UNIVERSIDADE FEDERAL DO PARANÁ

MATEUS MAIDEL ALVES DA SILVA

MEMORIAL DE PROJETOS: A APLICAÇÃO PRÁTICA DO CICLO ÁGIL NO DESENVOLVIMENTO *FULL-STACK*

CURITIBA

2025

MATEUS MAIDEL ALVES DA SILVA

MEMORIAL DE PROJETOS: A APLICAÇÃO PRÁTICA DO CICLO ÁGIL NO DESENVOLVIMENTO *FULL-STACK*

Memorial de Projetos apresentado ao curso de 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

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 **MATEUS MAIDEL ALVES DA SILVA**, intitulada: **MEMORIAL DE PROJETOS: A APLICAÇÃO PRÁTICA DO CICLO ÁGIL NO DESENVOLVIMENTO *FULL-STACK***, 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, 25 de Novembro de 2025.

RAZER ANTHOM NIZER ROJAS MONTANO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial apresenta o desenvolvimento de diferentes projetos realizados ao longo da Pós-Graduação em Desenvolvimento Ágil de Software da Universidade Federal do Paraná, com o objetivo de consolidar os conhecimentos adquiridos nas disciplinas por meio da construção de soluções práticas que abrangem desde o levantamento e análise de requisitos até a implementação de sistemas funcionais. Durante o curso, foram desenvolvidos diversos artefatos de apoio, como diagramas, modelos de dados, códigos e testes, aplicados em cenários que simulam demandas reais do mercado. O documento inclui, ainda, um parecer técnico que sintetiza a evolução dos conhecimentos e evidencia a contribuição dos projetos desenvolvidos para a prática profissional. A metodologia adotada baseou-se na aplicação incremental dos conteúdos, permitindo a evolução contínua dos projetos e o fortalecimento das práticas de engenharia de software. Como resultado, obteve-se um conjunto de entregas que demonstram a capacidade de planejar, projetar e implementar soluções tecnológicas de forma estruturada, reforçando a importância da integração entre teoria e prática. Conclui-se que a experiência contribuiu significativamente para o aprimoramento profissional, ampliando a visão sobre o desenvolvimento ágil de software e fortalecendo a habilidade de lidar com desafios reais da área de tecnologia.

Palavras-chave: desenvolvimento ágil; engenharia de software; projetos de software; metodologia incremental;

ABSTRACT

This memorial presents the development of different projects carried out during the Graduate Program in Agile Software Development at the Federal University of Paraná, aiming to consolidate the knowledge acquired in the courses through the construction of practical solutions ranging from requirements analysis to the implementation of functional systems. Throughout the program, several supporting artifacts were developed, such as diagrams, data models, source code, and tests, applied in scenarios that simulate real market demands. The document also includes a technical report that synthesizes the evolution of knowledge and highlights the contribution of the developed projects to professional practice. The methodology adopted was based on the incremental application of the course contents, allowing the continuous evolution of the projects and the strengthening of software engineering practices. As a result, a set of deliverables was produced that demonstrates the ability to plan, design, and implement technological solutions in a structured manner, reinforcing the importance of integrating theory and practice. It is concluded that this experience contributed significantly to professional improvement, broadening the understanding of agile software development and strengthening the ability to deal with real challenges in the field of technology.

Keywords: agile development; software engineering; software projects; incremental methodology;

LISTA DE FIGURAS

FIGURA 1 - MAPA MENTAL (PARTE 1 DE 3)....12
FIGURA 2 - MAPA MENTAL (PARTE 2 DE 3)....13
FIGURA 3 - MAPA MENTAL (PARTE 3 DE 3)....13
FIGURA 4 - DIAGRAMA DE CASO DE USO DE NIVEL 1....14
FIGURA 5 - DIAGRAMA DE CLASSES COM ATRIBUTOS ASSOCIADOS....16
FIGURA 6 - CÁLCULO DA VELOCIDADE....17
FIGURA 7 - RESULTADO FINAL....20
FIGURA 8 - DIAGRAMA DE FLUXO CUMULATIVO....20
FIGURA 9 - EVIDÊNCIAS DOS TESTES QUE PASSARAM....22
FIGURA 10 - MODELO LÓGICO....24
FIGURA 11 - TELA DE LOGIN....30
FIGURA 12 - TELA INICIAL....30
FIGURA 13 - TELA DE PROJETOS....31
FIGURA 14 - CONTAINER EM EXCECUÇÃO....35
FIGURA 15 - BROWSER....36
FIGURA 16 - LOG DO COMMIT....36
FIGURA 17 - CÓDIGO E EXECUÇÃO DO TESTE AUTOMATIZADO COM PLAYWRIGHT....38

SUMÁRIO

1 PARECER TÉCNICO..... 8
2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE..... 10
2.1 ARTEFATOS DO PROJETO..... 11
3 DISCIPLINA: MAG1 E MAG2 – MODELAGEM ÁGIL DE SOFTWARE 1 E 2.....13
3.1 ARTEFATOS DO PROJETO..... 13
4 DISCIPLINA: GAP1 E GAP2 – GERENCIAMENTO ÀGIL DE PROJETOS DE SOFTWARE 1 E 2.....15
4.1 ARTEFATOS DO PROJETO..... 15
5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO.....20
5.1 ARTEFATOS DO PROJETO..... 21
6 DISCIPLINA: BD – BANCO DE DADOS.....22
6.1 ARTEFATOS DO PROJETO..... 23
7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO.....26
7.1 ARTEFATOS DO PROJETO..... 27
8 DISCIPLINA: WEB1 E WEB2 – DESENVOLVIMENTO WEB 1 E 2.....29
8.1 ARTEFATOS DO PROJETO..... 29
9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE.....30
9.1 ARTEFATOS DO PROJETO..... 31
10 DISCIPLINA: MOB1 E MOB2 – DESENVOLVIMENTO MOBILE 1 E 2.....33
10.1 ARTEFATOS DO PROJETO..... 34
11 DISCIPLINA: INFRA - INFRAESTRUTURA PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SOFTWARE (DEVOPS).....35
11.1 ARTEFATOS DO PROJETO..... 36
12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS.....38
12.1 ARTEFATOS DO PROJETO..... 40
13 CONCLUSÃO..... 41
REFERÊNCIAS..... 42

1 PARECER TÉCNICO

Este memorial apresenta uma análise integrada e reflexiva dos projetos desenvolvidos ao longo da Especialização em Desenvolvimento Ágil de Software. Mais do que apenas uma sucessão de entregas, esta trajetória evidencia um processo de aprendizado contínuo, onde os acertos solidificaram o conhecimento e os desafios proporcionaram as lições mais valiosas sobre a aplicação real dos princípios ágeis.

A jornada iniciou na disciplina de Métodos Ágeis para Desenvolvimento de Software (MADS), com a criação de um mapa mental. A entrega inicial, reconhecidamente superficial, carecia do detalhamento necessário para tópicos cruciais como Scrum (Schwaber; Sutherland, 2020), princípios ágeis conforme definidos pelo Manifesto Ágil (Beck *et al.*, 2001), Extreme Programming (XP) (Beck, 2004), Lean Software Development (Poppendieck; Poppendieck, 2003) e Entrega Contínua (Humble; Farley, 2010). Este feedback foi fundamental, pois destacou a importância de se aprofundar nos fundamentos teóricos que servem como alicerce para toda a prática subsequente. Um entendimento robusto desses conceitos é indispensável para a tomada de decisões consciente no desenvolvimento de software.

A fase de planejamento, exercitada em Gerenciamento Ágil de Projetos (GAP1 e GAP2), trouxe outro aprendizado crucial. A elaboração do plano de release para um sistema de *delivery* expôs uma dificuldade prática comum: a criação de um planejamento irrealista. A distribuição inadequada das histórias de usuário, onde a soma das estimativas excedia em muito a velocidade da *sprint*, resultou em um *backlog* impossível de ser executado no tempo previsto. Este erro, fruto de uma execução apressada, reforçou na prática um dos pilares do ágil: a importância do planejamento realista e baseado em dados, essencial para a transparência e previsibilidade do projeto (Poppendieck; Poppendieck, 2003).

As disciplinas de Modelagem Ágil de Software (MAG1 e MAG2) forneceram a ponte necessária entre a teoria e o código. A modelagem de um sistema de gestão de condomínio, utilizando diagramas UML (Booch; Rumbaugh; Jacobson, 2005) e histórias de usuário como instrumento de elicitação e comunicação de requisitos (Cohn, 2004), demonstrou como uma boa documentação ágil facilita a comunicação

e reduz ambiguidades, preparando o terreno para um desenvolvimento mais eficiente e agilidade.

O desenvolvimento prático começou com Introdução à Programação (INTRO) e Banco de Dados (BD), onde a implementação de um sistema bancário seguindo a prática de *Test-Driven Development* (TDD) introduziu a cultura de qualidade desde o início do ciclo. Martin (2008) defende que o TDD é uma prática de design que resulta em código mais limpo e confiável. Este fundamento foi essencial para Aspectos Ágeis de Programação (AAP), onde o desafio foi a refatoração do algoritmo *Bubble Sort*. O feedback recebido aqui foi revelador: a simples remoção de comentários não era suficiente. O código precisava se autoexplicar através de nomes significativos e métodos concisos para reduzir sua complexidade ciclomática. Esta foi uma lição prática e inesquecível sobre os princípios de *Clean Code* conforme propostos por Martin (2008), mostrando que a clareza do código é um requisito não funcional crítico para a manutenibilidade e agilidade.

As disciplinas de Desenvolvimento Web (WEB1 e WEB2), Desenvolvimento Mobile (MOB1 e MOB2) e Experiência do Usuário (UX) aplicaram esses conceitos no desenvolvimento *front-end* e *back-end*, criando aplicações responsivas e centradas no usuário. Por fim, as disciplinas de Infraestrutura para Desenvolvimento e Implantação de Software (DevOps) e Testes Automatizados (TEST) fecharam o ciclo, automatizando processos de integração, entrega e garantia de qualidade com o uso de Docker para empacotamento e execução de ambientes isolados (Boettiger, 2015) e Selenium para automação de testes de interface (Selenium, 2023).

Em conclusão, os projetos realizados demonstram de forma coesa que o desenvolvimento ágil é um sistema integrado que requer tanto competência técnica quanto disciplinas de gestão e planejamento. Os desafios encontrados: a necessidade de aprofundamento teórico, a importância de um planejamento realista e a busca incessante por um código claro e legível, não foram obstáculos, mas sim as peças centrais do aprendizado. Eles evidenciam que a verdadeira agilidade não é sobre perfeição desde o primeiro passo, mas sobre a capacidade de iterar, aprender com os feedbacks e melhorar continuamente, que é a essência máxima do modelo ágil.

