Data Pagamento, Valor Condomínio.

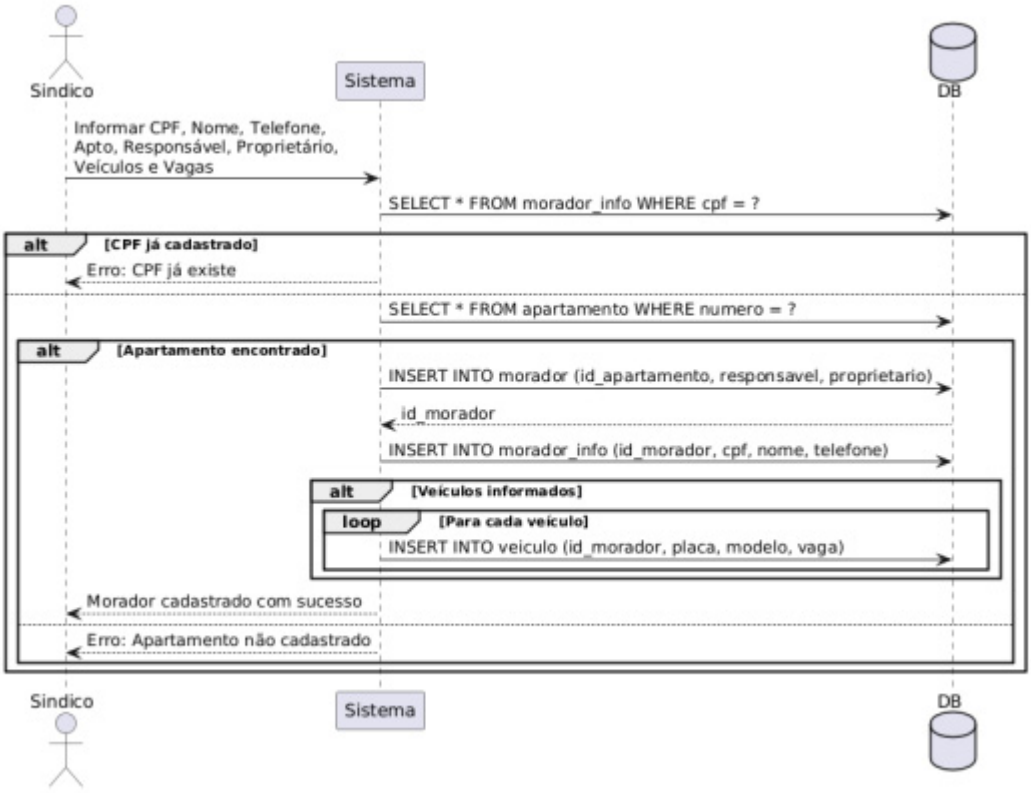
FONTE: O autor (2025)

FIGURA 7 – DIAGRAMA DE CLASSE (MAG 2)



FONTE: O autor (2025)

FIGURA 8 – EXEMPLO DE DIAGRAMA DE SEQUÊNCIA (MAG 2)



FONTE: O autor (2025)

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

O projeto da disciplina Gerenciamento Ágil de Projetos de Software 1 consistiu na elaboração de um plano de *release* para o desenvolvimento de um software, aplicando conceitos ágeis. Este plano de release consistia na escrita de *sprints*, com datas de início e fim, contendo histórias de usuário e cálculo de horas/ *story points*. A FIGURA 9 apresenta um exemplo de sprint, enquanto a FIGURA 10 demonstra o cálculo de *story points*.

Ao realizarmos um planejamento ágil, colocamos em prática tópicos estudados em MADS, e buscamos uma execução previsível, mas também flexível, que nos permita exercitar gestão de tempo, mensuração de esforço e priorização de tarefas. Dado que as sprints são construídas ao redor de histórias de usuário, elas também se mostram centradas no cliente, o que é peça fundamental do desenvolvimento ágil.

Para a realização do projeto, foi proposto o desenvolvimento de um software de gerenciamento de uma clínica médica, onde recepcionistas e médicos podem gerir pacientes de maneira mais rápida e fácil no seu cotidiano.

O projeto da disciplina Gerenciamento Ágil de Projetos de Software 2 envolveu o jogo *Kanban Board Game* (http://www.kanbanboardgame.com/). A FIGURA 11 mostra o jogo em andamento e a FIGURA 12 uma das telas do resultado após o final do jogo. Neste jogo, simulou-se o gerenciamento de uma equipe técnica para executar tarefas de desenvolvimento de software. O desafio do jogo envolve fazer com que as tarefas evoluam de maneira fluida, respeitando limites de WIP (*Work in Progress*), com priorização de tarefas e sem gargalos ou bloqueios.

4.1 ARTEFATOS DO PROJETO

FIGURA 9 – EXEMPLO DE SPRINTS COM SUAS HISTÓRIAS DE USUÁRIO DO PLANO DE RELEASE (PROJETO GAP1)

Plano de Release:

Cadastro de Pacientes e Médicos
Data Início: 28/07/25
Data Fim: 08/08/25
Cadastro de Pacientes *SENDO* Um recepcionista da clínica *QUERO* Cadastrar novos pacientes no sistema, informando nome completo, data de nascimento, CPF, telefone e endereço *PARA* Manter um registro completo dos pacientes da clínica *ESTIMATIVA* 3 pontos
Edição de Pacientes *SENDO* Um recepcionista da clínica *QUERO* Editar os dados de pacientes já cadastrados *PARA* Manter as informações dos pacientes atualizadas *ESTIMATIVA* 2 Ponto
Listagem de Pacientes *SENDO* Um recepcionista da clínica *QUERO* Visualizar uma lista de todos os pacientes cadastrados *PARA* Facilitar a busca e o gerenciamento dos pacientes *ESTIMATIVA* 1 Ponto

FONTE: O autor (2025)

