UNIVERSIDADE FEDERAL DO PARANÁ

LUCAS ANDRADE PONTES

MEMORIAL DE PROJETOS: DESENVOLVIMENTO DE SOFTWARE ALIADO ÀS METODOLOGIAS ÁGEIS

CURITIBA

2025

LUCAS ANDRADE PONTES

MEMORIAL DE PROJETOS: DESENVOLVIMENTO DE SOFTWARE ALIADO ÀS METODOLOGIAS ÁGEIS

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: 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 **LUCAS ANDRADE PONTES**, intitulada: **MEMORIAL DE PROJETOS: DESENVOLVIMENTO DE SOFTWARE ALIADO ÀS METODOLOGIAS ÁGEIS**, 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, 10 de Setembro de 2025.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

RAZER ANTHOM NIZER ROJAS MONTAÑO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial tem como objetivo apresentar a trajetória formativa no curso de Especialização em Desenvolvimento Ágil de Software, demonstrando como os conhecimentos adquiridos foram aplicados em projetos práticos que simularam situações reais do mercado de tecnologia. Ao longo das disciplinas, foram desenvolvidos diversos artefatos — como diagramas, protótipos, códigos, planos de release, scripts de banco de dados e testes automatizados — que evidenciam a integração entre teoria e prática sob a ótica ágil. O conteúdo abordado permitiu consolidar competências técnicas e metodológicas relacionadas à modelagem enxuta, desenvolvimento web e mobile, gestão ágil de projetos, testes, UX e DevOps. A interdisciplinaridade das atividades mostrou-se essencial para a construção de uma visão sistêmica e colaborativa, onde a entrega contínua de valor e a adaptação às mudanças são prioridades. O memorial conclui que a formação proporcionada pelo curso capacita o profissional a atuar de forma ágil, inovadora e centrada no usuário em projetos reais de software.

Palavras-chave: Desenvolvimento de Software; Metodologias Ágeis.

ABSTRACT

This report presents the learning journey throughout the Postgraduate Course in Agile Software Development, demonstrating how the acquired knowledge was applied in practical projects that simulated real-world scenarios in the technology industry. Throughout the disciplines, various artifacts were developed — such as diagrams, prototypes, source code, release plans, database scripts, and automated tests — highlighting the integration between theory and practice through an agile perspective. The content covered enabled the consolidation of technical and methodological skills related to lean modeling, web and mobile development, agile project management, testing, UX, and DevOps. The interdisciplinary nature of the activities proved essential in building a systemic and collaborative mindset, where continuous value delivery and adaptability to change are key. The report concludes that the course provided solid training for professionals to act in real software projects with agility, innovation, and a user-centered approach.

Keywords: Software Development; Agile Methodologies.

SUMÁRIO

1 PARECER TÉCNICO

Diante de um mercado cada vez mais exigente, compreender e aplicar os princípios do desenvolvimento ágil de software tornou-se essencial para profissionais que desejam atuar em todas as etapas do ciclo de vida de um sistema, reforçando a importância de práticas pragmáticas na carreira de programadores. Estudos acadêmicos confirmam que abordagens ágeis respondem melhor à volatilidade do mercado de tecnologia (Abrahamsson *et al*., 2002).

Este parecer técnico tem como objetivo demonstrar como a especialização em Desenvolvimento Ágil de Software contribuiu para o aprimoramento de competências práticas, a partir de disciplinas integradas e projetos aplicados — abrangendo desde a modelagem e análise até a entrega de soluções reais.

Ao longo do documento, serão destacados os principais aprendizados, os artefatos desenvolvidos e as metodologias utilizadas, com ênfase na aplicação dos princípios ágeis em diferentes contextos. A proposta é evidenciar como a formação técnica e prática proporcionada pelo curso permitiu consolidar uma visão sistêmica, colaborativa e adaptável, alinhada às demandas atuais do mercado de tecnologia.

Os fundamentos do desenvolvimento ágil foram introduzidos logo nas etapas iniciais, por meio da disciplina de Métodos Ágeis de Desenvolvimento de Software (MADS). Nela, foram abordados modelos de processo, o Manifesto Ágil, Scrum, Extreme Programming, entre outros métodos, estabelecendo as bases para a aplicação da agilidade ao longo do curso (Beck; Andres, 2004).

Em seguida, as disciplinas de Modelagem Ágil de Software I e II (MAG1 e MAG2) proporcionaram uma abordagem prática e enxuta da modelagem, por meio da utilização de diagramas da *Unified Modeling Language* (UML) (OMG, 2017) e técnicas como histórias de usuário, alinhando o processo de modelagem aos princípios colaborativos descritos por Cockburn (2006).

Complementarmente, as disciplinas de Gerenciamento Ágil de Software 1 e 2 ofereceram uma visão aprofundada sobre gestão de projetos, com foco em abordagens como *Scrum* (Schwaber; Sutherland, 2020), *Kanban* (Anderson, 2010) e *Lean Inception* (Caroli, 2018). Os artefatos, papéis e cerimônias foram estudados de forma aplicada, com base nas boas práticas do mercado e estudos de caso.

Na sequência, a disciplina de *User Experience* (UX) no Desenvolvimento Ágil de Software introduziu uma abordagem centrada no usuário, integrando princípios de design fluido às iterações rápidas, com foco na entrega contínua de valor ao cliente.

Paralelamente, a disciplina de Banco de Dados aprofundou o conhecimento sobre organização e manipulação de dados, garantindo integridade e desempenho adequados às soluções desenvolvidas.

Para embasar a parte técnica do desenvolvimento, a disciplina de Introdução à Programação apresentou os fundamentos da orientação a objetos em Java, práticas diretamente relacionadas aos padrões de projeto e princípios SOLID (*Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation e Dependency Inversion*) abordados por Martin (2002). A As disciplinas de Desenvolvimento Web I e II ampliaram a compreensão sobre o desenvolvimento full-stack, enquanto Desenvolvimento Mobile I e II exploraram *Flutter* (Google, 2025c) e *Firebase* (Google, 2025b), promovendo a construção de aplicações modernas e escaláveis.

A disciplina de Aspectos Ágeis de Programação destacou práticas voltadas à qualidade e manutenibilidade do código, fundamentadas nas técnicas de refatoração sistematizadas por Fowler (2018). Em complemento, a disciplina de Infraestrutura para Desenvolvimento e Implantação de Software (DevOps) abordou práticas de automação, integração e entrega contínua, essenciais para ciclos ágeis de desenvolvimento.

Por fim, a disciplina de Testes Automatizados consolidou o entendimento sobre qualidade de software, alinhando-se às melhores práticas para testes em equipes ágeis (Crispin; Gregory, 2009).

Em síntese, o curso de Especialização em Desenvolvimento Ágil de Software proporcionou uma formação integrada e prática, na qual análise, modelagem, desenvolvimento, testes e implantação são partes interdependentes de um processo contínuo. A experiência adquirida permitiu o desenvolvimento de uma mentalidade ágil, colaborativa e orientada à entrega de valor — características essenciais para enfrentar os desafios reais do mercado de tecnologia (Abrahamsson *et al*., 2002).