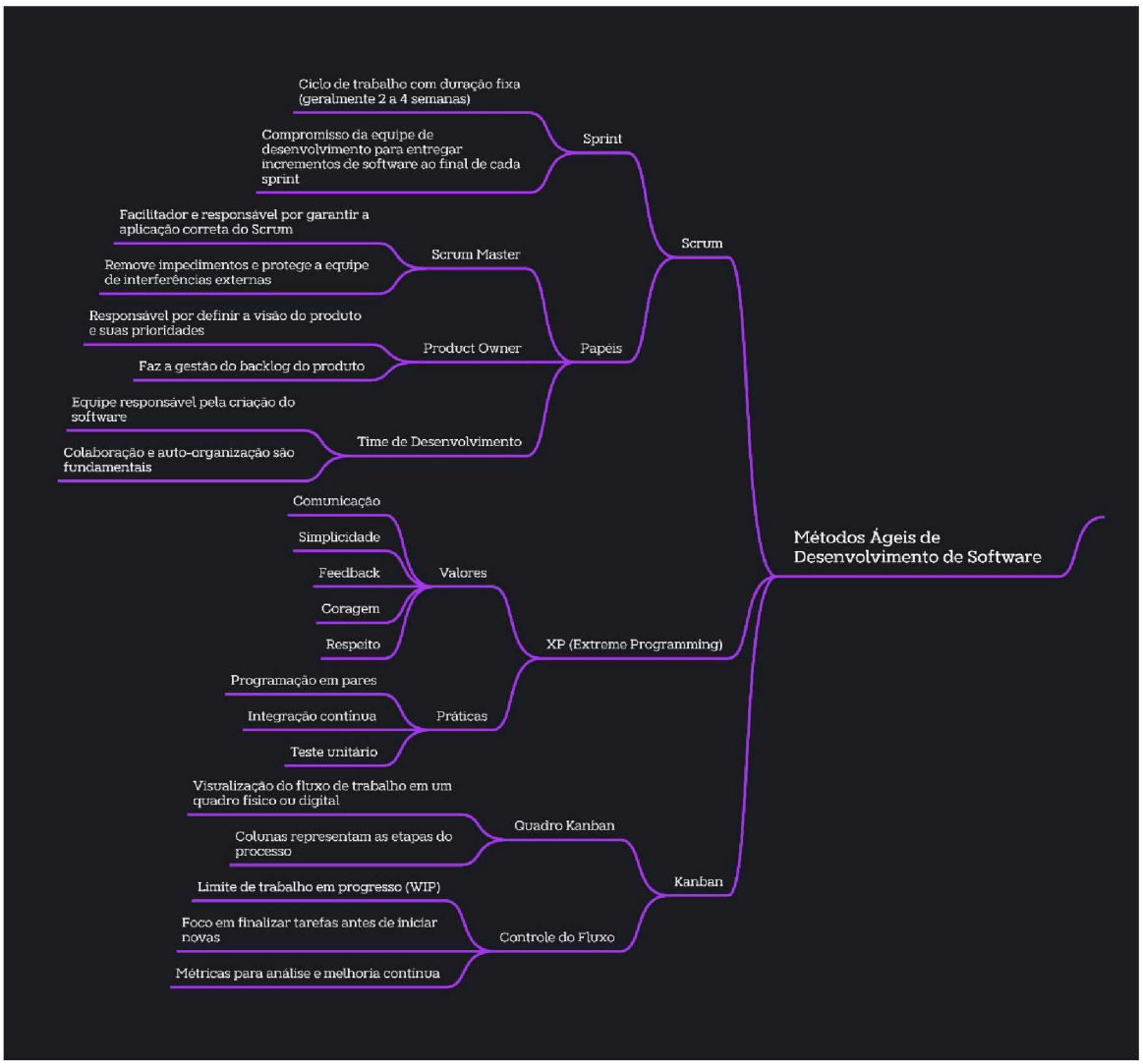
2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

O projeto da disciplina foi a criação de um mapa mental sobre métodos ágeis. A versão entregue, desenvolvida com pouco tempo disponível, foi avaliada como superficial, faltando detalhamento em tópicos essenciais como Scrum (Schwaber; Sutherland, 2020), Extreme Programming (XP) (Beck, 2004), Lean Software Development (Poppendieck; Poppendieck, 2003) e Entrega Contínua (Humble; Farley, 2010).

Embora o mapa não tenha sido refeito, o feedback recebido foi crucial. Ele deixou claro que a teoria é a base da prática. Essa lição foi levada para as disciplinas seguintes, mostrando que entender os conceitos é fundamental antes de aplicar qualquer ferramenta ou método.

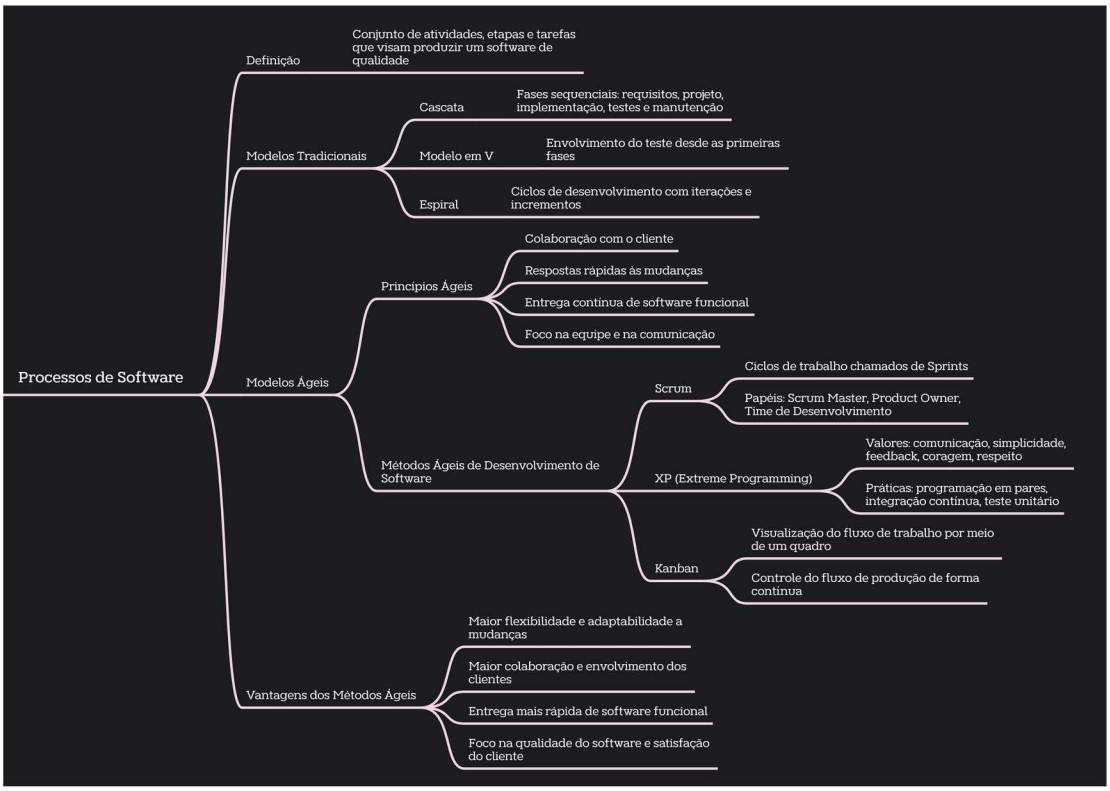
2.1 ARTEFATOS DO PROJETO

FIGURA 1 - MAPA MENTAL (PARTE 1 DE 3)



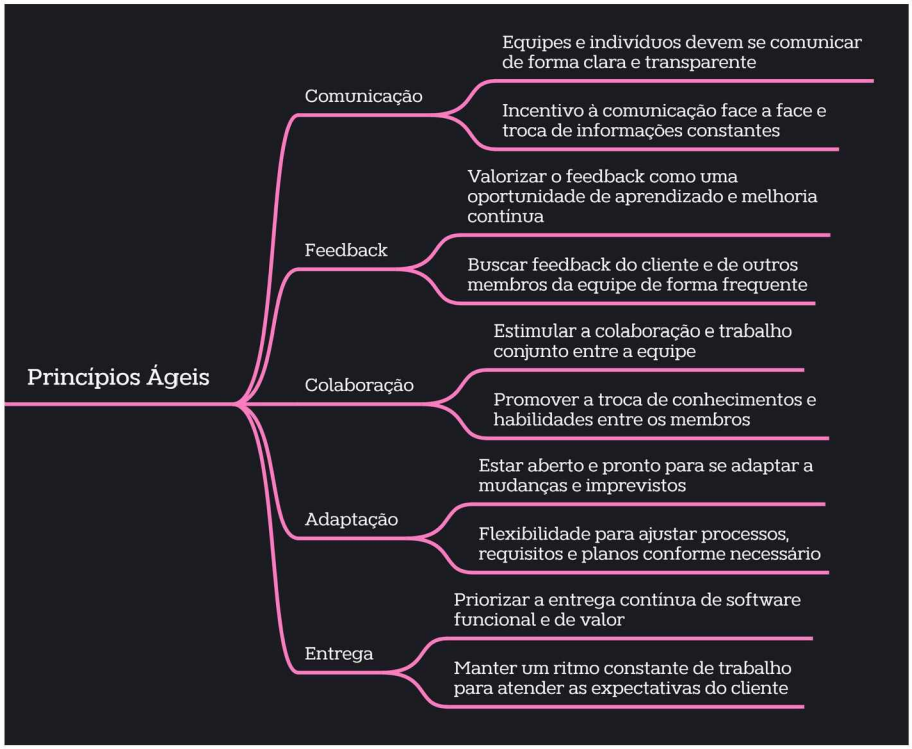
FONTE: O AUTOR (2025)

FIGURA 2 - MAPA MENTAL (PARTE 2 DE 3)



FONTE: O AUTOR (2025)

FIGURA 3 - MAPA MENTAL (PARTE 3 DE 3)



FONTE: O AUTOR (2025)

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

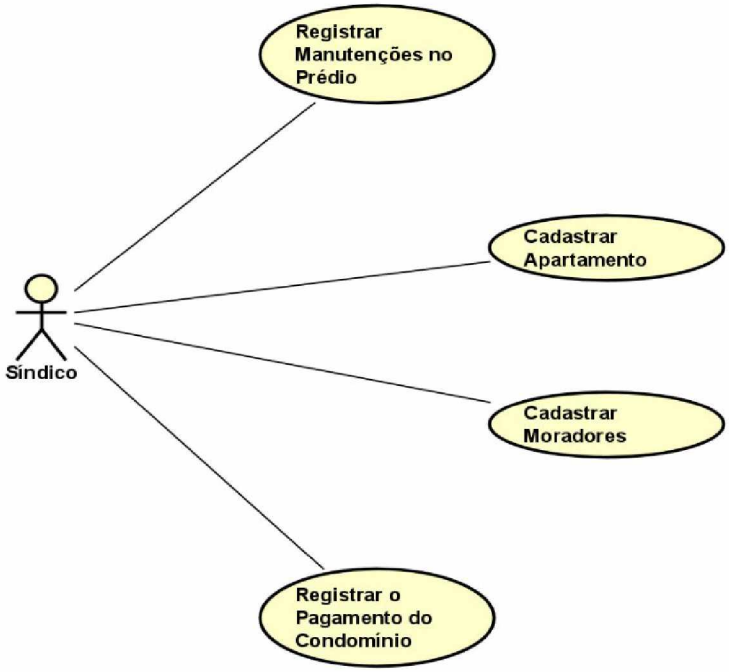
Em MAG1 e MAG2, o projeto foi modelar um sistema de gestão de condomínio. Em MAG1, focamos na visão funcional, criando diagramas de caso de uso e histórias de usuário detalhadas. Em MAG2, evoluímos para a visão estrutural, com diagramas de classes e diagramas de sequência, seguindo as notações definidas na Unified Modeling Language (UML) (Booch; Rumbaugh; Jacobson, 2005).