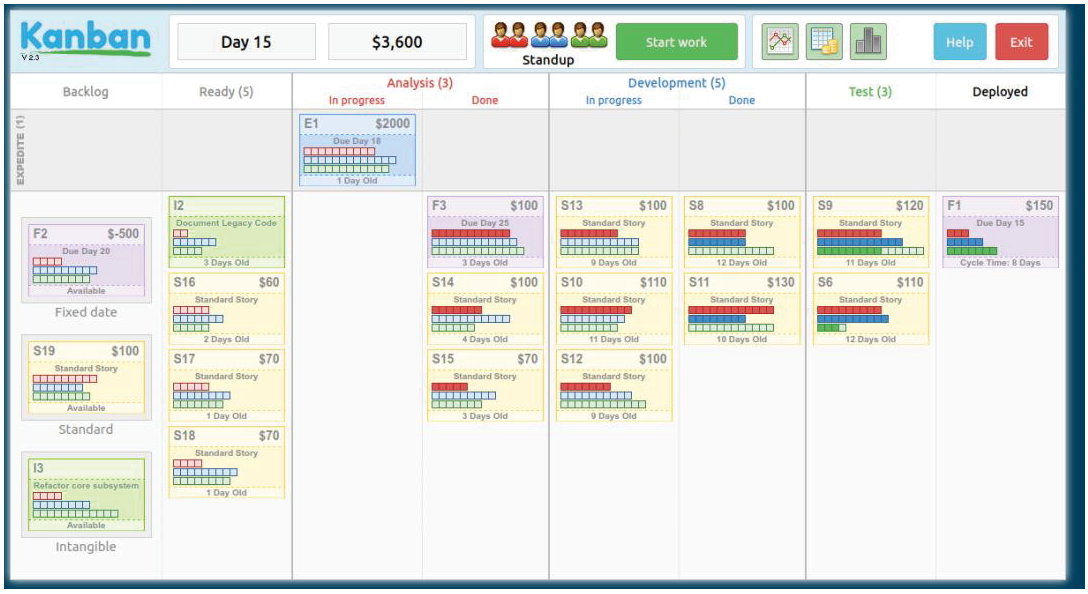
FIGURA 10 – CÁLCULO DA VELOCIDADE DAS SPRINTS (PROJETO GAP1)

Cálculo da Velocidade:

Horas disponíveis por dia:	8 horas por dia	Tamanho da Sprint:	2 semanas
Horas disponíveis por Sprint:	2 semanas * 5 dias por semana * 8 horas por dia = 80 horas por Sprint	Velocidade:	10 pontos por dia

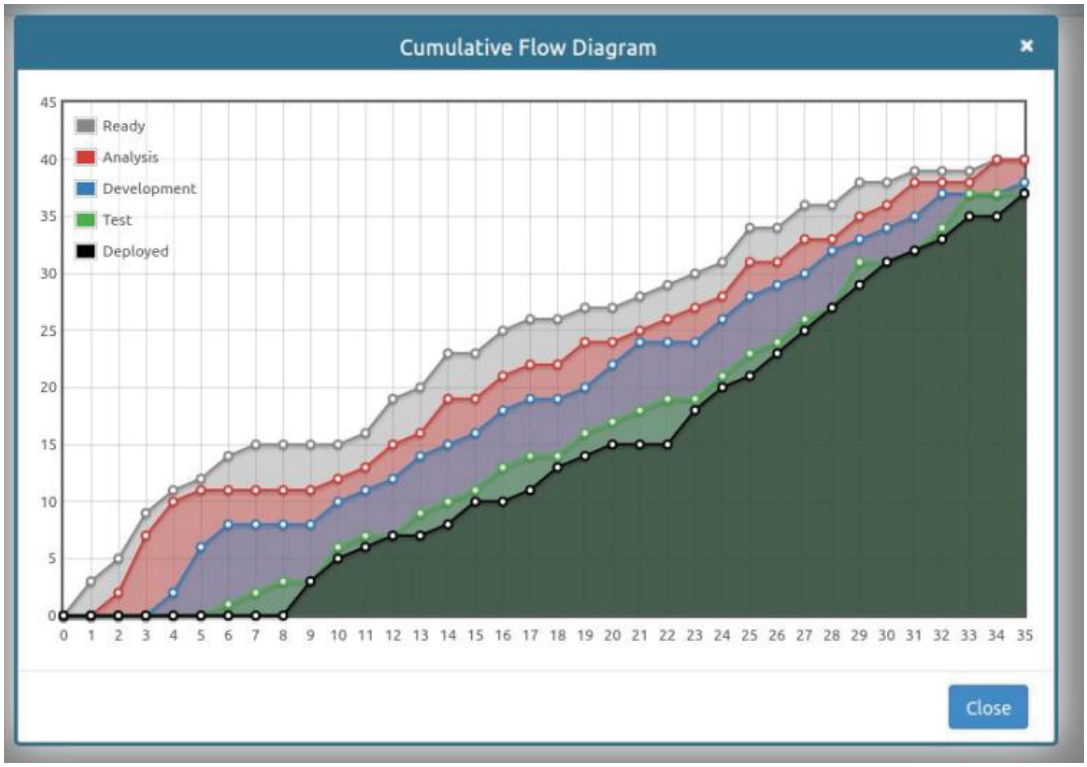
FONTE: O autor (2025)

FIGURA 11 – TELA DE UM JOGO EM ANDAMENTO DE KANBAN BOARDGAME (PROJETO GAP2)



FONTE: O autor (2025)

FIGURA 12 – TELA DO DIAGRAMA DE FLUXO ACUMULADO AO FINAL DE UM JOGO DE KANBAN BOARDGAME (PROJETO GAP2)



FONTE: O autor (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

O projeto da disciplina Into - Introdução à Programação envolvia o desenvolvimento de um *back-end* para um sistema bancário envolvendo o controle de contas de clientes, tanto de contas corrente quanto de contas de investimento. Como a disciplina deu ênfase em OOP (Programação Orientada a Objetos) e TDD (Desenvolvimento Orientado a Testes), o trabalho foi realizado em Java, utilizando de conceitos de OOP (como classes, interfaces e herança), tendo como objetivo final passar nos testes já previamente escritos.

O material inicial do projeto envolvia um projeto NetBeans para Java com 42 testes escritos, sendo o desafio do aluno implementar classes e métodos para fazer com que ao menos 40 dos testes passassem para atingir nota máxima. A FIGURA 14 demonstra o resultado dos testes que passaram com sucesso.

O projeto envolveu o desenvolvimento de classes como Cliente e Conta, com os métodos para verificar saldo, realizar saque, realizar depósito, etc. A FIGURA 13 mostra a declaração da classe abstrata Conta.

Durante o desenvolvimento do projeto, o aluno deveria executar os testes e vê-los passar à medida que ia desenvolvimento as funcionalidades, como é a prática do TDD, que tem como princípio primeiro o desenvolvimento de testes que testam lógica de negócio e depois o desenvolvimento das funcionalidades que sejam suficientes para fazer os testes passarem.