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

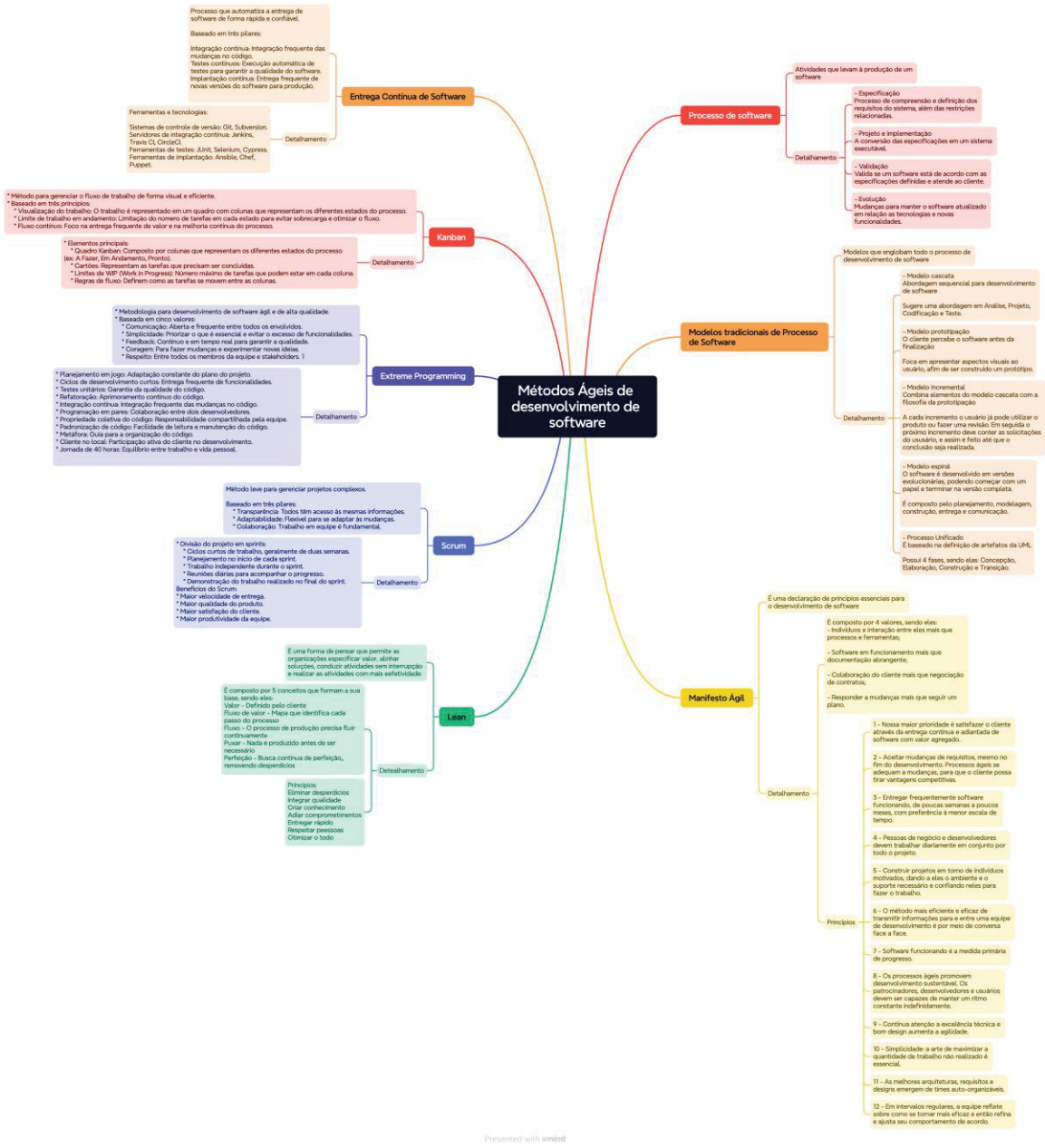
A disciplina de Métodos Ágeis de Desenvolvimento de Software (MADS) teve como principal entrega um mapa mental que sintetiza os conceitos estudados ao longo do curso, com o objetivo de representar os principais processos, princípios e métodos ágeis. O material abordou com mais profundidade tópicos como os modelos tradicionais de processo de *software*, o Manifesto Ágil (BECK *et al.*, 2001), o *Lean Software Development* (Poppendieck; Poppendieck, 2003), o *Scrum* (Schwaber; Sutherland, 2020) e o *Kanban* (Anderson, 2010), entre outros métodos relevantes.

Os conhecimentos adquiridos foram fundamentais para entender e aplicar respostas ágeis às mudanças, algo essencial em projetos de software, nos quais os requisitos evoluem constantemente. O foco no cliente, o trabalho em equipe e a comunicação eficaz complementam os demais conteúdos abordados, promovendo a entrega contínua de valor e fortalecendo a colaboração entre os membros da equipe.

O conteúdo de MADS serviu como alicerce para as disciplinas subsequentes da especialização em Desenvolvimento Ágil de Software na Universidade Federal do Paraná (UFPR), oferecendo os fundamentos práticos e teóricos que justificam plenamente o nome e a proposta do curso.

2.1 ARTEFATOS DO PROJETO

Figura 1 – Mapa mental



FONTE: O autor (2025)

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

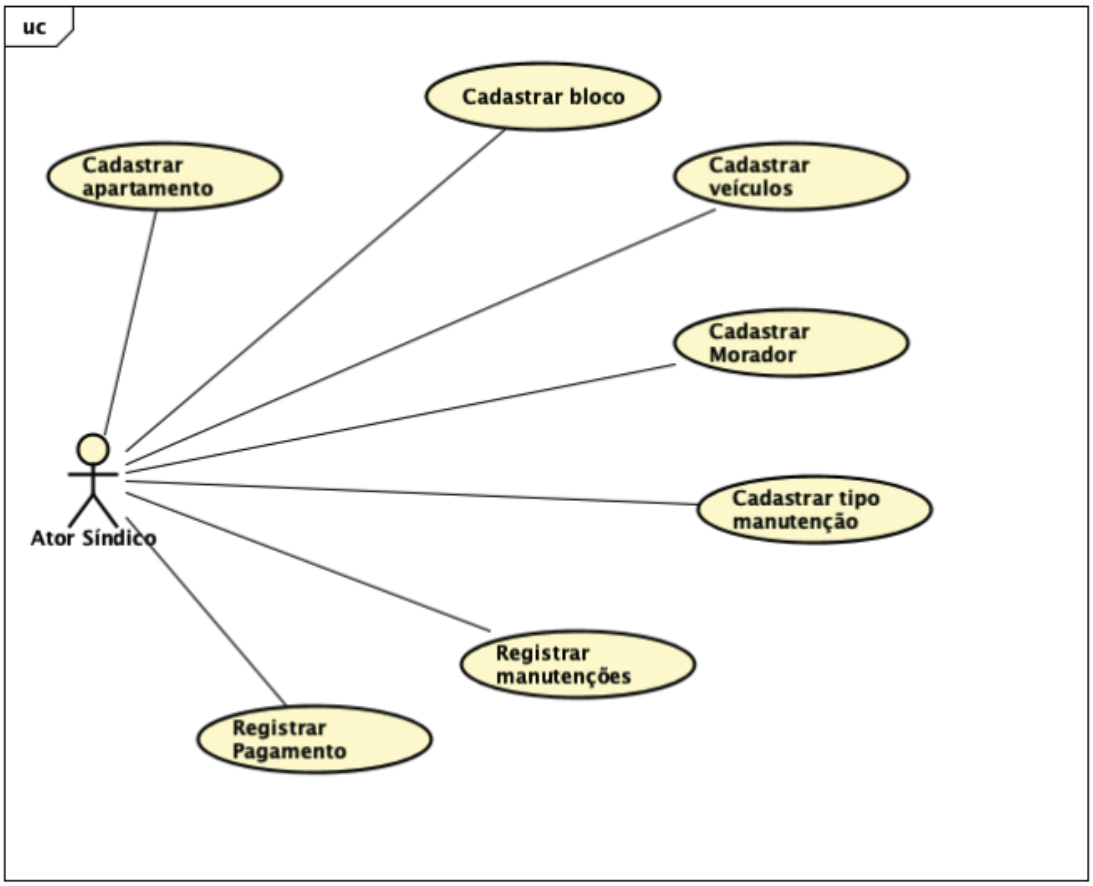
A disciplina de Modelagem Ágil de Software 1 e 2 teve como ponto de partida o estudo dos motivos pelos quais a modelagem deve ser ágil para se alinhar à filosofia dos Métodos Ágeis, com aprofundamento dos conceitos gerais da UML (OMG, 2017), especificação de requisitos, casos de uso, histórias de usuário, além de diversos diagramas como de sequência, atividades, transição de estados, entre outros.

As duas disciplinas tiveram como projeto final o desenvolvimento de artefatos relacionados à modelagem de um Sistema de Gestão de Condomínio, sendo eles: um diagrama de casos de uso (níveis 1 e 2) e as histórias de usuário elaboradas de forma completa. Já em MAG2, o projeto foi ampliado com a inclusão do diagrama de classes, tabelas do banco de dados e diagramas de sequência correspondentes a todas as histórias de usuário.