A modelagem foi essencial para definir os requisitos de forma clara antes de partir para o código. As histórias de usuário com critérios de aceitação bem escritos serviram como guia para o desenvolvimento nas disciplinas de programação (INTRO, WEB). Já os diagramas de classes (MAG2) foram a base direta para a criação do banco de dados na disciplina de BD.

Essa é a essência do ágil: planejar e modelar de forma eficiente para desenvolver com mais precisão e menos retrabalho.

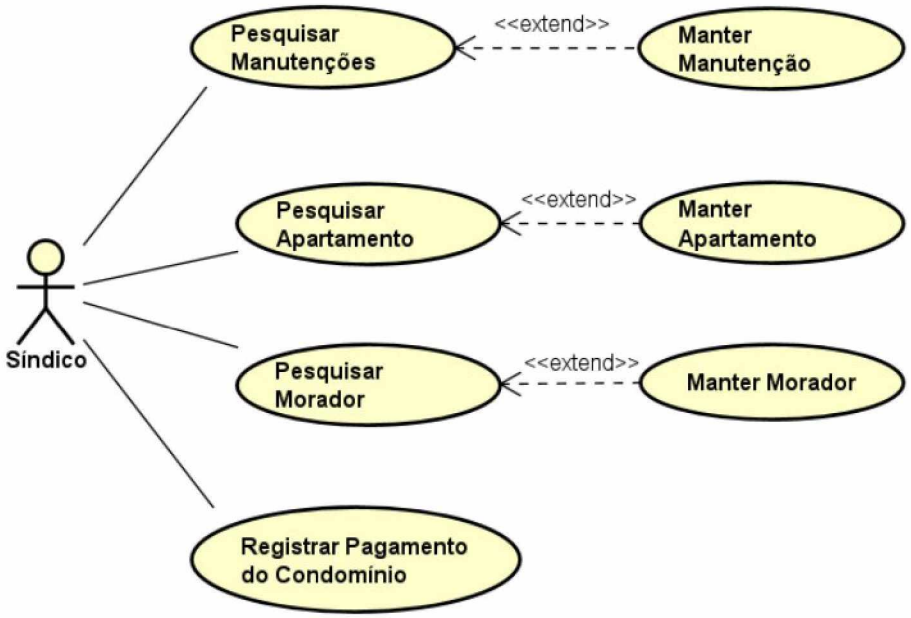
3.1 ARTEFATOS DO PROJETO

FIGURA 4 - DIAGRAMA DE CASO DE USO DE NIVEL 1



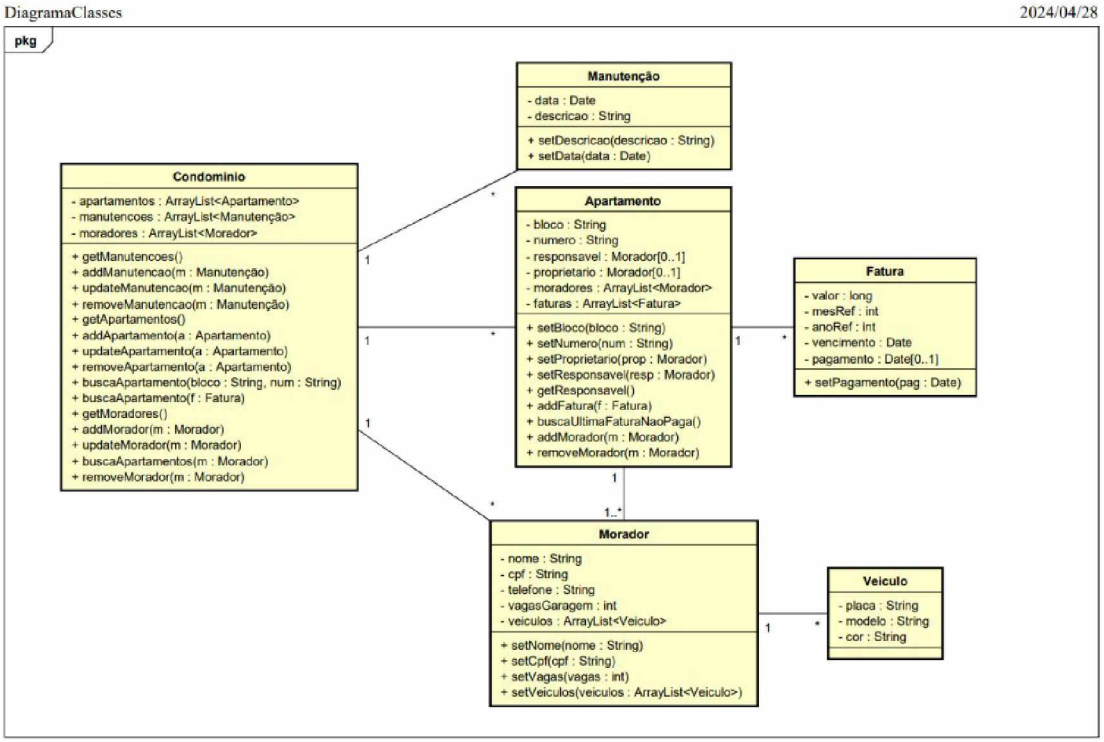
FONTE: O AUTOR (2025)

Figura 5 – Diagrama de Caso de Uso de Nível 2



FONTE: O AUTOR (2025)

FIGURA 5 - DIAGRAMA DE CLASSES COM ATRIBUTOS ASSOCIADOS



FONTE: O AUTOR (2025)

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

Em GAP1, o projeto foi criar um plano de release para um aplicativo de adoção de gatos. A atividade envolvia calcular a velocidade da equipe e distribuir histórias de usuário estimadas ao longo de *sprints*. O *feedback* recebido foi crucial: a soma das estimativas das histórias em uma *sprint* excedia a velocidade da equipe, resultando em um planejamento irrealista. Essa lição prática destacou a importância de um planejamento factível, que é a base para a transparência e previsibilidade no ágil.

Em GAP2, o foco foi na execução e melhoria contínua do fluxo de trabalho, utilizando uma simulação de Kanban para gerenciamento visual de tarefas (Anderson, 2010). O objetivo era organizar o quadro de trabalho para maximizar a eficiência e analisar os resultados por meio de um Diagrama de Fluxo Cumulativo (Cumulative Flow Diagram), útil para observar gargalos e evolução do processamento ao longo do tempo (Kniberg; Skarin, 2010). Esta disciplina complementou o GAP1, mostrando na prática como visualizar e melhorar o fluxo de valor após o planejamento inicial.

Juntas, essas disciplinas formam o ciclo de gestão ágil: planejar o que será feito (GAP1) e otimizar continuamente como é feito (GAP2). O plano de release de GAP1 forneceria o *backlog* priorizado para as disciplinas de desenvolvimento (WEB I e II, MOB I e II), enquanto o Kanban de GAP2 gerenciaria o fluxo de tarefas dessas mesmas disciplinas.

4.1 ARTEFATOS DO PROJETO

FIGURA 6 - CÁLCULO DA VELOCIDADE

Cálculo da Velocidade:

Horas disponíveis por dia:	**horas**	Tamanho da Sprint:	**2 semanas**
Horas disponíveis por Sprint:	**30 horas**	Velocidade:	**7 pontos**

FONTE: O AUTOR (2025)

QUADRO 7 - CÁLCULO DA VELOCIDADE

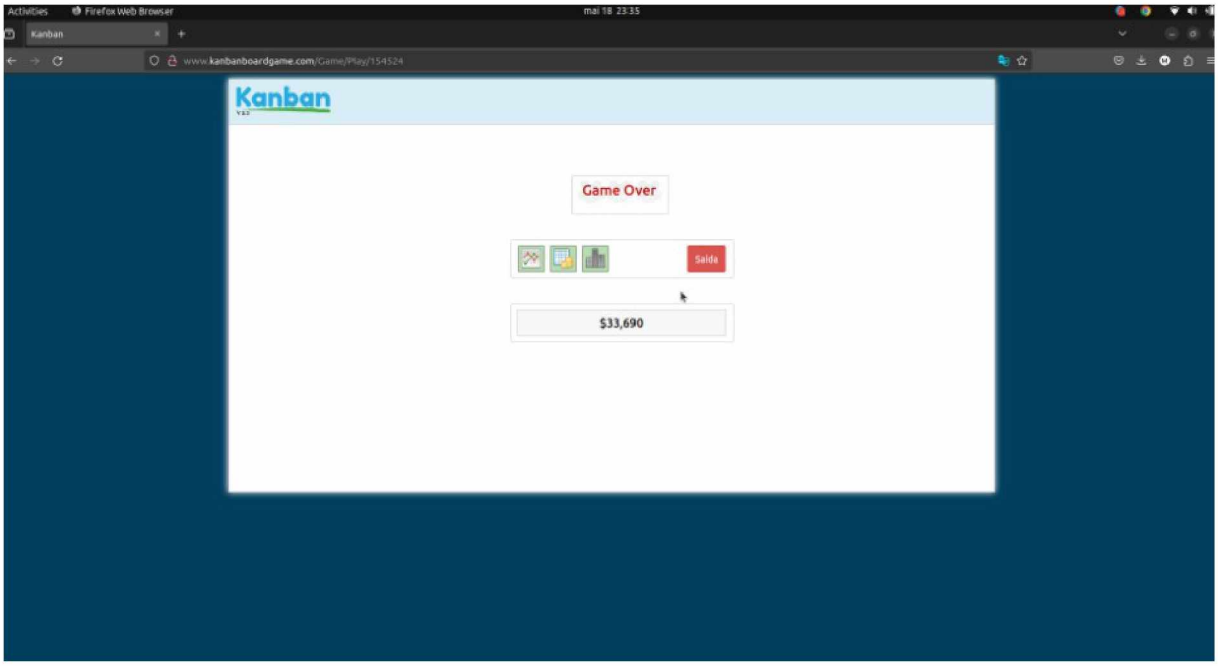
ID	SENDO	QUERO	PARA	Estimativa (pts)	Sprint / Iteração
HU01	Usuário em busca de gatos para adoção	Ver uma lista de gatos disponíveis para adoção, separados por bairro	Facilitar a busca de gatos para adoção	5	Sprint 1
HU02	Usuário interessado em um gato	Enviar uma mensagem para o atual responsável do gato	Obter mais informações sobre o gato ou iniciar o processo de adoção	3	Sprint 1
HU03	Usuário pronto para adotar um gato	Enviar documentos necessários para o processo de adoção	Concluir o processo de adoção de forma rápida e eficiente	5	Sprint 1
HU04	Usuário com um gato para adoção	Adicionar um novo gato para adoção, incluindo informações e fotos	Aumentar a disponibilidade de gatos para adoção no aplicativo	4	Sprint 2
HU05	Usuário que precisa atualizar informações de um gato	Editar as informações de um gato já cadastrado	Manter as informações dos gatos atualizadas e precisas	3	Sprint 2
HU06	Usuário que não deseja mais manter um gato para adoção	Remover um gato da lista de adoção	Manter a lista de gatos para adoção atualizada e relevante	3	Sprint 2
HU07	Usuário que adotou um gato	Marcar o gato como adotado no aplicativo	Atualizar o status do gato na lista e encerrar o processo de adoção	3	Sprint 3
HU08	Novo usuário do aplicativo	Criar uma conta para acessar todas as funcionalidades do aplicativo	Aproveitar todas as funcionalidades do aplicativo, incluindo a adoção de gatos	4	Sprint 3
HU09	Usuário procurando um gato com características específicas	Filtrar os gatos disponíveis por características como idade, cor, raça, etc.	Encontrar um gato que atenda às minhas preferências e necessidades	4	Sprint 1
HU010	Usuário interessado em receber recomendações de gatos para adoção	Receber sugestões de gatos com base nas minhas preferências	Facilitar a escolha de um gato para adoção	5	Sprint 1
HU011	Usuário querendo ver mais fotos de um gato	Visualizar uma galeria de fotos do gato	Conhecer melhor o gato antes de decidir pela adoção	3	Sprint 2