Esta disciplina se conecta diretamente, pois, à disciplina de AAP, por ambas tratarem de OOP (orientação a objeto) e qualidade de código, assim como à disciplina TEST (disciplina que trata de maneira aprofundada o tema de testes no desenvolvimento de software), pois o projeto de INTRO propõe a aplicação do conceito de TDD.

5.1 ARTEFATOS DO PROJETO

FIGURA 13 – CRIAÇÃO DA CLASSE ABSTRATA CONTA E SEU CONSTRUTOR, CONTENDO AS PROPRIEDADES ID, CLIENTE E SALDO

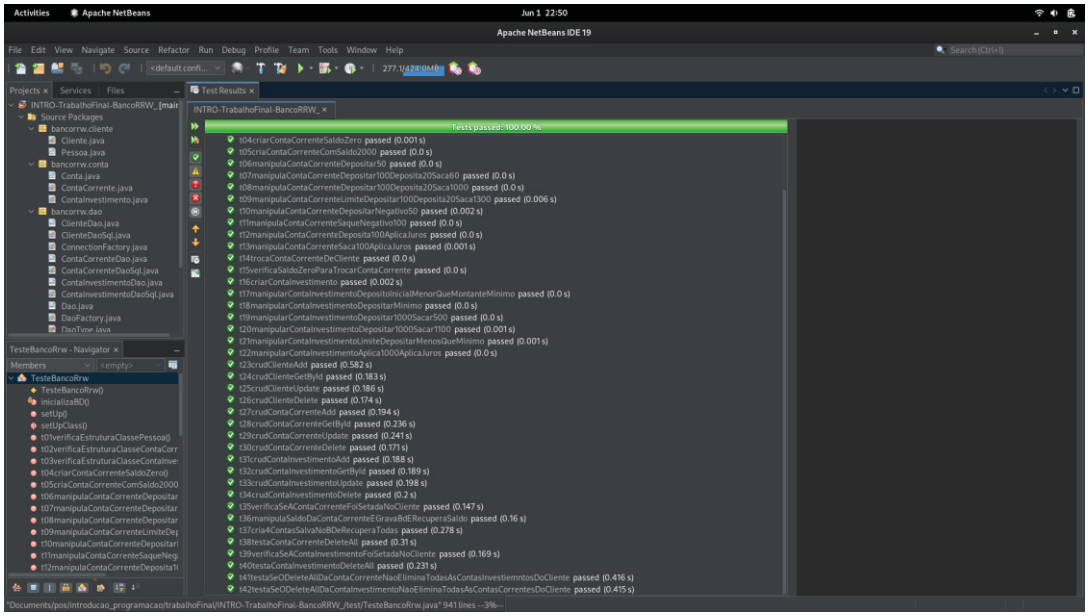
```java
public abstract class Conta {

    private long id;
    private Cliente cliente;
    private double saldo;

    public Conta(
        long id,
        Cliente cliente,
        double saldo
    ) {
        this.id = id;
        this.cliente = cliente;
        this.saldo = saldo <= 0 ? 0 : saldo;
    }
```

FONTE: O autor (2025)

FIGURA 14 – DEMONSTRAÇÃO DE QUE TODOS OS TESTES PASSARAM COM SUCESSO



FONTE: O autor (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

O projeto da disciplina BD – Banco de Dados foi composta de duas partes, uma de modelagem conceitual de um banco de dados e outra de implementação em SQL de uma modelagem.

A primeira questão teve caráter conceitual, propondo a modelagem de dados de um sistema de controle de biblioteca. O aluno deve, portanto, aplicar nesta modelagem os conceitos de modelagem ágil aprendidos na disciplina MAG.

Dois artefatos deveriam ser entregues nesta questão. O primeiro consistia em um Modelo Entidade-Relacionamento Conceitual (mostrado na FIGURA 15), contendo apenas as entidades e os relacionamentos do sistema da biblioteca. Já o segundo (apresentado na FIGURA 16) consistia em um Modelo Lógico, um Diagrama de Entidade Relacionamento, contendo tabelas, campos, chaves primárias, chaves estrangeiras e relacionamentos.

As modelagens deveriam abranger obras (livros, periódicos), usuários (que emprestam obras), funcionários (responsáveis por fazer os empréstimos), empréstimos e reservas.

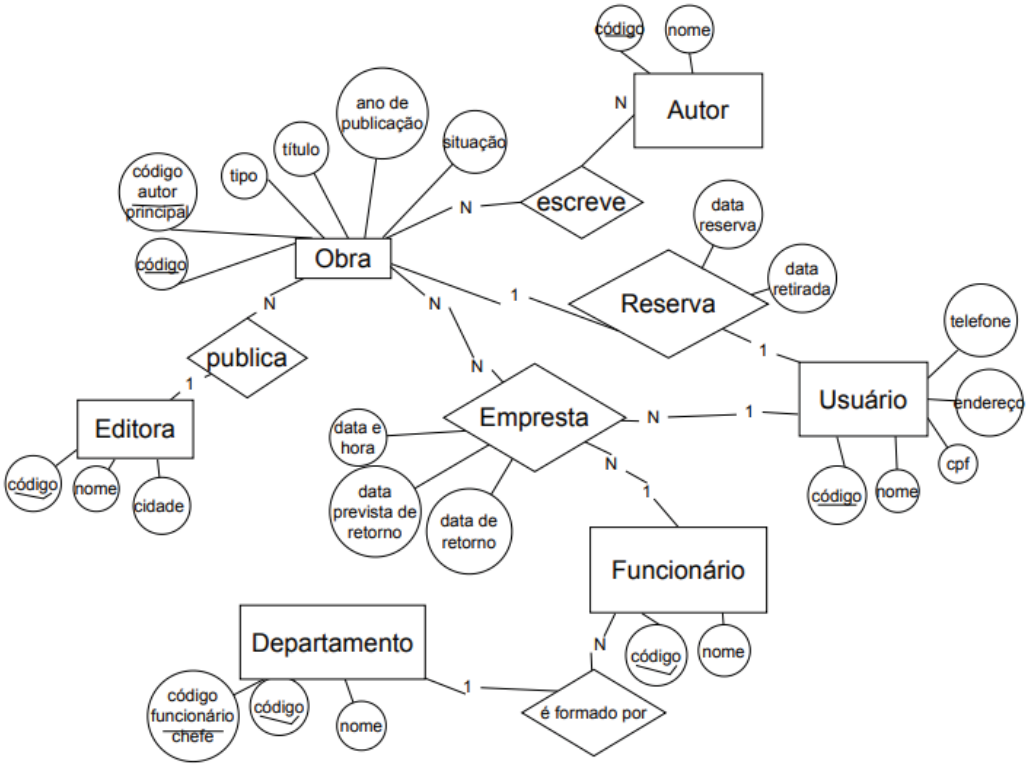
A segunda questão envolvia uma modelagem em linguagem SQL de um sistema da escolha do aluno. Dois artefatos principais deveriam ser entregues para esta questão.