Esse projeto foi fundamental para a consolidação dos conhecimentos práticos de modelagem no contexto ágil, demonstrando como a documentação visual e objetiva contribui para a comunicação entre os membros da equipe e para a evolução contínua do software. A construção incremental e colaborativa dos modelos reforçou a importância da modelagem como ferramenta de apoio ao desenvolvimento iterativo, permitindo entregas mais rápidas, organizadas e alinhadas às necessidades do produto. Além disso, os artefatos desenvolvidos dialogam diretamente com outras disciplinas do curso, mostrando a integração entre áreas técnicas essenciais no desenvolvimento ágil. Dessa forma, o projeto de MAG1 e MAG2 representou uma aplicação prática e estruturada dos princípios ágeis, conectando teoria, técnica e colaboração no processo de construção de software.

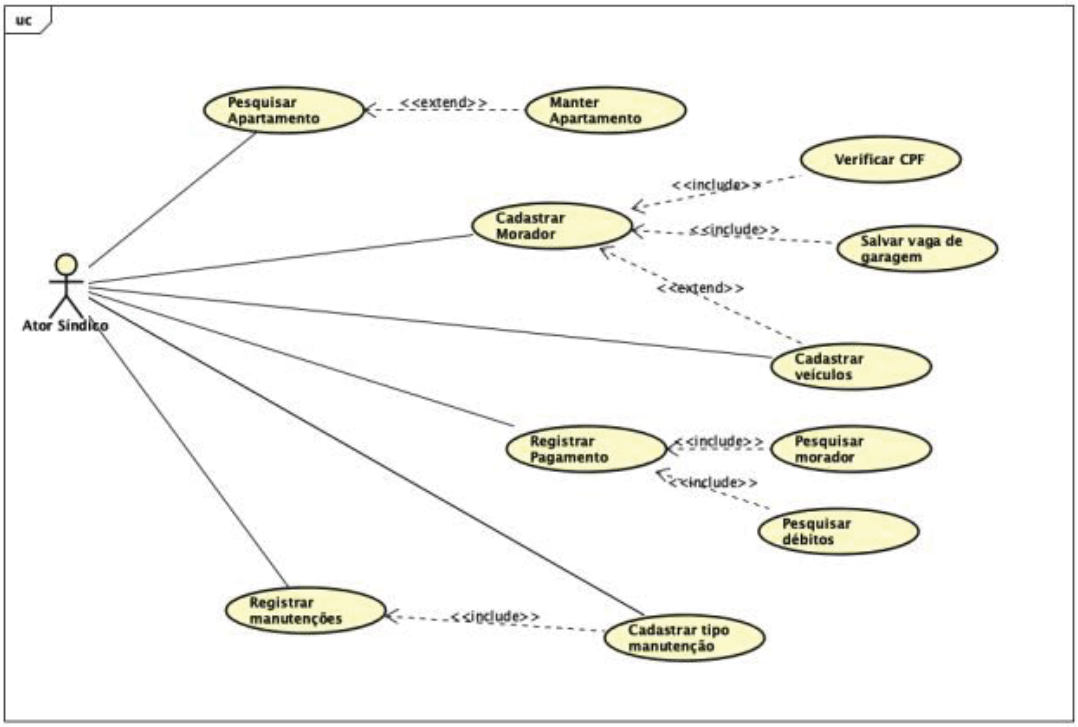
3.1 ARTEFATOS DO PROJETO - MAG1

Figura 2 - Diagrama de casos de uso – Nível 1



FONTE: O autor (2025)

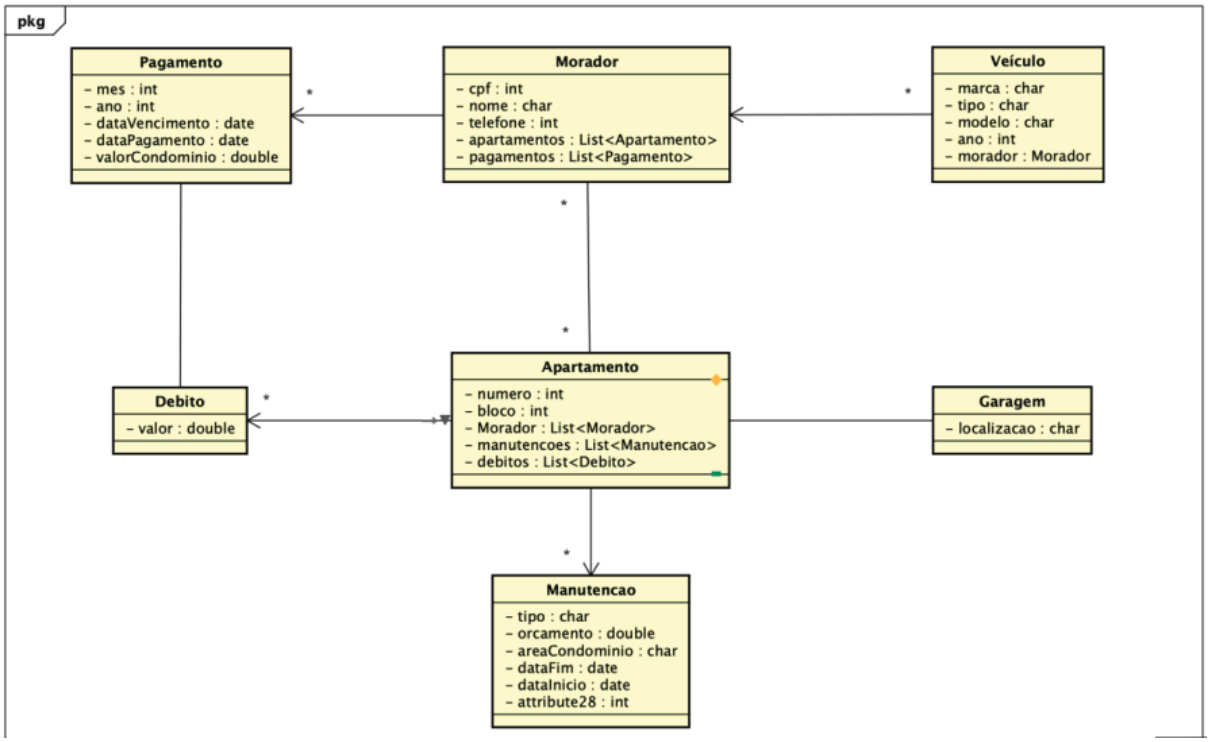
Figura 3 - Diagrama de casos de uso - Nível 2



FONTE: O autor (2025)