HU012	Usuário desejando compartilhar informações sobre um gato nas redes sociais	Compartilhar informações sobre o gato em redes sociais como Facebook e Twitter	Aumentar a visibilidade dos gatos e facilitar a adoção	2	Sprint 2
HU013	Usuário querendo receber notificações sobre novos gatos disponíveis	Receber notificações sobre novos gatos adicionados ao aplicativo	Não perder a oportunidade de adotar um novo gato	4	Sprint 3
HU014	Usuário interessado em informações detalhadas de um gato	Visualizar informações como idade, raça, temperamento, saúde, etc.	Tomar uma decisão informada sobre a adoção	3	Sprint 3
HU015	Usuário que deseja salvar gatos favoritos	Marcar gatos como favoritos para fácil acesso posterior	Facilitar a revisitação de gatos que me interessaram	2	Sprint 3
HU016	Usuário interessado em eventos de adoção	Visualizar eventos de adoção próximos à minha localização	Participar de eventos para encontrar o gato perfeito para adoção	3	Sprint 4
HU017	Usuário querendo ver avaliações de outros adotantes	Ler avaliações e comentários de outros adotantes sobre gatos para adoção	Obter feedback sobre a experiência e a personalidade dos gatos	3	Sprint 4
HU018	Usuário interessado em suporte pós-adoção	Ter acesso a recursos e suporte sobre cuidados com o gato após a adoção	Garantir uma transição suave e cuidados adequados ao gato adotado	4	Sprint 1
HU019	Usuário que encontrou um gato para adoção	Agendar uma visita para conhecer o gato pessoalmente antes da adoção	Garantir uma boa compatibilidade entre adotante e gato	2	Sprint 2
HU020	Usuário que deseja contribuir para a comunidade de adoção	Ter a opção de doar para abrigos ou apoiar financeiramente a causa	Ajudar a fornecer cuidados e apoio contínuo a gatos necessitados	3	Sprint 3
HU021	Usuário que encontrou um gato, mas precisa de mais tempo para decidir	Marcar um gato como “favorito temporário” para revisão posterior	Facilitar a consideração e a tomada de decisão sobre a adoção	2	Sprint 4
HU022	Usuário	Ter acesso a	Entender o	3	Sprint 4

	interessado em saber mais sobre a personalidade de um gato	descrições detalhadas de comportamento e personalidade	temperamento do gato antes da adoção		
HU023	Usuário que adotou um gato e deseja compartilhar sua experiência	Deixar um depoimento ou avaliação sobre o processo e o gato adotado	Compartilhar experiência positiva e encorajar outros a adotarem	2	Sprint 4

FONTE: O AUTOR (2025)

FIGURA 8 - RESULTADO FINAL



FONTE: O AUTOR (2025)

FIGURA 9 - DIAGRAMA DE FLUXO CUMULATIVO

FONTE: O AUTOR (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

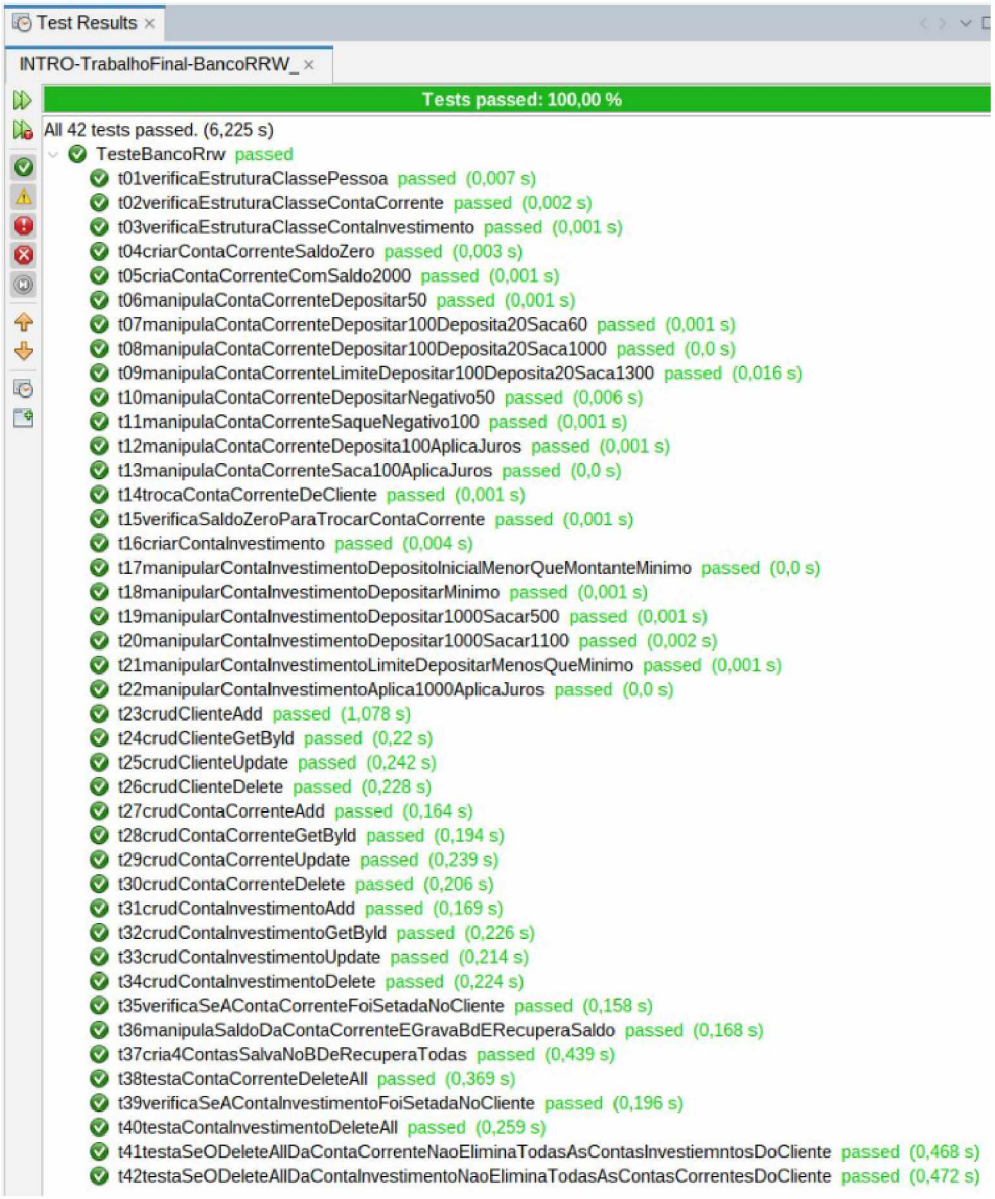
O projeto da disciplina foi implementar o backend de um sistema bancário em Java, com controle de clientes, contas-correntes e investimentos. O desenvolvimento seguiu a abordagem de *Test-Driven Development* (TDD), onde os testes unitários foram escritos antes do código (Beck, 2003) (Gamma *et al.*, 1994).

A disciplina foi fundamental para consolidar a base de programação necessária para o desenvolvimento ágil. A prática de TDD assegurou que o código fosse confiável desde o início, facilitando a refatoração e novas funcionalidades - princípios essenciais para a agilidade.

Este projeto serviu de base técnica para todas as disciplinas de desenvolvimento subsequentes. A lógica de programação e a estrutura de classes implementadas aqui foram diretamente aplicadas em Aspectos Ágeis de Programação (AAP), Desenvolvimento Web I e II (WEB) e Banco de Dados (BD).