O primeiro artefato seria o script SQL de criação das tabelas (exposto na FIGURA 17). O segundo artefato (mostrado na FIGURA 18) seria o script SQL de inserção de pelo menos 5 registros por tabela. Essas tabelas deveriam se relacionar entre si (com relacionamentos 1x1, 1xN e NxN) e as inserções deveriam evidenciar esses relacionamentos.

Para a realização da segunda questão, optamos por realizar a modelagem de um sistema de loja de roupas, com vendas, itens, promoções, categorias, clientes e funcionários.

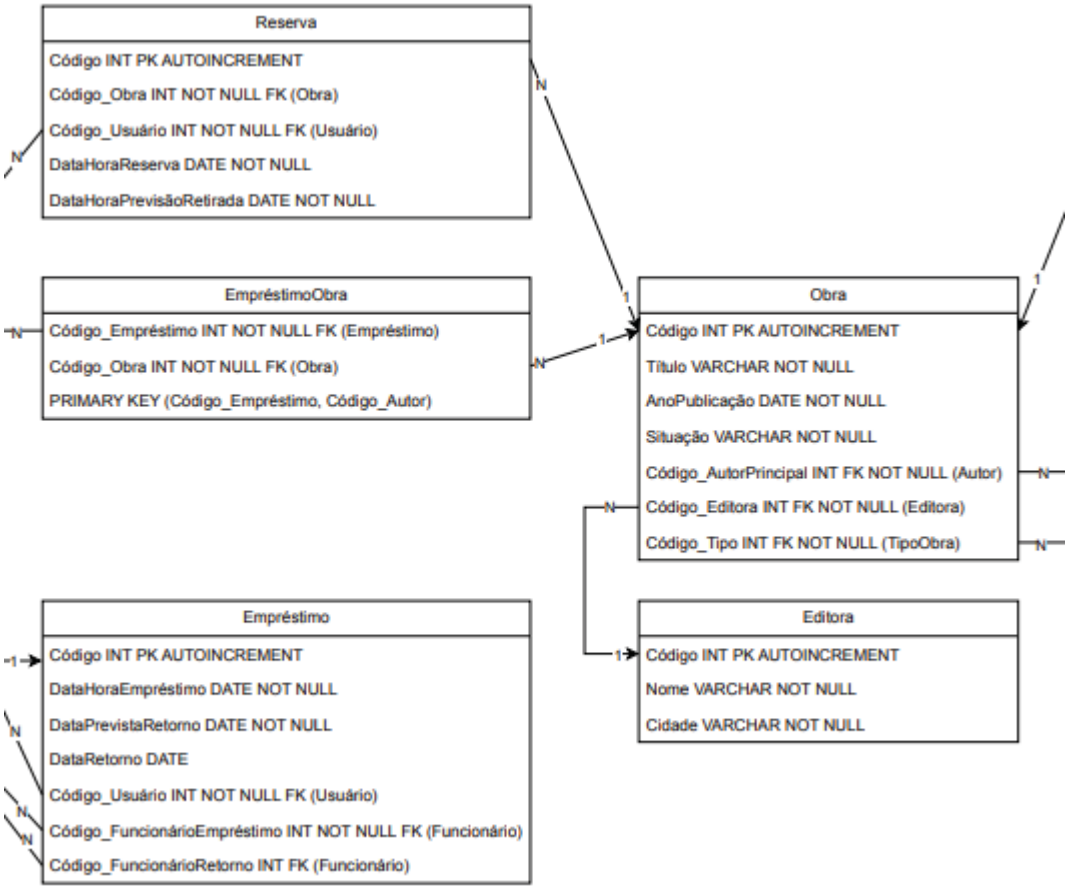
6.1 ARTEFATOS DO PROJETO

FIGURA 15 – MODELO ENTIDADE RELACIONAMENTO (QUESTÃO 1)



FONTE: O autor (2025)

FIGURA 16 – PARTE DO DIAGRAMA ENTIDADE RELACIONAMENTO



FONTE: O autor (2025)

FIGURA 17 – PARTE DO SCRIPT DE CRIAÇÃO DE TABELAS

```
CREATE TABLE sale (
    id SERIAL PRIMARY KEY,
    date TIMESTAMP NOT NULL,
    client_id INTEGER REFERENCES client(id),
    employee_id INTEGER NOT NULL REFERENCES employee(id),
    manager_id INTEGER NOT NULL REFERENCES employee(id)
);

CREATE TABLE item (
    id SERIAL PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    amount INTEGER NOT NULL
);

CREATE TABLE sale_item (
    sale_id INTEGER NOT NULL REFERENCES sale(id),
    item_id INTEGER NOT NULL REFERENCES item(id),
    unit_price DECIMAL(10, 2) NOT NULL,
    amount INTEGER NOT NULL,
    PRIMARY KEY (sale_id, item_id)
);
```

FONTE: O autor (2025)

FIGURA 18 – PARTE DO SCRIPT DE INSERÇÃO NAS TABELAS (QUESTÃO 2)

```
INSERT INTO employee_status
    ( id, name )
VALUES
    ( 1, 'Active' ),
    ( 2, 'On vacation' ),
    ( 3, 'On sick leave' ),
    ( 4, 'Former employee')
;

INSERT INTO employee_position
    ( id, name )
VALUES
    ( 1, 'Cashier ' ),
    ( 2, 'Assistant Manager' ),
    ( 3, 'Manager' )
;

INSERT INTO employee
    ( id, name, exit_date, position_id, status_id )
VALUES
    ( 1, 'João da Silva', NULL, 1, 1 ),
    ( 2, 'Maria Oliveira', NULL, 2, 1 ),
    ( 3, 'Joana Casagrande', NULL, 3, 1 ),
    ( 4, 'Luis Lopes', NULL, 1, 2 ),
    ( 5, 'Andfe Rocha', NULL, 2, 3 ),
    ( 6, 'Ana Ferreira', '2024-06-10', 1, 4 ),
```

FONTE: O autor (2025)

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

O projeto da disciplina AAP – Aspectos Ágeis de Programação consistiu em um desafio de aplicar conceitos de *Clean Code* em um código Java que implementa o algoritmo Bubble Sort.