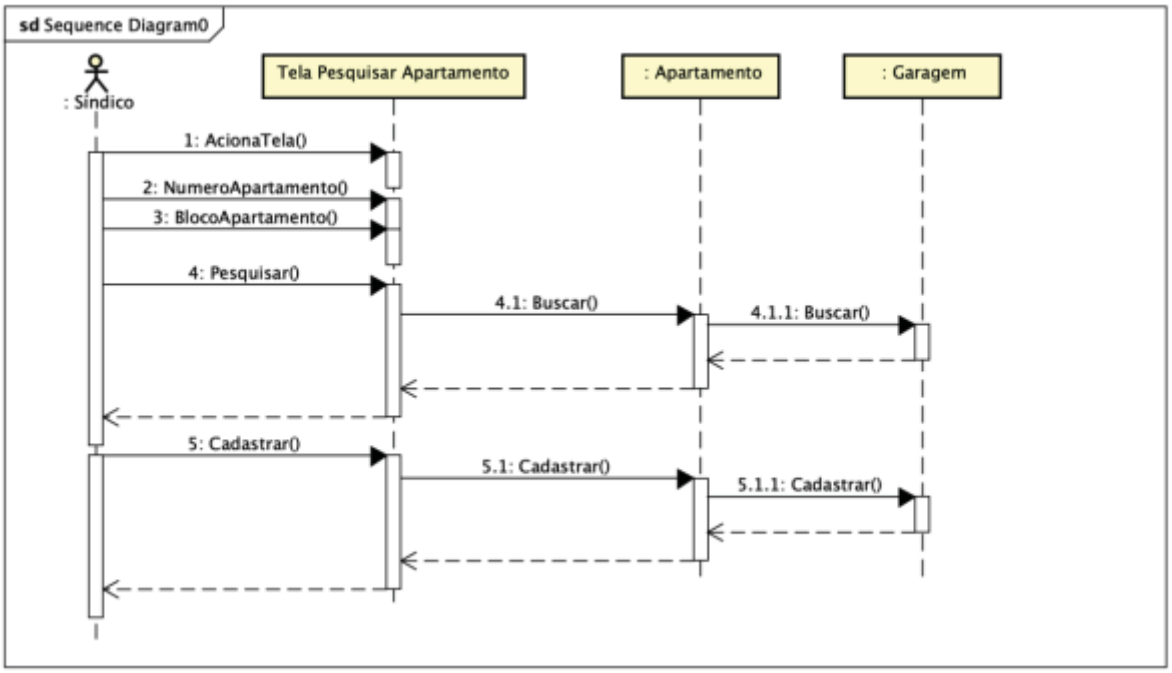
3.2 ARTEFATOS DO PROJETO – MAG2

Figura 4 - Diagrama de classes



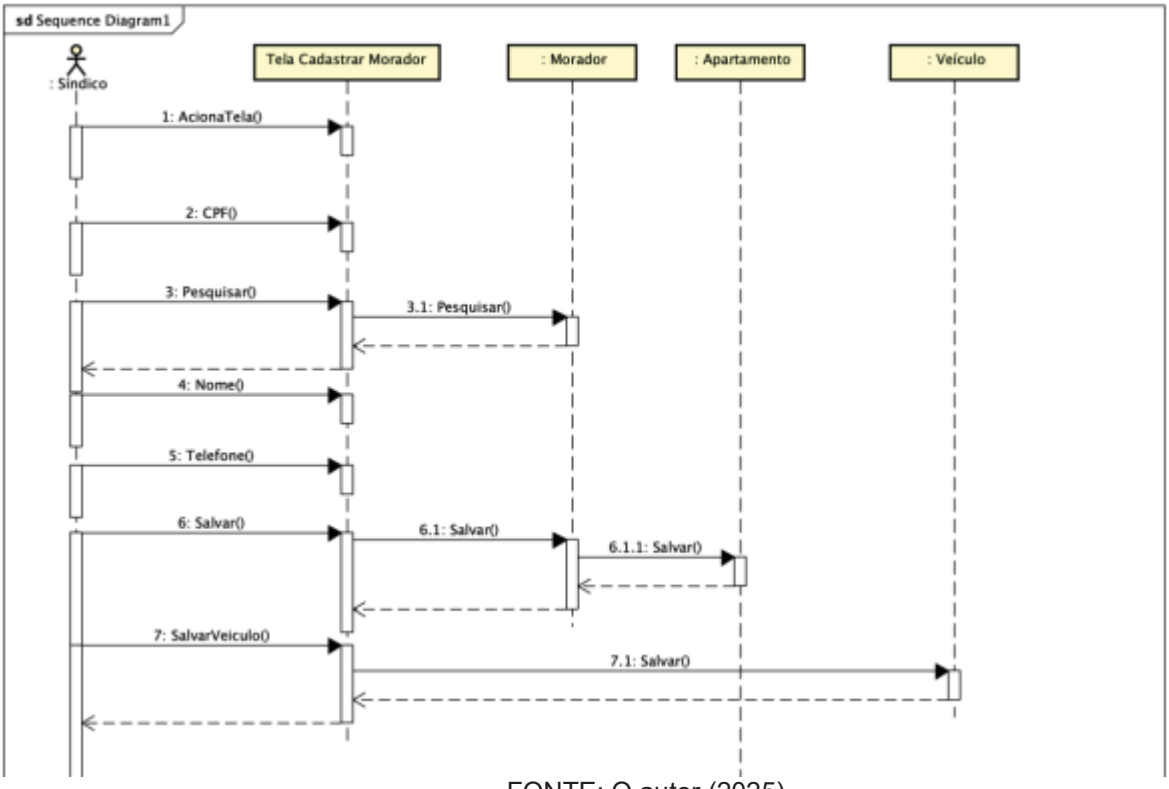
FONTE: O autor (2025)

Figura 5 - Diagramas de sequência 1



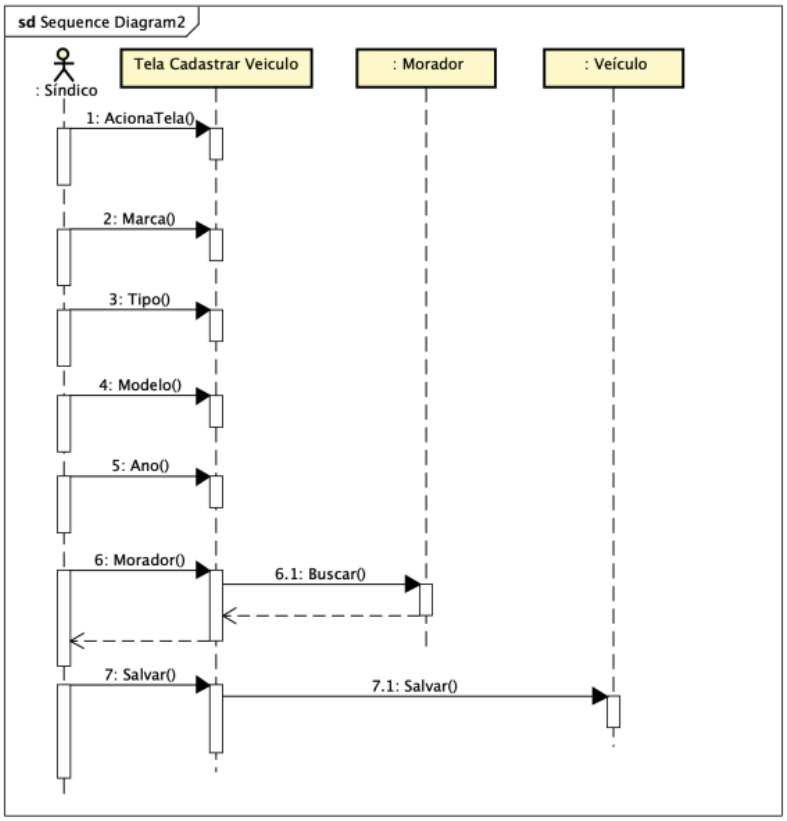
FONTE: O autor (2025)

Figura 6 – Diagramas de sequência 2



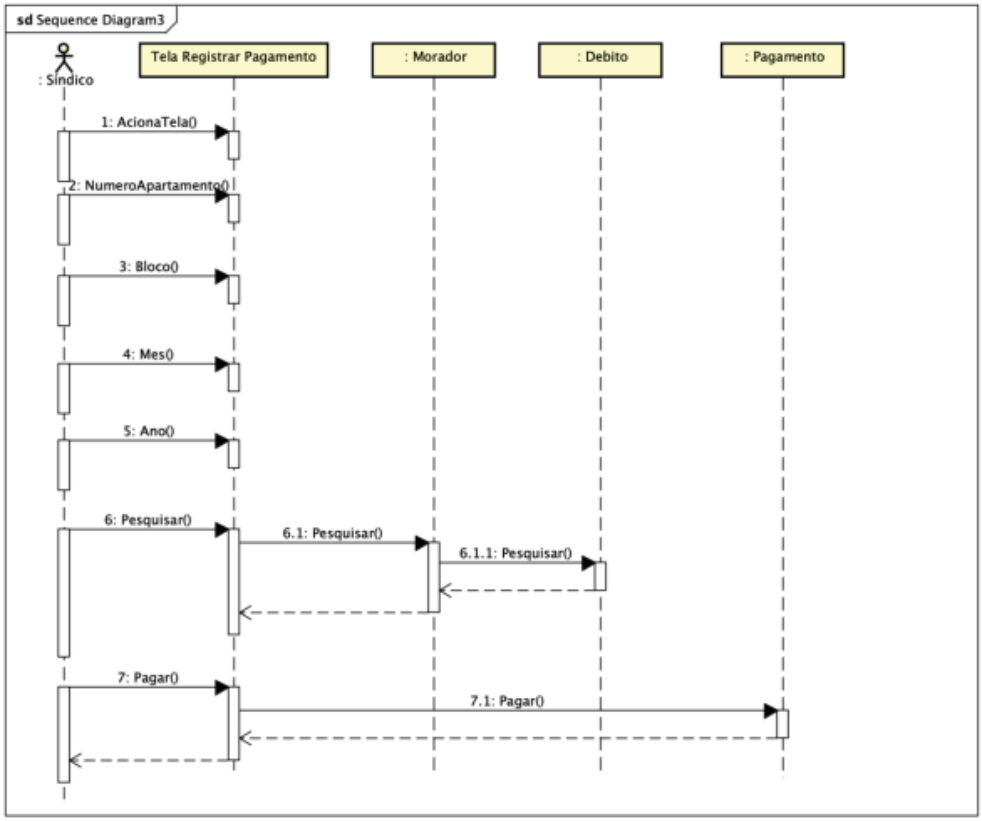
FONTE: O autor (2025)