5.1 ARTEFATOS DO PROJETO

FIGURA 10 - EVIDÊNCIAS DOS TESTES QUE PASSARAM



FONTE: O AUTOR (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

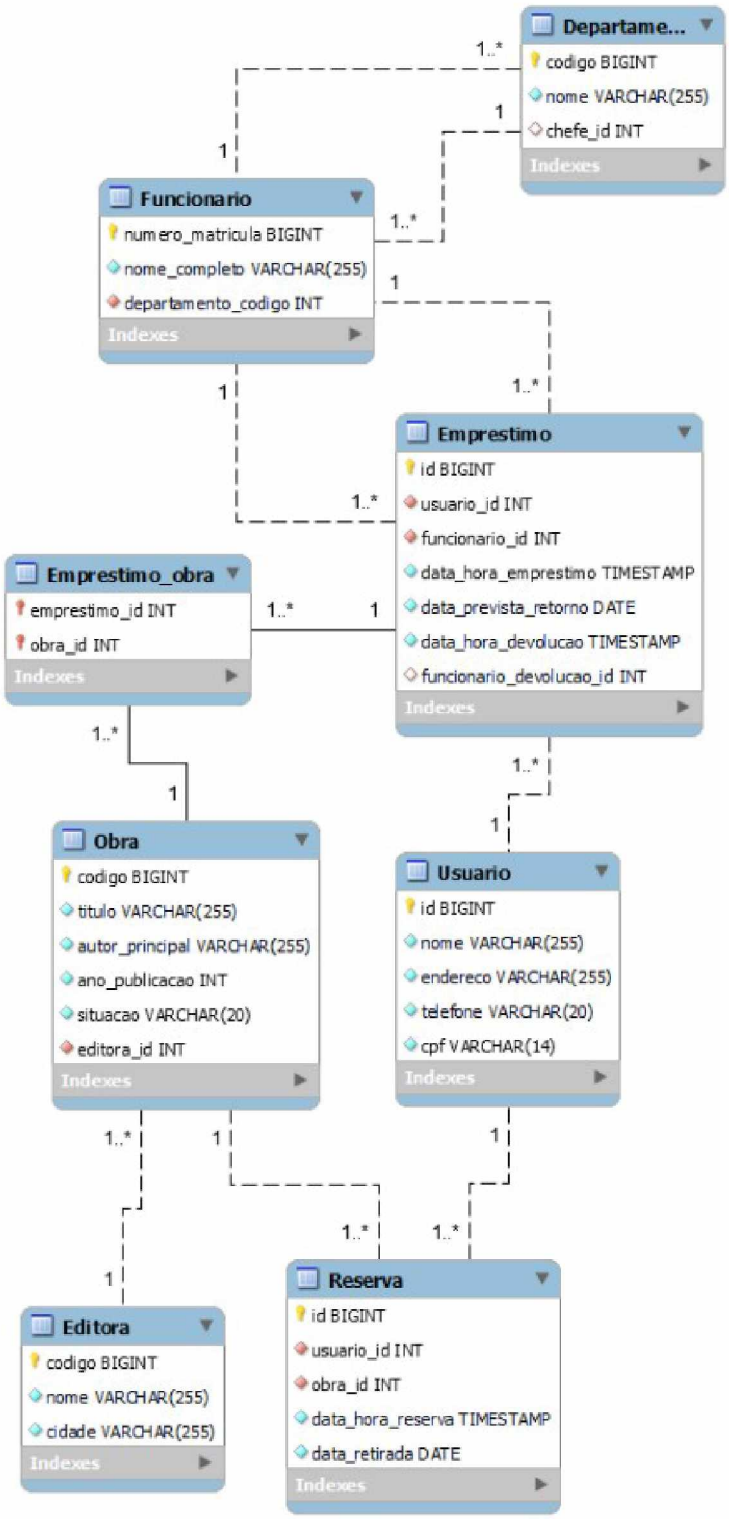
Na disciplina de Banco de Dados, o projeto envolveu a modelagem e implementação de dois sistemas: um para controle de biblioteca e outro para gestão de estoque e pedidos. A primeira parte focou na criação do modelo conceitual e lógico, seguindo as diretrizes de modelagem de dados relacionais (Elmasri; Navathe, 2019). A segunda etapa exigiu a implementação completa em SQL, com criação de tabelas, inserção de dados e estabelecimento de relacionamentos complexos entre entidades (Silberschatz; Korth; Sudarshan, 2019).

Este projeto foi essencial para entender como a modelagem de dados sustenta o desenvolvimento ágil. Um banco de dados bem estruturado é a base sobre a qual as aplicações das disciplinas de programação (INTRO, WEB, MOB) são construídas. Os diagramas de entidade-relacionamento criados aqui são a materialização dos diagramas de classes desenvolvidos em MAG2, mostrando a integração direta entre modelagem de software e modelagem de dados.

A implementação prática em SQL, com relacionamentos 1:N e N:M, forneceu a base de dados necessária para que os sistemas desenvolvidos nas outras disciplinas tivessem onde persistir e recuperar informações de forma eficiente.

6.1 ARTEFATOS DO PROJETO

FIGURA 11 - MODELO LÓGICO



FONTE: O AUTOR (2025)

CÓDIGO 1 – SCRIPT DE CRIAÇÃO DAS TABELAS

```
CREATE TABLE IF NOT EXISTS `mydb`.`Categoria` (
`categoria_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
`nome` VARCHAR(255) NOT NULL,
`descricao` TEXT NULL DEFAULT NULL,
PRIMARY KEY (`categoria_id`));
CREATE TABLE IF NOT EXISTS `mydb`.`Fornecedor` (
`fornecedor_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
`nome` VARCHAR(255) NOT NULL,
`nome_contato` VARCHAR(255) NULL DEFAULT NULL,
`endereco` TEXT NULL DEFAULT NULL,
`telefone` VARCHAR(20) NULL DEFAULT NULL,
`email` VARCHAR(255) NULL DEFAULT NULL,
PRIMARY KEY (`fornecedor_id`));
CREATE TABLE IF NOT EXISTS `mydb`.`Produto` (
`produto_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
`nome` VARCHAR(255) NOT NULL,
`descricao` TEXT NULL DEFAULT NULL,
`categoria_id` INT NULL DEFAULT NULL,
`fornecedor_id` INT NULL DEFAULT NULL,
`quantidade_em_estoque` INT NULL DEFAULT 0,
`nivel_reposicao` INT NULL DEFAULT 0,
`preco_unitario` DECIMAL(10,2) NULL DEFAULT NULL,
PRIMARY KEY (`produto_id`),
INDEX (`categoria_id` ASC) VISIBLE,
INDEX (`fornecedor_id` ASC) VISIBLE,
CONSTRAINT ``
FOREIGN KEY (`categoria_id`)
REFERENCES `mydb`.`Categoria` (`categoria_id`),
CONSTRAINT ``
FOREIGN KEY (`fornecedor_id`)
REFERENCES `mydb`.`Fornecedor` (`fornecedor_id`));
CREATE TABLE IF NOT EXISTS `mydb`.`EntradaEstoque` (
`entrada_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
`produto_id` INT NULL DEFAULT NULL,
`quantidade` INT NOT NULL,
`data_entrada` TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
`fornecedor_id` INT NULL DEFAULT NULL,
`preco_unitario` DECIMAL(10,2) NULL DEFAULT NULL,
PRIMARY KEY (`entrada_id`),
INDEX (`produto_id` ASC) VISIBLE,
INDEX (`fornecedor_id` ASC) VISIBLE,
CONSTRAINT ``
FOREIGN KEY (`produto_id`)
REFERENCES `mydb`.`Produto` (`produto_id`),
CONSTRAINT ``
FOREIGN KEY (`fornecedor_id`)
REFERENCES `mydb`.`Fornecedor` (`fornecedor_id`));
CREATE TABLE IF NOT EXISTS `mydb`.`SaidaEstoque` (
`saida_id` INT NULL DEFAULT NULL AUTO_INCREMENT,
`produto_id` INT NULL DEFAULT NULL,
`quantidade` INT NOT NULL,
`data_saida` TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
PRIMARY KEY (`saida_id`),
INDEX (`produto_id` ASC) VISIBLE,
CONSTRAINT ``
```

```
FOREIGN KEY (`produto_id`)
REFERENCES `mydb`.`Produto` (`produto_id`));
CREATE TABLE IF NOT EXISTS `mydb`.`Pedido` (
`pedido_id` INT NOT NULL,
`data_pedido` TIMESTAMP NULL,
`cliente_nome` VARCHAR(45) NULL,
`cliente_endereco` TEXT NULL,
`total` DECIMAL(10,2) NULL,
PRIMARY KEY (`pedido_id`))
CREATE TABLE IF NOT EXISTS `mydb`.`DetalhesPedido` (
`produto_id` INT NOT NULL,
`pedido_id` INT NOT NULL,
`quantidade` INT NULL,
`preco_unitario` DECIMAL(10,2) NULL,
PRIMARY KEY (`produto_id`, `pedido_id`),
INDEX `fk_table1_Pedido1_idx` (`pedido_id` ASC)
VISIBLE,
CONSTRAINT `fk_table1_Produto1`
FOREIGN KEY (`produto_id`)
REFERENCES `mydb`.`Produto` (`produto_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_table1_Pedido1`
FOREIGN KEY (`pedido_id`)
REFERENCES `mydb`.`Pedido` (`pedido_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
```

FONTE: O AUTOR (2025)

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

Esta disciplina focou na qualidade do código como pilar do desenvolvimento ágil. O projeto de refatoração do algoritmo *Bubble Sort*, um algoritmo de ordenação simples amplamente apresentado em livros introdutórios de estruturas de dados (Cormen *et al.*, 2012), foi um exercício prático para aplicar os princípios de *Clean Code* (Martin, 2008). O desafio central foi transformar um algoritmo complexo em código legível e autoexplicativo, reduzindo sua complexidade ciclomática por meio da extração de métodos e da escolha de nomes significativos.

O *feedback* recebido foi crucial: a verdadeira refatoração vai além de remover comentários, exigindo uma reestruturação que torne o código claro por si só. Esta lição foi aplicada imediatamente nas disciplinas de WEB I e II e MOB I e II, onde a manutenibilidade do código se mostrou essencial para iterações ágeis e sustentáveis.

7.1 ARTEFATOS DO PROJETO

CÓDIGO 2 - CÓDIGO DO BUBBLE SORT REFATORADO

```java
// Implementação otimizada do algoritmo Bubble Sort em Java
// Autor: Mateus Maidel Alves da Silva

/*
 * Melhorias implementadas:
 * 1. Melhoria nas nomenclaturas: Variáveis e funções foram renomeadas para melhorar a legibilidade e compreensão do código.
 * 2. Criação de uma função dedicada para trocar as posições do array: Isso melhora a modularidade e facilita a manutenção do código.
 * 3. Redução de parâmetros nas funções: A função `swapArrayPositions` utiliza a posição como parâmetro em vez de passar múltiplos parâmetros, simplificando a interface da função.
 * 4. Remoção de comentários desnecessários: Comentários redundantes ou que explicavam o óbvio foram removidos para manter o código limpo.
 * 5. Melhora da indentação: A indentação foi revisada para garantir que o código esteja organizado e fácil de ler.
 * 6. Simplificação do loop de impressão do array: O loop `for` foi simplificado para tornar o código mais conciso.
 * 7. Reposicionamento de funções para seguir a sequência lógica: As funções foram reorganizadas para aparecerem na ordem em que são chamadas, facilitando a leitura e entendimento do fluxo do código.
 */

import java.io.*;

class BubbleSort {
    public static void main(String args[]) {
        int array[] = { 64, 34, 25, 12, 22, 11, 90 };
        bubbleSort(array);
        System.out.println("Array ordenado: ");
        printArray(array);
    }

    static void bubbleSort(int array[]) {
        int arraySize = array.length;
        boolean swapped;
        for (int i = 0; i < arraySize - 1; i++) {
            swapped = false;
            for (int j = 0; j < arraySize - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    swapArrayPositions(array, j);
                    swapped = true;
                }
            }
            if (!swapped)
                break;
        }
    }

    static void swapArrayPositions(int array[], int arrayPosition) {
        int temp = array[arrayPosition];
        array[arrayPosition] = array[arrayPosition + 1];
        array[arrayPosition + 1] = temp;
    }
```

```
    static void printArray(int[] array) {
        for (int value : array) {
            System.out.print(value + " ");
        }
        System.out.println();
    }
}
```

FONTE: O AUTOR (2025)

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

Dada minha experiência anterior com desenvolvimento web na graduação, optei por focar nos questionários teóricos desta disciplina, uma modalidade permitida pela coordenação. Esta escolha permitiu consolidar os conceitos de arquitetura web, comunicação entre *front-end* e *back-end* e padrões de persistência de dados sem a sobrecarga de repetir projetos práticos.