Clean Code fortalece as práticas e princípios ágeis por priorizar e garantir adaptabilidade, manutenibilidade, legibilidade, etc. Dominar práticas de *Clean Code* é uma forma prática e concreta de aplicar conceitos de agilidade no desenvolvimento de software no dia a dia do desenvolvedor, evidenciando que a agilidade não está apenas no nível de projeto e modelagem, mas também no de codificação.

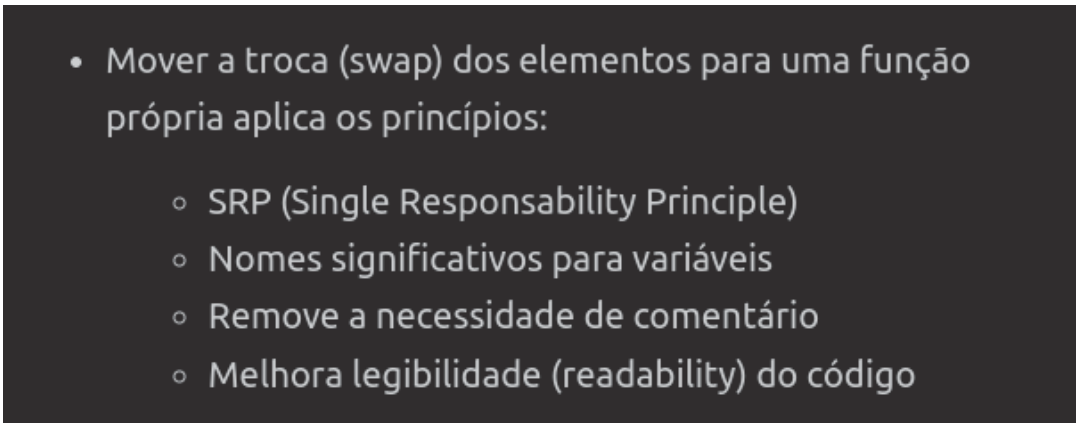
O código base fornecido (apresentado na FIGURA 20) da implementação vai de encontro às práticas de *Clean Code* ao abusar de comentários, declarar variáveis muito antes de utilizá-las, não usar nomes expressivos e significativos, agrupa muitas funcionalidades na mesma classe e nos mesmos métodos, confundir métodos públicos e privados e escrever métodos muito extensos.

O objetivo do trabalho seria, portanto, aplicar a boa prática da refatoração, ou seja, de aprimorar um código já existente, tornando-o mais legível, de mais fácil compreensão, mais adequado a padrões e mais fácil de se manter e estender. A FIGURA 21 demonstra o código refatorado. Além disso, seria preciso também entregar um texto explicativo (mostrado na FIGURA 19) explicando e justificando as mudanças realizadas no código.

Esta disciplina está, pois, intimamente relacionada com a disciplina INTRO, pois ambas tratam de práticas de OOP (Programação Orientada a Objetos), de *Clean Code*, qualidade de código e refatoração.

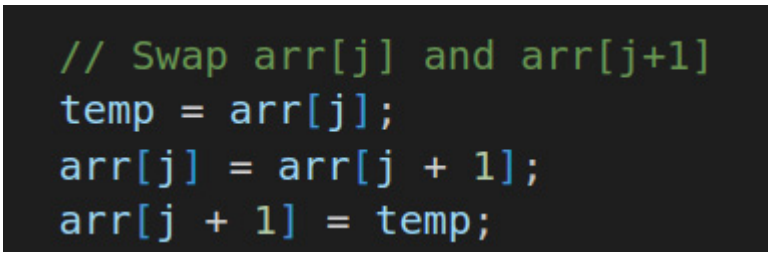
7.1 ARTEFATOS DO PROJETO

FIGURA 19 – TRECHO DO TEXTO EXPLICATIVO



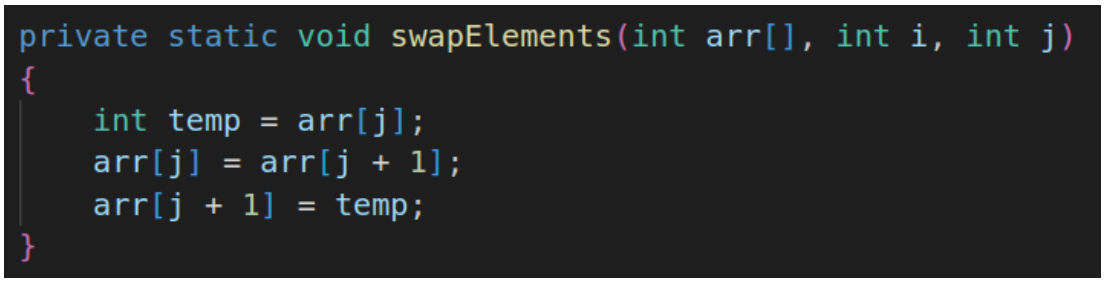
FONTE: O autor (2025)

FIGURA 20 – TRECHO DO CÓDIGO ORIGINAL ONDE É FEITA A TROCA DE ELEMENTOS NO ARRAY



FONTE: O autor (2025)

FIGURA 21 – TRECHO DO CÓDIGO REFATORADO ONDE É FEITA A TROCA DE ELEMENTOS NO ARRAY



FONTE: O autor (2025)

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

As disciplinas WEB 1 (Desenvolvimento Web 1) e WEB 2 (Desenvolvimento Web 2) abordar o desenvolvimento de aplicações web. Aplicações web são aplicações acessadas a partir de um navegador web (como Chrome, Firefox e Safari), as quais interagem com algum servidor web. Em aplicações clássicas, o usuário da aplicação deveria instalar um pedaço de software diretamente em sua máquina, e então interagir com ele. Em aplicações web, o único software que deve ser previamente instalado na máquina do usuário é o próprio navegador, sendo que todas as funcionalidades de todos os sites acessados decorrem, pois, de pedaços de software que são recebidos pelo navegador de algum servidor, sem a necessidade de instalação. Isso faz de um navegador web uma “aplicação de aplicações”, pois virtualmente qualquer software pode ser desenvolvido para navegadores web: software de banco, jogos, sites de notícias, redes sociais, software de gerenciamento de cursos (como é o caso do desafio da disciplina WEB 1), dentre outros.