Figura 7 – Diagramas de sequência 3



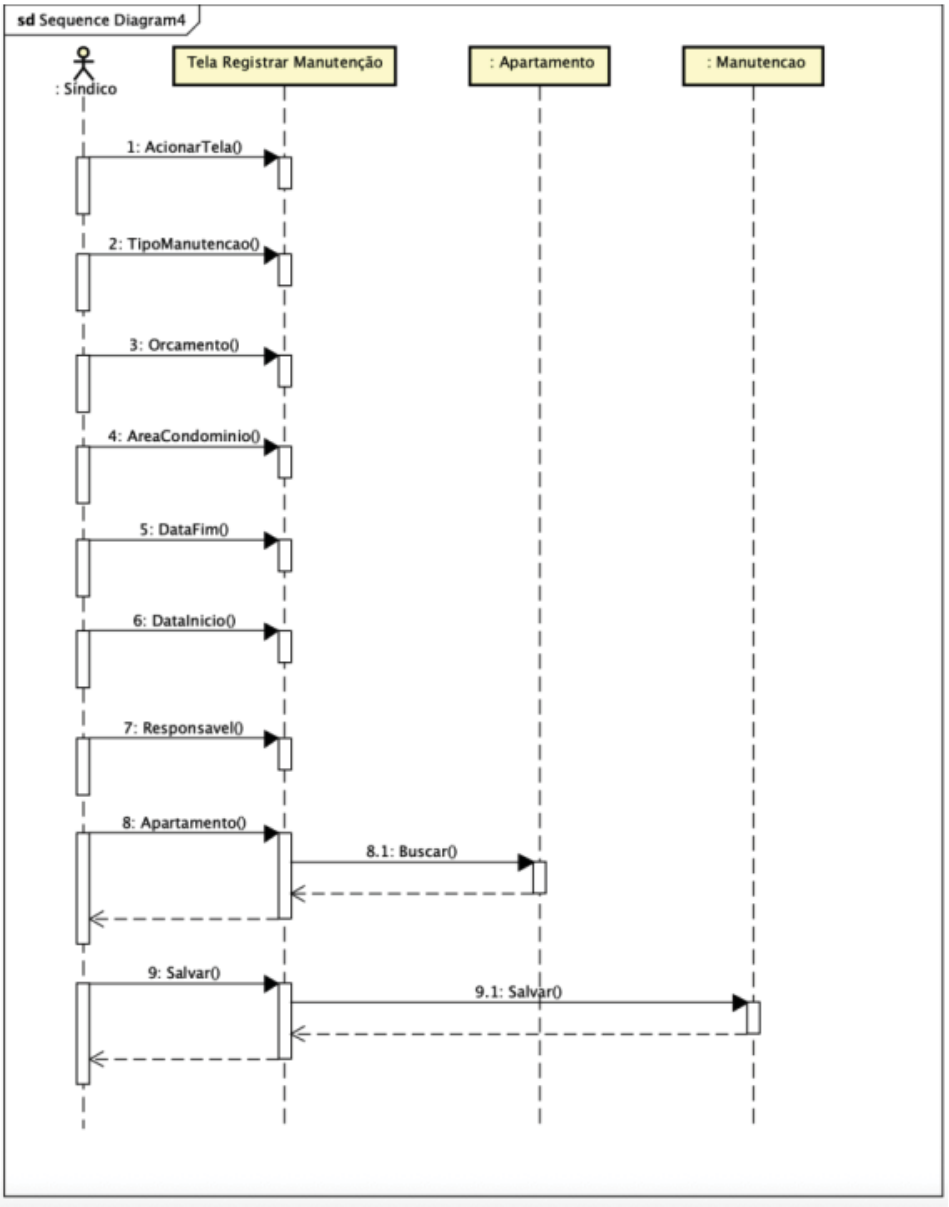
FONTE: O autor (2025)

Figura 8 – Diagramas de sequência 4



FONTE: O autor (2025)

Figura 9 – Diagramas de sequência 5



FONTE: O autor (2025)

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

As disciplinas de Gerenciamento Ágil de Projetos de Software 1 e 2 abordaram inicialmente os principais conceitos de gestão de projetos, com ênfase nas particularidades dos projetos de *software*. Foram estudadas abordagens tradicionais, como o *Project Management Body of Knowledge* (PMBOK) (PMI, 2017), e metodologias ágeis, como Scrum (Schwaber; Sutherland, 2020) e *Kanban* (Anderson, 2010), além de métricas aplicadas ao acompanhamento e à melhoria contínua de processos de desenvolvimento.

Em GAP1, o projeto consistiu na elaboração de um plano de *release* completo para o desenvolvimento de um software idealizado individualmente. O trabalho exigiu o cálculo da velocidade com base em pontos de história (sendo 1 ponto equivalente a 8 horas de trabalho) e a definição de todas as *sprints* com datas e histórias de usuário associadas. Essa atividade foi essencial para aplicar, de forma prática, os conceitos de planejamento ágil e estimativa de esforço, além de evidenciar a importância da organização incremental das entregas e da priorização do backlog com base em valor e viabilidade.

Já em GAP2, o foco foi a aplicação prática da gestão por fluxo contínuo, por meio da simulação de um ciclo de 35 dias no jogo online *Kanban Board Game* (2025). A atividade permitiu vivenciar os desafios de uma gestão orientada ao fluxo de trabalho, onde foi necessário tomar decisões estratégicas sobre alocação de recursos, limites de WIP (*Work In Progress*), resposta a gargalos e busca pela eficiência operacional. Os registros das etapas do jogo, com *prints* e geração do CFD (*Cumulative Flow Diagram)* (Anderson, 2010), proporcionaram uma análise concreta da evolução do time fictício e da receita obtida ao final do ciclo.

Ambos os projetos foram fundamentais para consolidar o entendimento de como a gestão ágil se aplica na prática, seja por meio de planejamento iterativo (como no *Scrum*) ou do fluxo contínuo (como no *Kanban*). Além disso, esses trabalhos dialogam diretamente com outras disciplinas do curso, reforçando a integração entre planejamento, execução e entrega contínua de valor. Assim, GAP1 e GAP2 demonstraram, de forma aplicada, a importância do gerenciamento ágil como elemento central no sucesso de projetos de software.

4.1 ARTEFATOS DO PROJETO - GAP1

Figura 10 – Plano de release

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

Cálculo da Velocidade:

Horas disponíveis por dia:	8	Tamanho da Sprint:	2 semanas
Horas disponíveis por Sprint:	80	Velocidade:	10

Plano de Release:

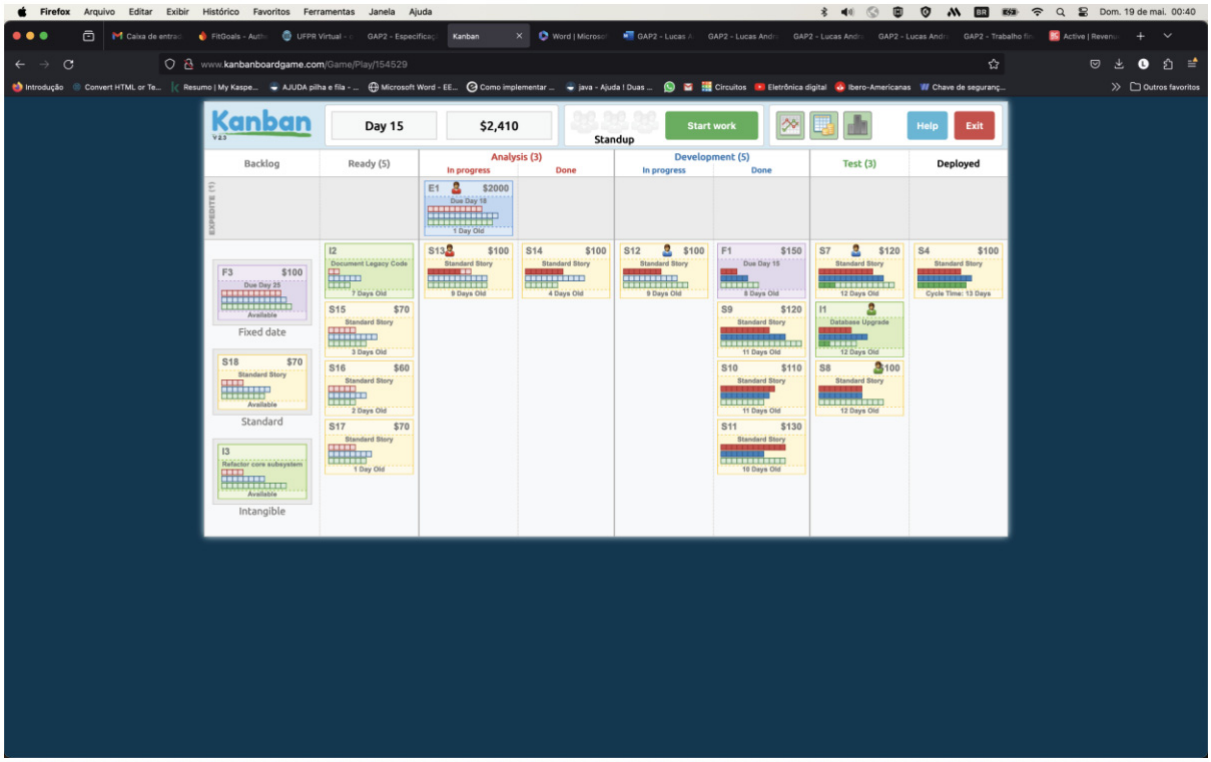
Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3
Data Início: 03/06/2024	Data Início: 17/06/2024	Data Início: 01/07/2024
Data Fim: 14/06/2024	Data Fim: 28/06/2024	Data Fim: 12/07/2024
HU001 **SENDO** um Síndico que deseja adicionar informações de novos apartamentos **QUERO** registrar o apartamento e bloco correspondente **PARA** manter um registro organizado de todos os apartamentos **ESTIMATIVA (5)**	**HU003** **SENDO** um administrador ou funcionário autorizado do condomínio **QUERO** poder cadastrar os veículos associados aos moradores já cadastrados **PARA** manter um registro atualizado dos veículos dos moradores e facilitar a gestão da administração do condomínio **ESTIMATIVA (3)**	**HU005** **SENDO** um funcionário encarregado da administração do condomínio **QUERO** poder registrar as manutenções realizadas no prédio, como pintura, limpeza de caixa d'água, jardinagem, etc. **PARA** manter um registro organizado das atividades de manutenção e garantir a boa conservação do condomínio **ESTIMATIVA (5)**
HU002 **SENDO** um administrador ou funcionário do condomínio **QUERO** poder cadastrar novos moradores fornecendo suas informações básicas, incluindo CPF, nome, telefone, número do apartamento, se é responsável (S/N) e se é proprietário (S/N), **PARA** manter um registro atualizado dos residentes do condomínio e facilitar a gestão da administração. **ESTIMATIVA (5)**	**HU004** **SENDO** um funcionário autorizado do condomínio **QUERO** poder registrar os pagamentos do condomínio feitos pelos moradores **PARA** manter um registro preciso dos pagamentos realizados e auxiliar na gestão financeira do condomínio **ESTIMATIVA (5)**	

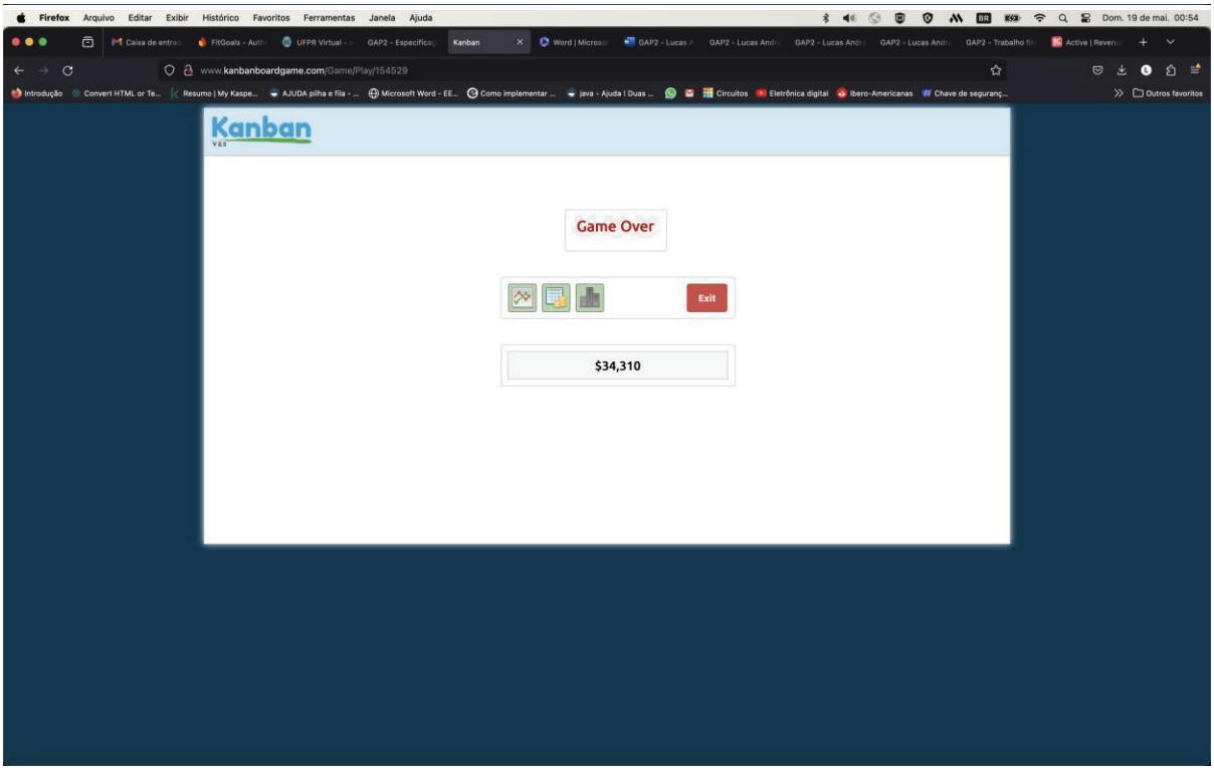
Aluno: Lucas Andrade Pontes

FONTE: O autor (2025)

4.2 ARTEFATOS DO PROJETO – GAP2

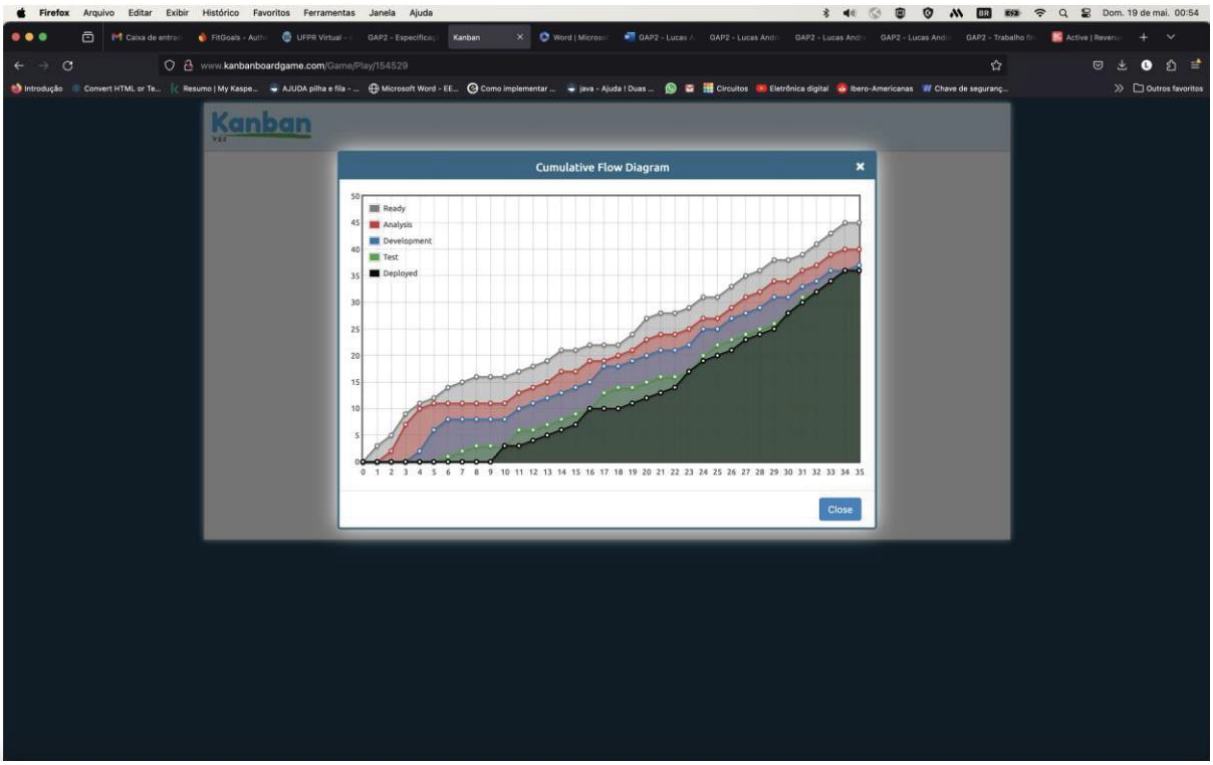
Figura 11 - Gestão por fluxo contínuo





FONTE: O autor (2025)