O conteúdo abordado, no entanto, foi fundamental para contextualizar as práticas ágeis no desenvolvimento *full-stack*. Os conceitos de criação de APIs com Spring Boot (VMware, 2023), integração com bancos de dados como o PostgreSQL (PostgreSQL, 2023) e o desenvolvimento de interfaces responsivas reforçaram como uma arquitetura bem planejada é crucial para permitir iterações rápidas, entregas contínuas e a adaptabilidade que o desenvolvimento ágil exige.

8.1 ARTEFATOS DO PROJETO

Nenhuma imagem ou projeto foi realizado para esta disciplina.

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

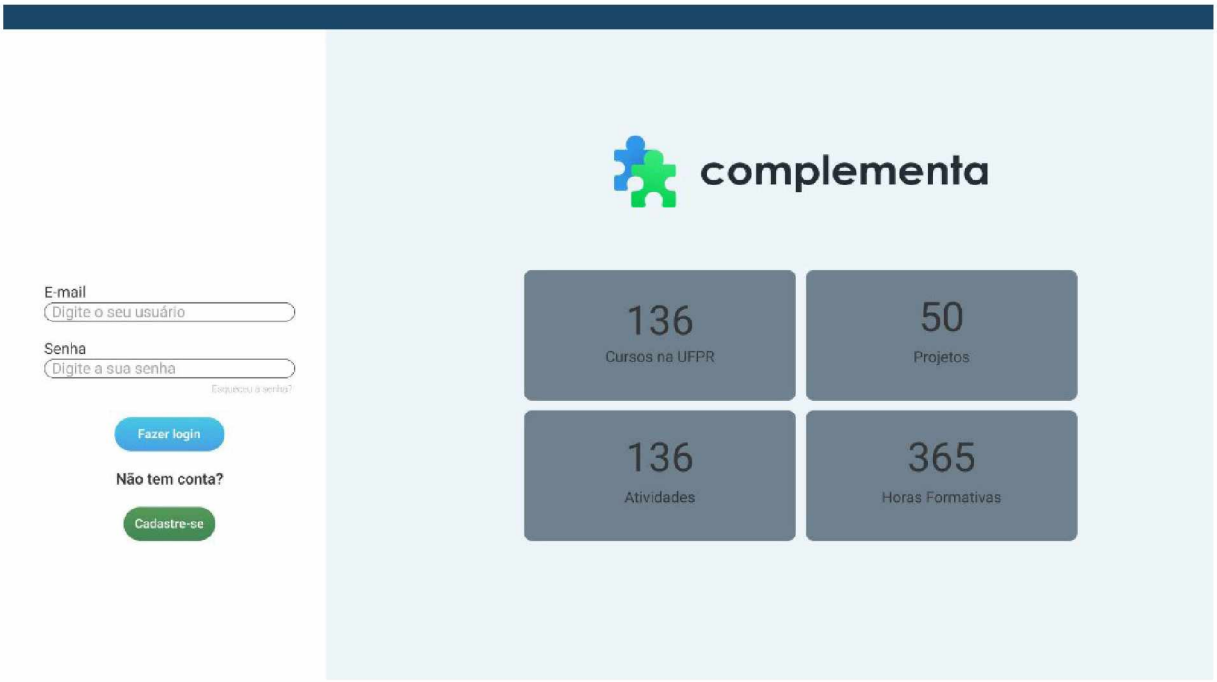
O projeto desta disciplina foi o desenvolvimento de um protótipo de alta fidelidade no Figma para um aplicativo acadêmico, batizado de Complementa UFPR. A proposta era inovadora: uma plataforma que funciona como um *Tinder de projetos*, conectando alunos que precisam de ajuda em seus trabalhos com monitores dispostos a auxiliar em troca de horas formativas (Figma, 2024).

O processo de design priorizou a clareza e acessibilidade, com uma paleta de cores neutra e profissional, *layout* limpo inspirado em sistemas já conhecidos pelos alunos (como o SIGA) e tipografia legível. O *feedback* de usuários colhido durante o processo validou a intuição da interface, apontando-a como interessante, prática e bem organizada, sugerindo apenas a adição de um tutorial para *onboarding* (Nielsen, 1994), (Lidwell; Holden; Butler, 2010).

A professora destacou que as escolhas foram bem feitas, reforçando a importância do design centrado no usuário. Este projeto evidenciou como a prototipação ágil e a iteração contínua baseada em *feedback* são fundamentais para criar produtos que não apenas funcionam, mas que são verdadeiramente úteis e adotados com entusiasmo pelos usuários finais.

9.1 ARTEFATOS DO PROJETO

FIGURA 12 - TELA DE LOGIN

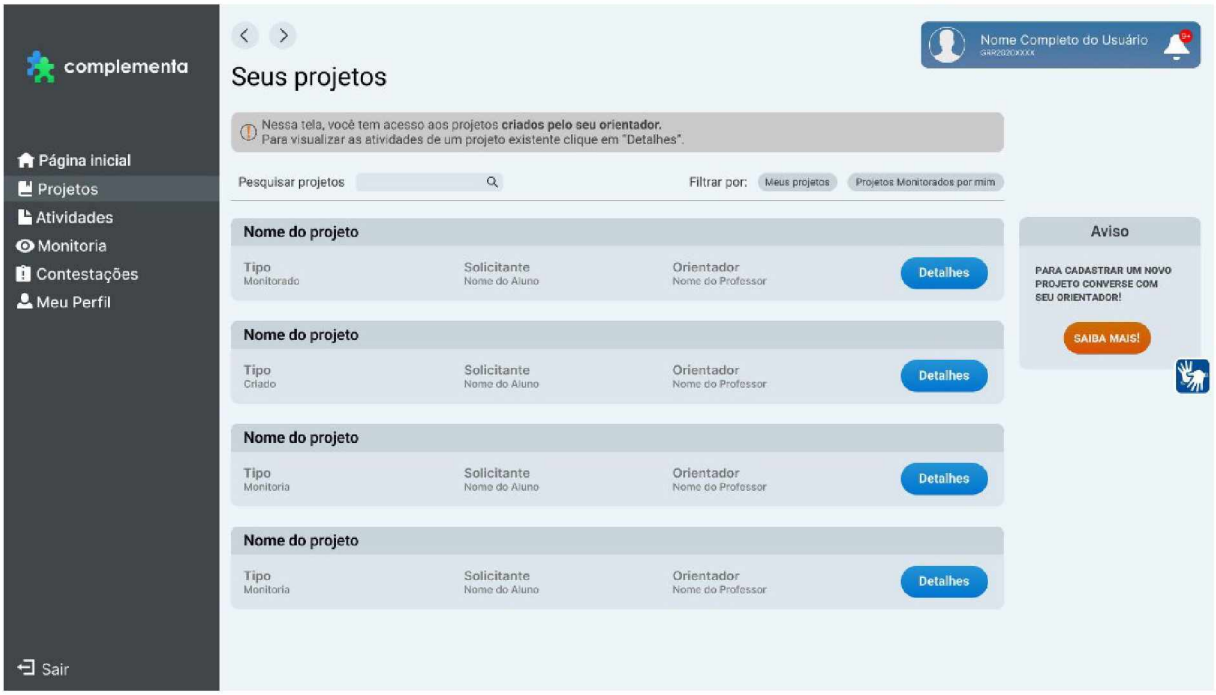


FONTE: O AUTOR (2025)

FIGURA 13 - TELA INICIAL

FONTE: O AUTOR (2025)

FIGURA 14 - TELA DE PROJETOS



FONTE: O AUTOR (2025)

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

As disciplinas de Desenvolvimento Mobile 1 e 2 tiveram como objetivo central a compreensão dos princípios e desafios específicos do desenvolvimento de software para plataformas móveis, sempre sob a ótica das metodologias ágeis.

Dada a familiaridade prévia com o tema e a modalidade alternativa de avaliação permitida pela coordenação, a participação concentrou-se na realização dos questionários teóricos. Este foco permitiu um estudo aprofundado de conceitos fundamentais, como as arquiteturas MVC e MVVM, amplamente utilizadas para organizar responsabilidades e favorecer manutenibilidade do código (Larman, 2016) (Freeman, 2020). Também foram estudados o ciclo de vida de componentes Android, essencial para compreender como atividades e fragments transitam entre estados de criação, execução, pausa e finalização ao longo do uso do aplicativo (Android, 2024), bem como o consumo eficiente de APIs RESTful, com destaque para o uso do Retrofit, que promove comunicação segura, tipada e orientada a boas práticas de rede (Square, 2024) (Fielding, 2000). Adicionalmente, investigaram-se as particularidades da experiência do usuário em dispositivos móveis, considerando limitações de tela, interação por toque e responsividade contínua.

A importância deste conhecimento para o desenvolvimento ágil é inquestionável. A natureza iterativa do mobile, com suas frequentes atualizações e a necessidade de feedback contínuo dos usuários finais, se alinha diretamente aos princípios do Manifesto Ágil, que enfatiza adaptação, colaboração e entrega contínua de valor (Beck *et al.*, 2001). Compreender padrões de projeto voltados para testabilidade e manutenibilidade do código é essencial para permitir rápidas iterações dentro de ciclos de desenvolvimento curtos, reforçando a ideia de evolução contínua.

A integração deste conhecimento com as demais disciplinas do curso é direta. Os princípios de código limpo e responsabilidade clara entre componentes, abordados em AAP, fundamentam a escrita de aplicações móveis sustentáveis e de fácil evolução (Martin, 2009). A modelagem de dados, estudada em MAG1/MAG2 e implementada em BD, é crucial para definir e estruturar as informações que o aplicativo irá consumir e manipular. Por fim, os elementos de UX tornam-se ainda mais críticos no contexto mobile, onde a interface, a fluidez e a experiência do

usuário são fundamentais para o sucesso da aplicação. Dessa forma, mesmo sem o desenvolvimento prático, o embasamento teórico obtido foi essencial para consolidar uma visão integrada e ágil do desenvolvimento de software.

10.1 ARTEFATOS DO PROJETO

Nenhuma imagem ou projeto foi realizado para esta disciplina.