O desenvolvimento web é composto de duas partes: *front-end* e *back-end*. O *front-end* refere-se ao pedaço de código que é executado pelo navegador, com o qual o usuário interage diretamente. O *back-end* refere-se ao pedaço de código que é executado pelo servidor, com o qual o usuário pode interagir diretamente, através de APIs, ou indiretamente, através de seu navegador (isto é, através do *front-end*). Aplicações compostas por *front-end* e *back-end* são chamadas de aplicações *full-stack*.

O projeto da disciplina WEB 1 consistiu no desenvolvimento do *front-end* de uma aplicação web. A aplicação tem como tema o gerenciamento de cursos e alunos. Nele, os usuários devem ser capazes de realizar operações *CRUD* (Acrônimo em inglês para “Create, Read, Update, Delete”, significando “Criar, Ler, Atualizar, Deletar”) com as entidades “Aluno” e “Curso”. Como não era preciso desenvolver um *back-end* para esta aplicação, o aluno deveria usar o armazenamento de dados do próprio navegador.

O projeto da disciplina WEB 2 consistiu no desenvolvimento de uma aplicação web *full-stack.* A aplicação tem como tema o gerenciamento de cursos, alunos e matrículas, servindo como uma continuação da aplicação *front-end* desenvolvida em WEB 1. Desta vez, contudo, também um *back-end* deveria ser desenvolvido. Este

back-end deveria se comunicar com um banco de dados, sendo necessário ao aluno, portanto, conhecimentos de modelagem aprendidos na disciplina MAG (1 e 2) e BD.

Tendo em vista que o trabalho destas disciplinas era opcional, não serão apresentados artefatos do projeto.

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

O projeto da disciplina UX – UX no Desenvolvimento Ágil de Software consistiu na prototipação de um site ou app, consistindo em quatro atividades próprias de um designer. Propomos um site chamado “Mamamigo”, que serviria para mães registrarem eventos e cuidarem da rotina de seus filhos.

A primeira tarefa consistia na explicação do produto e sua função, junto de uma justificativa para seu desenvolvimento. A FIGURA 22 apresenta esta explicação. Nesta parte tornou-se explícita a proposta de valor centrada no usuário, aspecto fundamental das metodologias ágeis, como apresentado na disciplina GAP.

A segunda tarefa consistia no desenvolvimento de cinco telas deste site. Uma destas telas é mostrada na FIGURA 23. Nesta parte o objetivo seria de construir um protótipo de interface, buscando aplicar conceitos da disciplina, desenvolvendo telas coesas e coerentes com o tema proposto.

A terceira tarefa consistia na explicação das escolhas feitas no desenvolvimento das telas. Nesta parte, justificou-se a escolha de cores, fontes e layout e explicou-se como as escolhas se alinhavam com a proposta do site. A FIGURA 24 exibe esta justificação.

A quarta tarefa consistia na apresentação do site para uma outra pessoa, que deveria responder perguntas. A FIGURA 25 apresenta os comentários da entrevistada. Esta tarefa é um exemplo de método de pesquisa de usuário e com ela ganhou-se boas respostas que indicavam o que funcionou e o que não funcionou no protótipo. Esta tarefa reforça o compromisso de metodologias ágeis com o cliente e o foco em iteratividade e ajustes constantes do produto com base em *feedbacks* reais.

9.1 ARTEFATOS DO PROJETO

FIGURA 22 – EXPLICAÇÃO DO SITE PARA A QUESTÃO 1

Mamamigo é um site para mães gerenciarem os cuidados com seus bebês em suas rotinas.
A plataforma oferece uma solução centralizada para que as mães cuidem de seus filhos de uma maneira mais simples, sem que precisem se preocupar com agendas, cadernos e diversos aplicativos.
Nele, as mães podem:

- Registrar as refeições, as sonecas e banhos de seus filhos, podendo monitorar a evolução da saúde e dos hábitos do bebê ao passar do tempo.
- Acompanhar agendamentos de consultas, exames e vacinas - assim como aniversários e eventos em geral.
- Criar Mimo Murais, murais virtuais para realizar colagens de fotos de momentos importantes na vida do bebê.

FONTE: O autor (2025)

FIGURA 23 – DESING DA LANDING PAGE PARA A QUESTÃO 2

FONTE: O autor (2025)

FIGURA 24 – JUSTIFICATIVA DE CORES, FONTE E LAYOUT PARA A QUESTÃO 3

- Cores

Foram escolhidas cores pasteis para combinar com o “tom da maternidade”. Amarelo pastel como cor principal e de fundo, roza pastel para a top bar (um contraste leve com a cor principal) e verde pastel para os botões de confirmação. Verde tem um bom contraste com o amarelo, funcionando bem para o destaque de ação principal e passando a ideia de “seguir caminho” ou “caminho correto”, geralmente utilizado para realizar a ação desejada (por exemplo, realizar o login na tela de login, ou criar um novo evento na tela de eventos).

- Fonte

Foi escolhida Quicksand como fonte principal devido ao seu estilo “suave” e um pouco infantil. Também foi escolhida Dancing Script, uma fonte de letra cursiva, na landing page para quebrar com a monotia de fontes tradicionais.

- Layout

O Layout busca ser simples e intuitivo, com bordas arredondadas e fontes grandes. Tentou-se passar, em geral, a ideia de acolhimento com o design das interfaces.

FONTE: O autor (2025)

FIGURA 25 – COMENTÁRIOS DA ENTREVISTADA PARA A QUESTÃO 4

Entrevistada: Gabrieli Sizilio

- Gabrieli relatou não ter gostado do contraste entre a cor de fundo e a cor do texto. Achou “seco”.
- Gostou do tamanho da fonte.
- Preferia que o botão de cadastro estivesse do lado direito e não esquerdo.
- Achou que o protótipo está muito cru e não chama atenção o suficiente.