Figura 12 – Cumulative Flow Diagram



FONTE: O autor (2025)

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação teve como foco principal a construção das bases da lógica de programação e da implementação de sistemas, com ênfase na linguagem Java. O conteúdo foi dividido em quatro tópicos principais: Introdução às linguagens de programação, Sintaxe de Programação, Orientação a Objetos e Integração com Banco de Dados.

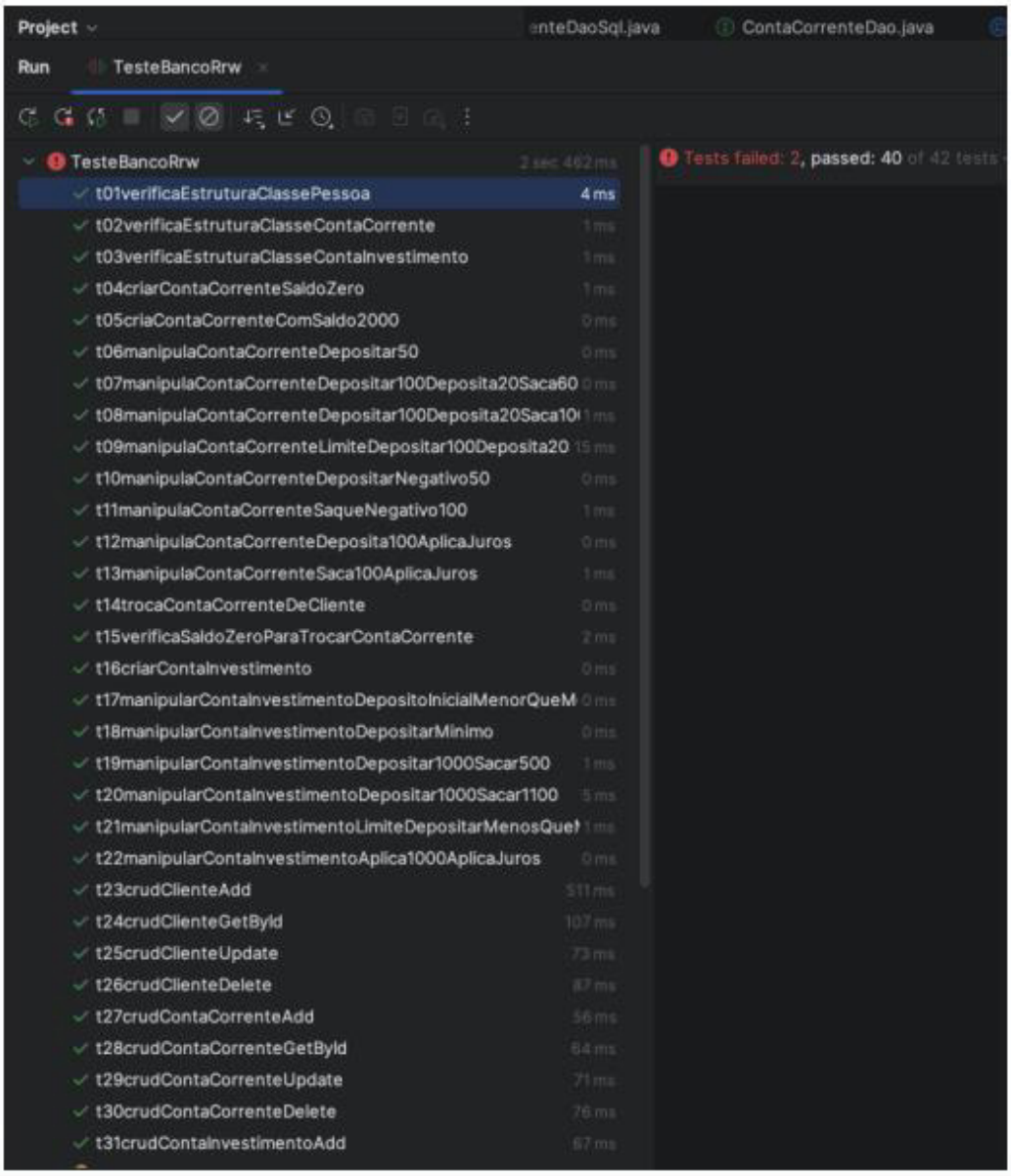
O trabalho final da disciplina consistiu na implementação de classes de um sistema a partir de um diagrama de classes previamente fornecido. Também foram utilizados um script DDL (*Data Definition Language*) para criação das tabelas no banco de dados e uma suíte de testes unitários, ambos disponibilizados para a execução do trabalho. A validação da atividade se deu por meio da execução, com sucesso, de 42 casos de testes unitários, evidenciando a correta implementação das funcionalidades esperadas.

Esse projeto foi fundamental para aplicar de forma prática os conceitos de orientação a objetos, encapsulamento, herança, composição e manipulação de dados persistidos em banco. O exercício de implementar as classes e garantir que passassem por todos os testes reforçou a importância da qualidade do código desde as etapas iniciais do desenvolvimento, algo essencial para ambientes ágeis, onde mudanças frequentes e entregas incrementais exigem uma base sólida e testável.

Dessa forma, a disciplina de Introdução à Programação teve papel central na formação técnica, fornecendo as ferramentas necessárias para que o desenvolvimento ágil de software seja realizado com qualidade, estrutura e capacidade de adaptação contínua.