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

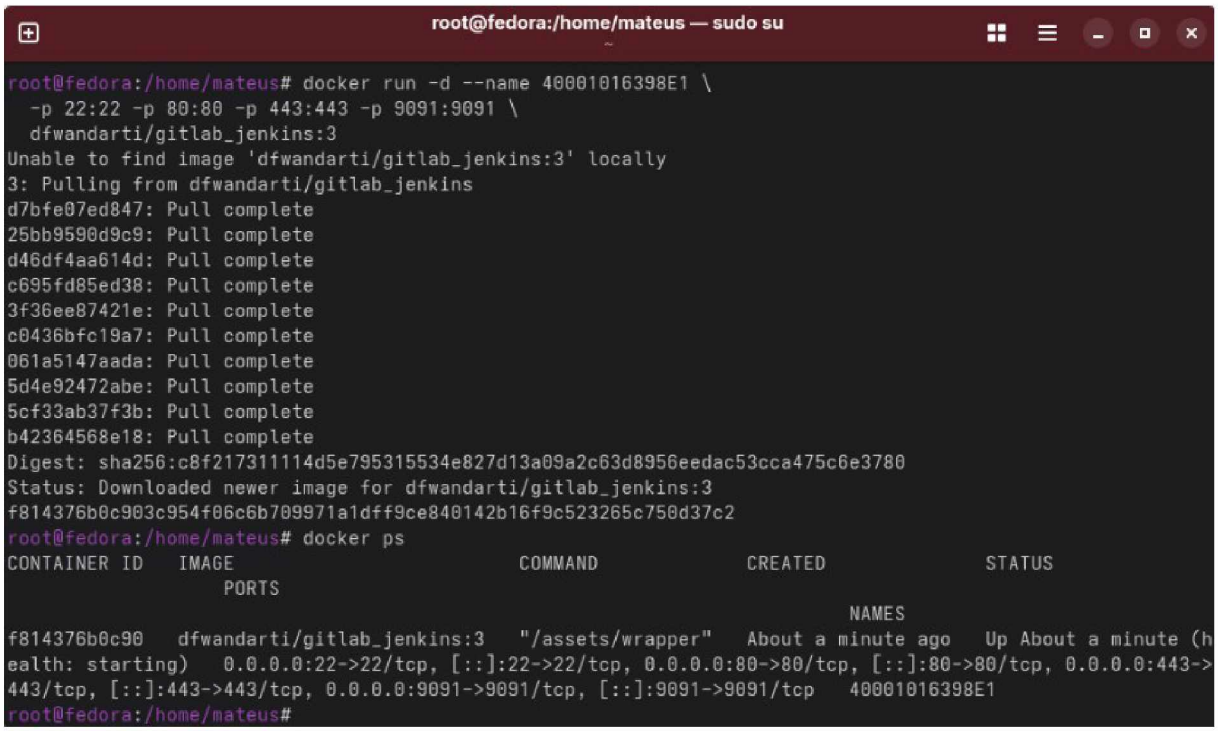
A disciplina de INFRA focou na prática essencial de DevOps, integrando desenvolvimento e operações para dar agilidade ao ciclo de vida do software. O projeto consistiu na configuração de um ambiente completo utilizando Docker, GitLab e Git, onde foi necessário criar um container, acessar o GitLab via navegador e executar comandos de versionamento (Docker, 2024) (GitLab, 2024) (Chacon; Straub, 2014).

A importância deste projeto para o desenvolvimento ágil é fundamental. A capacidade de conteinerizar ambientes com Docker garante consistência entre desenvolvimento e produção, eliminando o problema conhecido como discrepância entre o ambiente local e o ambiente de implantação (Merkel, 2014). O versionamento eficiente com Git (Chacon; Straub, 2014) e a integração contínua promovida pelo GitLab (GitLab, 2024) permitem que múltiplos desenvolvedores trabalhem em paralelo, com entregas frequentes, rastreáveis e automatizadas. Esses elementos reforçam princípios centrais da agilidade, como colaboração, adaptação constante e entrega contínua de valor.

A integração com as demais disciplinas é evidente: o ambiente Docker configurado aqui seria a base para executar os sistemas desenvolvidos em WEB I e II e MOB I e II; os comandos Git praticados são essenciais para o trabalho colaborativo em qualquer projeto de programação; e a cultura DevOps reforça a importância do código limpo de AAP e dos testes automatizados de TEST.

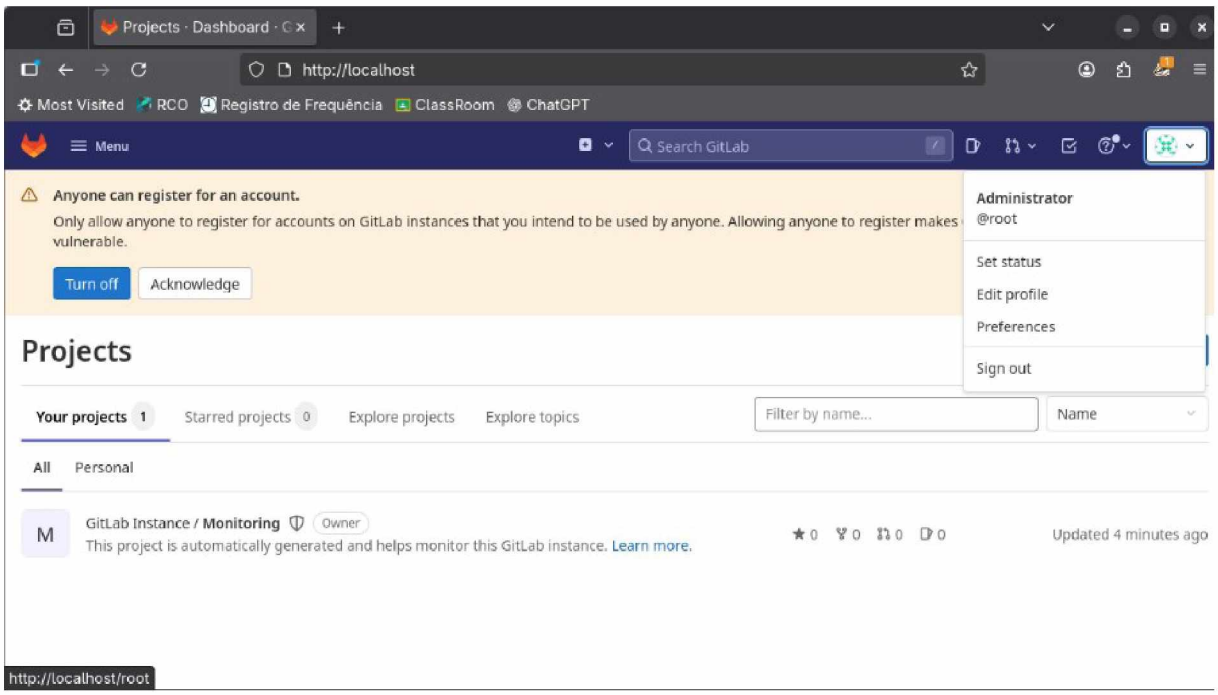
11.1 ARTEFATOS DO PROJETO

FIGURA 15 - CONTAINER EM EXCECUÇÃO



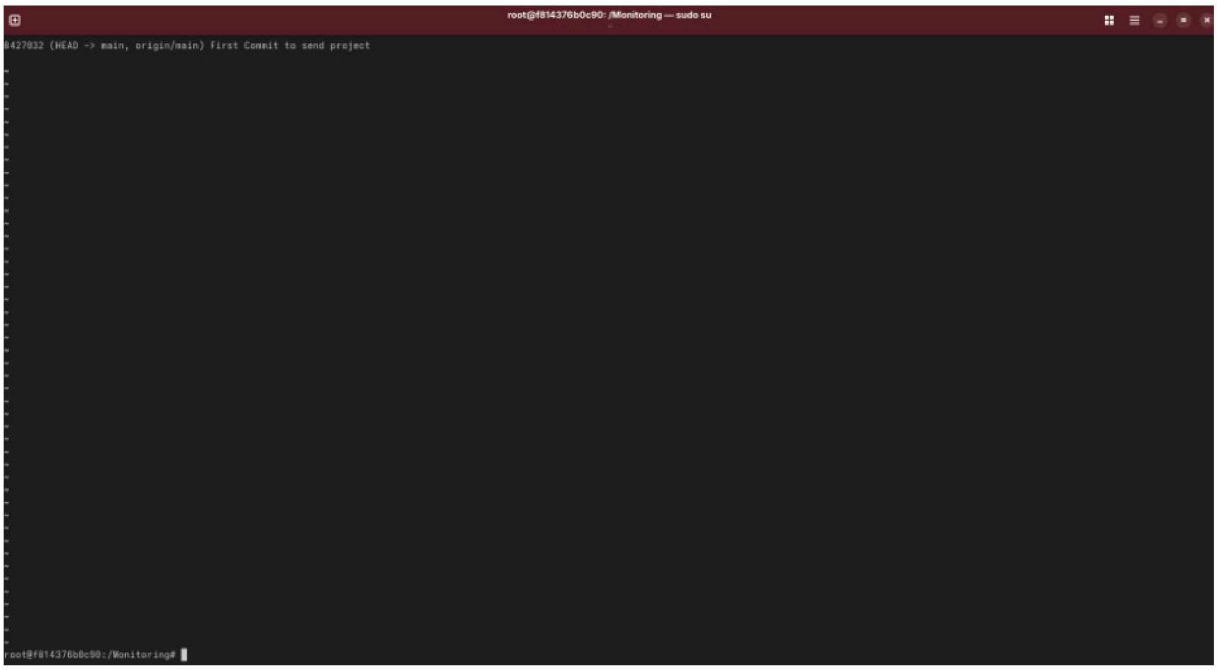
FONTE: O AUTOR (2025)

FIGURA 16 - BROWSER



FONTE: O AUTOR (2025)

FIGURA 17 - LOG DO COMMIT



FONTE: O AUTOR (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados aprofundou-se na aplicação de técnicas e ferramentas para a validação contínua da qualidade do software, enfatizando a automação como um pilar essencial do desenvolvimento ágil. O projeto prático consistiu na criação de um teste *end-to-end* utilizando a ferramenta *Playwright*, no qual foi automatizado o processo de acesso ao site Anotepad.com, preenchimento de um formulário com título, nome e matrícula do aluno e a verificação do comportamento da aplicação após a submissão dos dados (Anotepad, 2025) (Microsoft, 2024).

O Playwright é uma ferramenta moderna de automação de testes mantida pela Microsoft, que oferece suporte nativo a múltiplos navegadores (como Chromium, Firefox e WebKit) e possibilita a execução de testes de interface de forma confiável e esţável (Microsoft, 2024). Um diferencial importante da ferramenta é sua capacidade de lidar com comportamentos assíncronos e elementos dinâmicos na página, garantindo maior precisão na simulação de cenários reais de uso (Microsoft, 2024). Além disso, sua API de alto nível facilita a criação de scripts legíveis, favorecendo a manutenção e evolução dos testes ao longo do ciclo de desenvolvimento.

A relevância deste projeto para o desenvolvimento ágil é significativa. No contexto da Entrega Contínua, cada nova funcionalidade ou alteração no código deve ser validada rapidamente para evitar a propagação de erros para ambientes de produção. Testes *end-to-end* automatizados garantem que o sistema, como um todo, continue a funcionar conforme o esperado, preservando a experiência do usuário final. Essa capacidade de detecção precoce de falhas reduz custos, acelera o ciclo de *feedback* e aumenta a confiabilidade do *software* entregue (Humble; Farley, 2011) (Fowler, 2006).