FONTE: O autor (2025)

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

As disciplinas MOB 1 (Desenvolvimento Mobile 1) e MOB 2 (Desenvolvimento Mobile 2) abordam o tema de desenvolvimento de aplicativos para dispositivos móveis (principalmente *smartphones* e *tablets*). Com o surgimento de *smartphones*, surgiu um novo paradigma de desenvolvimento de software; e com a adoção massiva desses aparelhos pela população geral, surgiu a necessidade de implementar nesse novo paradigma versões de software que já existiam em paradigmas anteriores. Segundo o modelo clássico, o usuário deveria instalar um pedaço de software diretamente em sua máquina; segundo o modelo web, o usuário deveria instalar um navegador em sua máquina, e então interagir com aplicações web através de seu navegador; agora, segundo o modelo *mobile*, o usuário deve instalar aplicativos em seu dispositivo móvel através de um gerenciador de aplicativos que funciona como *martketplace.* Este gerenciador de aplicativos, pré-instalado nos dispositivos móveis, serve para publicação, instalação, atualização de aplicativos, sendo que o gerenciador se torna responsável por garantir que os aplicativos instalados sejam estáveis e seguros.

O projeto da disciplina MOB 1 consistia em um aplicativo *android* de finanças, no qual o usuário poderia ter um controle de sua vida financeira, administrando seus gastos (débitos) e ganhos (créditos). Esta aplicação deveria conter uma tela de dashboard, exibindo as funcionalidades existentes do aplicativo com as quais o usuário pode interagir; uma tela de cadastro de operações, onde o usuário pode inserir seus débitos e créditos; e uma tela de extrato, onde o usuário pode consultar operações realizadas.

O projeto da disciplina MOB 2 consistia em um aplicativo *android* com o tema *Harry Potter*. O aluno deveria consumir uma API pública da saga e realizar 3 funcionalidades principais. O usuário deveria ser capaz de recuperar informações sobre um personagem específico, de listar os professores da escola e de listar os alunos de uma casa. Nesta disciplina aproveita-se os conhecimentos de WEB 2, pois conhecimentos de consumo de APIs é necessário tanto para aplicações web quanto para aplicativos de dispositivos móveis.

Tendo em vista que o trabalho destas disciplinas era opcional, não serão apresentados artefatos do projeto.

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

O projeto da disciplina INFRA – Infraestrutura para Desenvolvimento e Implantação de Software (DevOps) consistia em seguir um passo para executar tarefas típicas de um profissional DevOps.

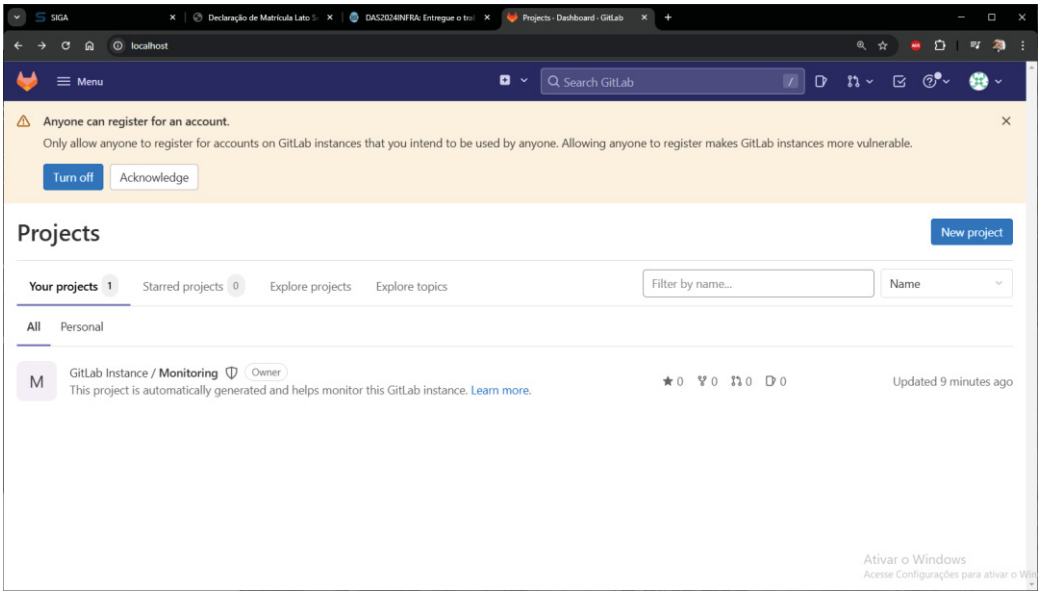
DevOps diz respeito à integração das etapas de desenvolvimento e implantação de um software. Suas etapas envolvem criação de ambientes, teste de código, implantação de código e monitoramento de métricas de produção (Kim *et. al.,* 2020).

Relacionado às práticas DevOps está o conceito de CI/CD (Integração Contínua / Entrega Contínua), que tem como objetivo automatização dos processos de desenvolvimento e implantação de software, além de tornar o ciclo de vida das aplicações fluido, previsível e seguro. Isso ressalta como práticas de DevOps e CI/CD estão intimamente alinhados aos métodos ágeis, como visto na disciplina MADS, ao propor desenvolver software de modo iterativo e incremental, com entregas contínuas e foco em feedback.

Neste projeto, o aluno entraria em contato com ferramentas comuns ao cotidiano das práticas DEVOPS, como Docker, Jenkins e GitLab; sendo Jenkins uma ferramenta de automação de processos, Docker uma ferramenta de gerenciamento de containers e GitLab uma plataforma de gerenciamento de repositórios Git. O desafio consistia em rodar uma imagem Docker baseada em GitLab e Jenkins expondo determinadas portas, realizar com *commit* e *push* de alterações no repositório, e registrar esses passos com a funcionalidade PrintScreen. A FIGURA 26 apresenta a interface do GitLab, a FIGURA 27 mostra o comando Docker para executar o container e a FIGURA 28 demonstra o *commit* e o *push* das alterações.