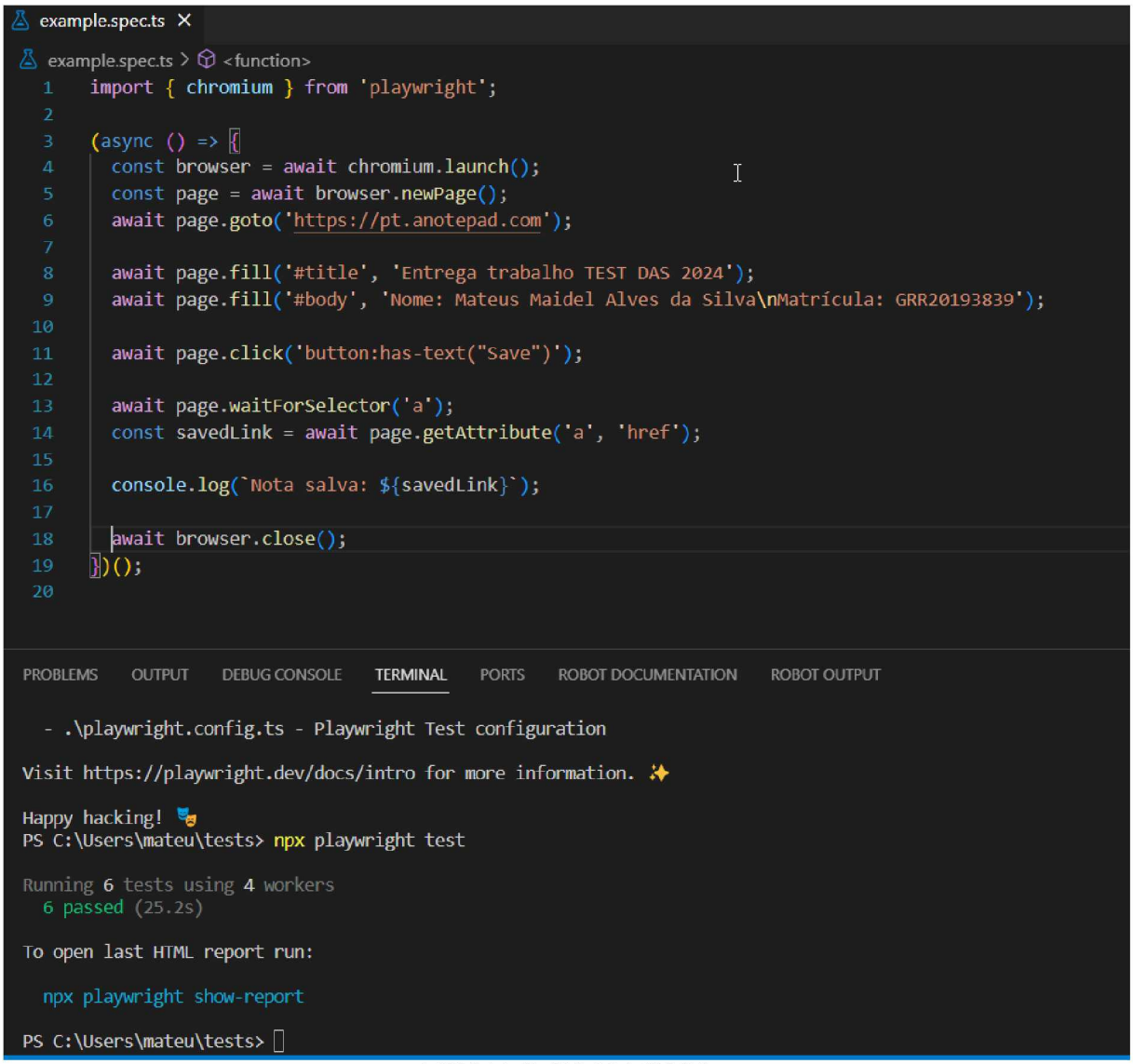
Além disso, a automação de testes se integra diretamente aos processos configurados na disciplina de Infraestrutura e DevOps. Ao inserir os testes de interface em um pipeline de integração contínua (CI), torna-se possível executar validações automáticas a cada *commit*, *merge* ou *deploy*, garantindo uma verificação sistemática e ininterrupta da qualidade. Isso reforça o princípio ágil de entregas

frequentes com segurança, evitando que erros somente sejam percebidos tardiamente em etapas finais do desenvolvimento.

A relação com outras disciplinas do curso também é evidente. Os princípios de clareza e boas práticas de programação estudados em Aspectos Ágeis de Programação (AAP) são fundamentais para criar códigos de teste compreensíveis e sustentáveis. As aplicações desenvolvidas em WEB I e II e MOB I e II tornam-se candidatos naturais para receber estes testes, assegurando que suas interfaces, APIs e fluxos de navegação atendam corretamente às expectativas de uso. Por fim, ao serem incorporados ao pipeline de DevOps configurado em INFRA, os testes automatizados completam um ciclo de desenvolvimento ágil maduro, no qual qualidade, entrega e evolução contínua formam partes integradas de um mesmo processo.

12.1 ARTEFATOS DO PROJETO

FIGURA 18 - CÓDIGO E EXECUÇÃO DO TESTE AUTOMATIZADO COM PLAYWRIGHT



FONTE: O AUTOR (2025)

13 CONCLUSÃO

Este memorial apresentou uma síntese das experiências e aprendizados adquiridos ao longo da Especialização em Desenvolvimento Ágil de Software, evidenciando a aplicação prática do ciclo ágil no desenvolvimento *full-stack*. As disciplinas percorreram todas as etapas do processo de engenharia de software, desde a modelagem e análise de requisitos até a implementação, testes e implantação de sistemas, integrando teoria e prática de forma progressiva.

Os projetos desenvolvidos demonstraram a importância da adaptação contínua, da colaboração e da entrega incremental de valor. A prática do *Test Driven Development* consolidou a qualidade do código desde as fases iniciais. As modelagens em UML e bancos de dados mostraram como o planejamento estruturado sustenta o desenvolvimento ágil. As disciplinas de DevOps e Testes Automatizados reforçaram a necessidade de automação e integração contínua, enquanto os projetos de UX, Web e Mobile destacaram a relevância do foco no usuário e da responsividade.

Durante o percurso, alguns desafios se mostraram evidentes: a dificuldade em equilibrar velocidade e qualidade de entrega, a necessidade de planejamento realista baseado em dados, a importância de comunicação eficaz entre equipes multidisciplinares e a manutenção da clareza e organização do código em ciclos curtos de desenvolvimento.

Em síntese, a experiência permitiu compreender que o desenvolvimento ágil vai além de métodos e ferramentas, constituindo uma mentalidade voltada à melhoria contínua, aprendizado coletivo e entrega de valor real. A especialização consolidou não apenas o domínio técnico, mas também a maturidade profissional necessária para aplicar o ciclo ágil de forma eficaz em projetos *full-stack* e em contextos reais do mercado de tecnologia.

REFERÊNCIAS

ANDERSON, David J. Kanban: **Successful Evolutionary Change for Your Technology Business**. Sequim: Blue Hole Press, 2010.

ANOTEPAD. **Anotepad - online notepad**. Disponível em: https://anotepad.com. Acesso em: 15 out. 2025.

BECK, Kent; *et al.* **Manifesto Ágil para Desenvolvimento de Software**. 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 15 out. 2025.

BECK, Kent. **Extreme Programming Explained: Embrace Change**. 2. ed. Boston: Addison-Wesley, 2004.

BECK, Kent. **Test Driven Development: By Example**. Boston: Addison-Wesley, 2003.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário**. 2. ed. Porto Alegre: Bookman, 2005.

CHACON, Scott; STRAUB, Ben. **Pro Git. 2. ed**. Berkeley: Apress, 2014. Disponível em: https://git-scm.com/book. Acesso em: 15 out. 2025.

COHN, Mike. **User Stories Applied: For Agile Software Development**. Boston: Addison-Wesley, 2004.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Algoritmos: teoria e prática. 3**. ed. São Paulo: LTC, 2012.

DOCKER. **Docker documentation**. Disponível em: https://docs.docker.com/. Acesso em: 15 out. 2025.

FIGMA**.** ***Fig*ma: collaborative interface design tool**. Disponível em: https://www.figma.com. Acesso em: 15 out. 2025.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese (Doutorado) – University of California, Irvine.

GIT. CHACON, S.; STRAUB, B. ***Pro Git.*** 2. ed. Berkeley: Apress, 2014. Disponível em: https://git-scm.com/book/en/. Acesso em: 15 out. 2025.

GITLAB. **GitLab Documentation**. 2024. Disponível em: https://docs.gitlab.com. Acesso em: 15 out. 2025.

GOOGLE. **Android Developers Documentation**. 2024. Disponível em: https://developer.android.com/guide. Acesso em: 15 out. 2025.

HUMBLE, Jez; FARLEY, David. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. Boston: Addison-Wesley, 2010.

KNIBERG, Henrik; SKARIN, Mattias. **Kanban and Scrum: Making the Most of Both**. 1. ed. Lulu Press, 2010.

LARMAN, Craig; VODDE, Bas. **Large-Scale Scrum: More with LeSS**. 1. ed. Boston: Addison-Wesley, 2016.

LIDWELL, W.; HOLDEN, K.; BUTLER, J. **Universal Principles of Design**. Beverly: Rockport Publishers, 2010.

MARTIN, Robert C. Clean Code: **A Handbook of Agile Software Craftsmanship**. Upper Saddle River, NJ: Prentice Hall, 2008.

MERKEL, Dirk. **Docker: Lightweight Linux Containers for Consistent Development and Deployment**. Linux Journal, n. 239, 2014.

MICROSOFT. **Playwright Documentation**. 2024. Disponível em: https://playwright.dev/. Acesso em: 15 out. 2025.

NIELSEN, J. **Usability Engineering**. San Diego: Morgan Kaufmann, 1994.

POPPENDIECK, Mary; POPPENDIECK, Tom. **Lean Software Development: An Agile Toolkit**. Boston: Addison-Wesley, 2003.

POSTGRESQL Global Development Group. **PostgreSQL Documentation**. 2024. Disponível em: https://www.postgresql.org/docs/. Acesso em: 15 out. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**. 2020. Disponível em: https://scrumguides.org. Acesso em: 15 out. 2025.

SELENIUM. **Selenium Documentation**. 2023. Disponível em: https://www.selenium.dev/documentation/. Acesso em: 15 out. 2025.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Operating System Concepts**. 10. ed. Boston: Addison-Wesley, 2019.

SQUARE. **Retrofit Documentation**. 2024. Disponível em: https://square.github.io/retrofit/. Acesso em: 15 out. 2025.

VMWARE. **Spring Boot Reference Documentation**. 2023. Disponível em: https://docs.spring.io/spring-boot/docs/current/reference/html/. Acesso em: 15 out. 2025.