11.1 ARTEFATOS DO PROJETO

FIGURA 26 – INTERFACE DO GITLAB RODANDO DESDE O CONTAINER DOCKER



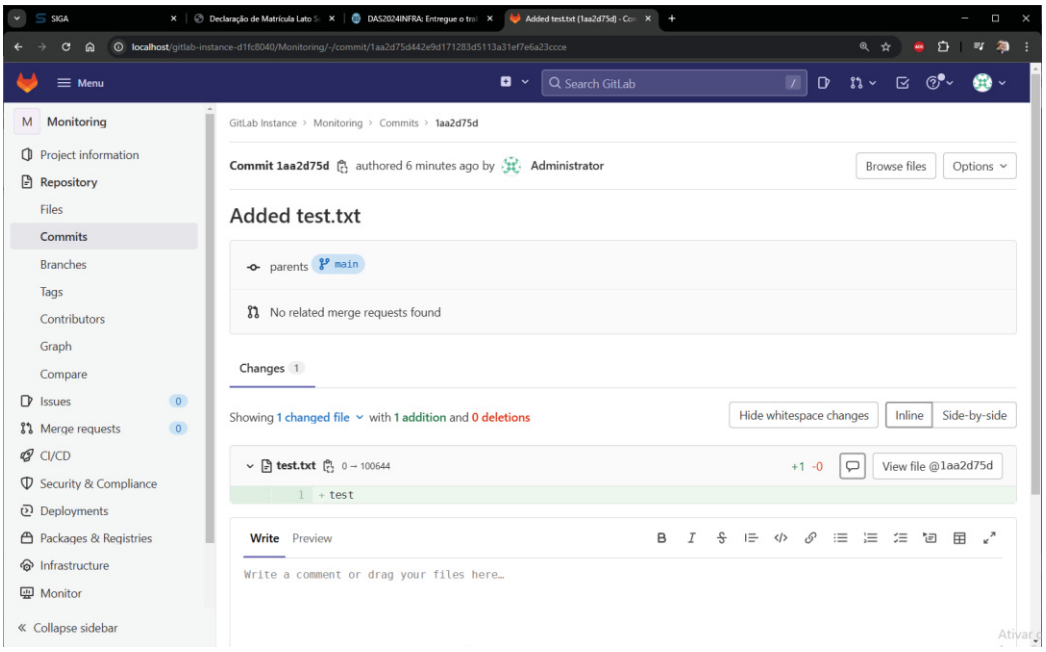
FONTE: O autor (2025)

FIGURA 27 – COMANDO PARA RODAR O CONTAINER DUCKER A PARTIR DA IMAGEM GITLAB/JENKINS

```
C:\Users\josep>docker container run -d --name 201800093924 -p 22:22 -p 80:80 -p 443:443 -p 9091:9091 dfwandarti/gitlab_jenkins:3
```

FONTE: O autor (2025)

FIGURA 28 – COMMIT E PUSH DE UM ALTERAÇÃO NO REPOSITÓRIO



FONTE: O autor (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

O projeto da disciplina TEST – Testes Automatizados consistiu na entrega de um código que realizasse um teste do tipo ponta a ponta. Um artefato deveria ser entregue nesta disciplina, a saber, um arquivo que executasse o teste.

Os testes ponta a ponta são realizados utilizando o driver de um navegador. Com esse driver, podemos emular o comportamento de um usuário real e simular comandos como cliques de mouse e pressionamento de teclas do teclado. Ao emular o comportamento do usuário, podemos testar a aplicação num alto nível, focando nos casos de uso da aplicação.

Neste projeto, o aluno deveria escrever um teste ponta a ponta utilizando o site aNotepad (https://pt.anotepad.com), onde é possível escrever notas. O teste deveria escrever o título da disciplina no título da nota e o nome e matrícula do aluno no corpo da nota. O código que executa estas tarefas é apresentado na FIGURA 29.

Esta disciplina trata de conceitos relacionados à disciplina INFRA, pois a parte de testes é essencial nas práticas de DevOps e no ciclo de CI/CD. Além disso, os conceitos aplicados nesta disciplina também são utilizados na disciplina INTO, cujo projeto consistia em aplicar a abordagem TDD (*Test Driven Development*, Desenvolvimento Orientado a Testes).

12.1 ARTEFATOS DO PROJETO

FIGURA 29 – COMENTÁRIOS DA ENTREVISTADA PARA A QUESTÃO 4

```
with sync_playwright() as playwright:

    firefox = playwright.firefox
    browser = firefox.launch(headless=HEADLESS, slow_mo=SLOW_MO)

    page = browser.new_page()
    page.goto(URL)

    title_input = page.locator(TITLE_INPUT_ID)
    title_input.fill(TITLE)

    body_input = page.locator(BODY_INPUT_ID)
    body_input.fill(BODY)

    browser.close()
```

FONTE: O autor (2025)

13 CONCLUSÃO

Neste documento, apresentou-se um conjunto de projetos desenvolvidos no contexto das disciplinas do curso de Pós-Graduação em Desenvolvimento Ágil de Software, estruturados na forma de um memorial de projetos. Esses trabalhos evidenciam a aplicação prática dos princípios, métodos e valores que sustentam o desenvolvimento ágil, demonstrando não apenas o domínio técnico alcançado, mas também o amadurecimento na compreensão das dinâmicas colaborativas e iterativas que caracterizam esse paradigma.

Observa-se como a adoção de práticas ágeis impacta positivamente o ciclo de desenvolvimento e a entrega de valor ao cliente. Os projetos desenvolvidos mostraram que a agilidade vai além de métodos e ferramentas: trata-se de uma mentalidade voltada à colaboração, transparência e adaptação constante. Assim, o curso proporcionou não apenas capacitação técnica, mas também o desenvolvimento de competências interpessoais.

Contudo, também ficou evidente que a implementação efetiva de métodos ágeis ainda enfrenta desafios significativos. A transição de modelos tradicionais para abordagens ágeis requer tempo, comprometimento e uma mudança cultural profunda dentro das organizações (Oliveira; Pedron, 2021). Olhando para o futuro, observa-se uma tendência de integração cada vez maior entre agilidade, engenharia de software moderna e práticas de DevOps. Assim, o aprendizado adquirido neste curso não se encerra com sua conclusão, mas serve como base sólida para enfrentar novos desafios, promover a inovação contínua e consolidar uma cultura de melhoria permanente no desenvolvimento de software.

REFERÊNCIAS

BECK, K. S. K. S. J. E. A.. **Manifesto for agile software development**. [S. l.: s. n.], 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 23 out. 2025.

KIM, Gene; BEHR, Kevin; SPafford, George; HUMBLE, Jez. **The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations**. 2.ª ed. IT Revolution Press, 2020.

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional.** 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILLE, I. **Software engineering**. 8. ed. [s.l.] Addison-Wesley, 2007.

OLIVEIRA, R. L. F.; PEDRON, C. D. **Métodos ágeis: uma revisão sistemática sobre benefícios e limitações**. Brazilian Journal of Development, 7(1): 4520–4534, 2021.

LAPORTE, C. Y.; APRIL, A. Software quality assurance. Hoboken, Nj: Ieee Press, 2018.