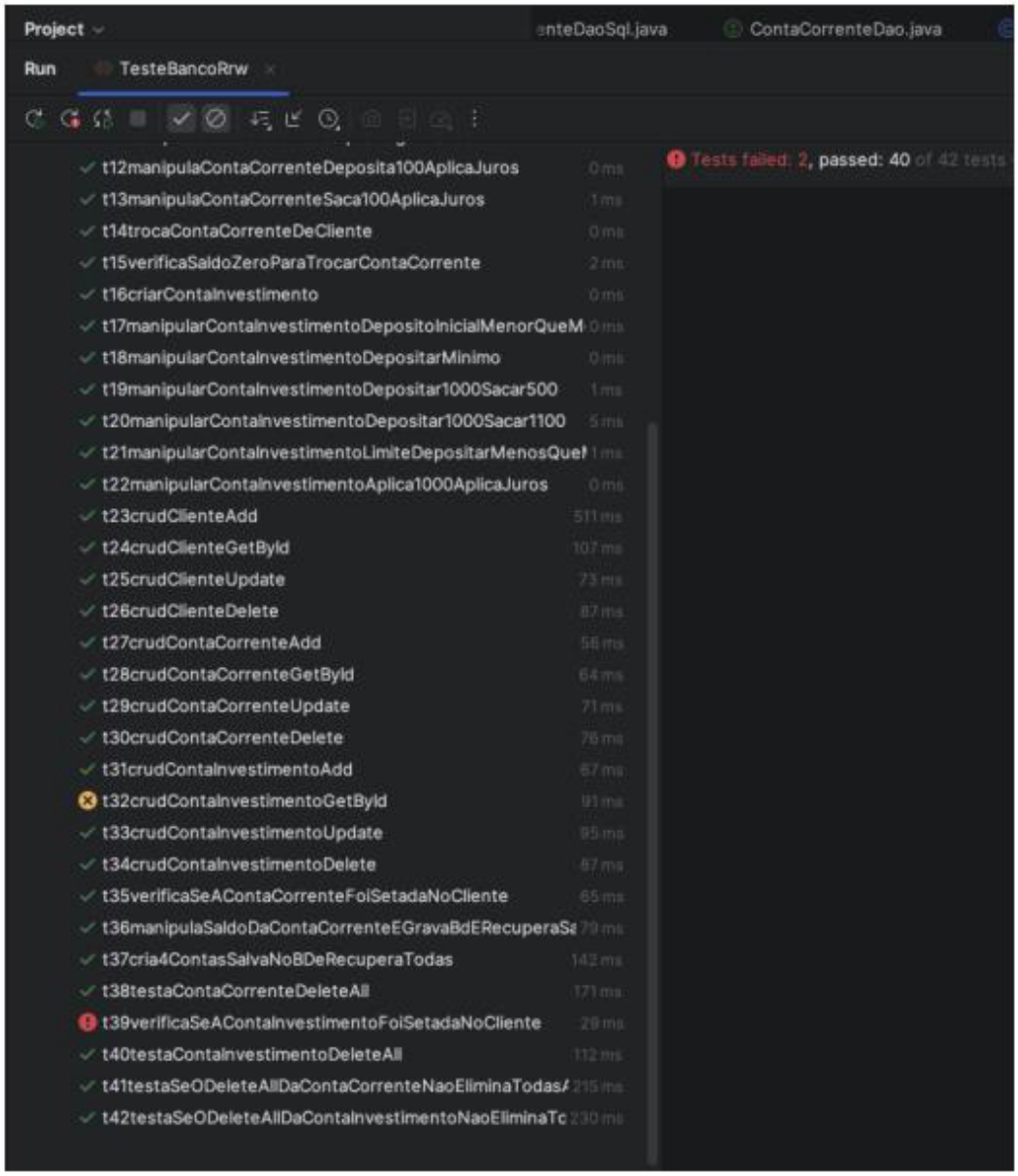
5.1 ARTEFATOS DO PROJETO

Figura 13 – Testes unitários 1



FONTE: O autor (2025)

Figura 14 – Testes unitários 2



FONTE: O autor (2025)

6 DISCIPLINA: BD – BANCO DE DADOS

A disciplina de Banco de Dados teve como objetivo fornecer uma base sólida sobre o armazenamento, organização e manipulação de dados em sistemas computacionais. Os conteúdos foram distribuídos em três unidades, abordando desde os fundamentos teóricos até a aplicação prática com modelagem e linguagem SQL.

Na Unidade 1, foram apresentados os Fundamentos de Banco de Dados, incluindo a evolução dos modelos de organização da informação, desde a estrutura baseada em arquivos até os Sistemas Gerenciadores de Banco de Dados (SGBDs). Foram estudados os modelos existentes, com destaque para o modelo relacional, suas propriedades transacionais (como atomicidade, consistência, isolamento e durabilidade – ACID) e os conceitos fundamentais propostos por Edgar Frank Codd (Codd, 1970).

A Unidade 2 focou na Modelagem de Dados, com a construção de modelos conceituais e lógicos, promovendo a compreensão de entidades, relacionamentos e suas restrições. Já na Unidade 3, foi introduzida a linguagem SQL como ferramenta essencial para a criação, consulta e manipulação de dados em bancos relacionais.

O trabalho final da disciplina foi composto por duas partes. A primeira consistiu na modelagem de um sistema de controle de biblioteca, a partir de um conjunto de requisitos fornecido. O projeto incluiu a criação do Modelo Entidade-Relacionamento Conceitual (focado em entidades e relacionamentos) e o Modelo Lógico com tabelas, campos, chaves primárias e estrangeiras. Essa etapa foi essencial para consolidar o entendimento da estrutura e integridade dos dados.

A segunda parte do trabalho envolveu a especificação e implementação em SQL (Date, 2003) de um sistema com tema livre, desde que contemplasse ao menos um exemplo de cada tipo de relacionamento (1:1, 1:N e N:N). A atividade exigiu a geração do modelo lógico e a criação do script SQL correspondente, incluindo *constraints* e inserção de registros. Com isso, foi possível exercitar a construção completa de um esquema de banco de dados, desde a concepção até a implementação prática.

Esse projeto permitiu aplicar os conhecimentos adquiridos de forma concreta e integrada, desenvolvendo habilidades essenciais para o planejamento e sustentação de sistemas com persistência de dados. Além disso, reforçou o papel do banco de dados como componente fundamental em qualquer aplicação de software,

especialmente em contextos ágeis, onde a consistência, a clareza do modelo e a eficiência nas consultas e atualizações impactam diretamente a qualidade e a velocidade das entregas.

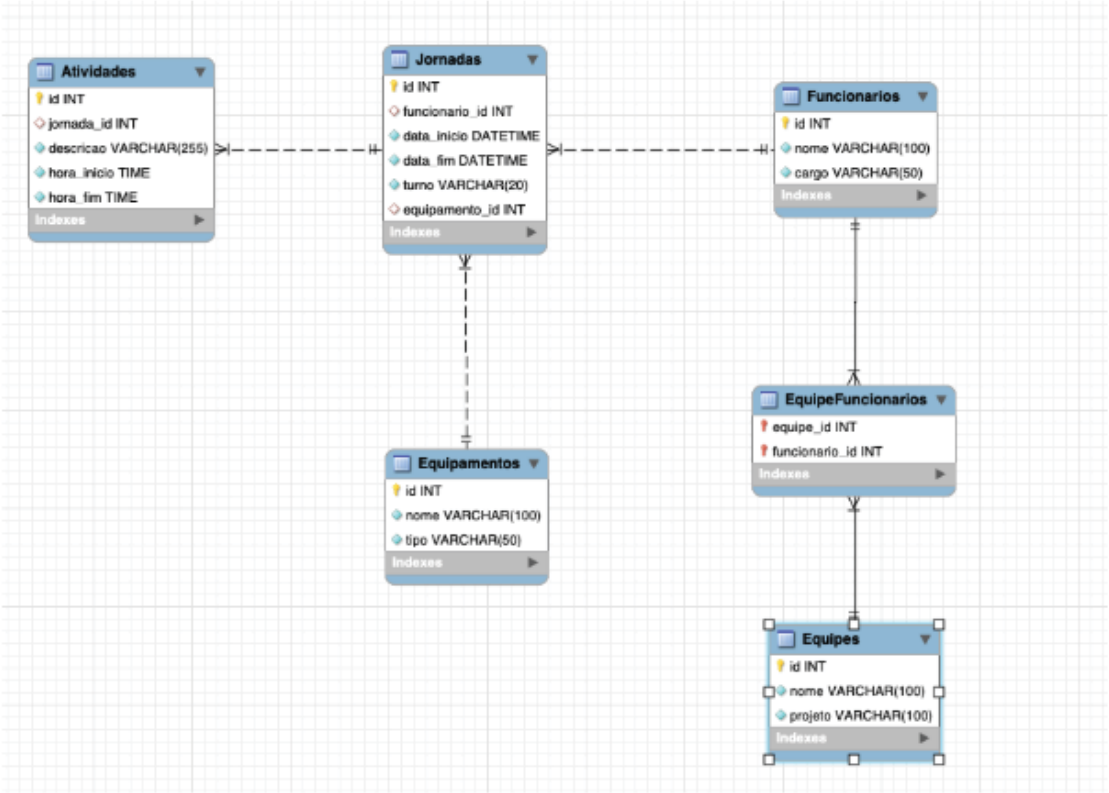
6.1 ARTEFATOS DO PROJETO

Figura 15 – EER Diagram

Sistema de controle de jornadas

Relacionamento 1x1 (Jornadas e Equipamentos): Cada jornada de trabalho pode utilizar um único equipamento, e cada equipamento é associado a uma única jornada por vez.
Relacionamento 1xN (Funcionários e Jornadas): Um funcionário pode ter várias jornadas de trabalho, mas cada jornada está associada a um único funcionário.
Relacionamento NxN (Funcionários e Equipes): Funcionários podem pertencer a várias equipes, e equipes podem ter vários funcionários.

EER Diagram



FONTE: O autor (2025)

Figura 16 – Script SQL

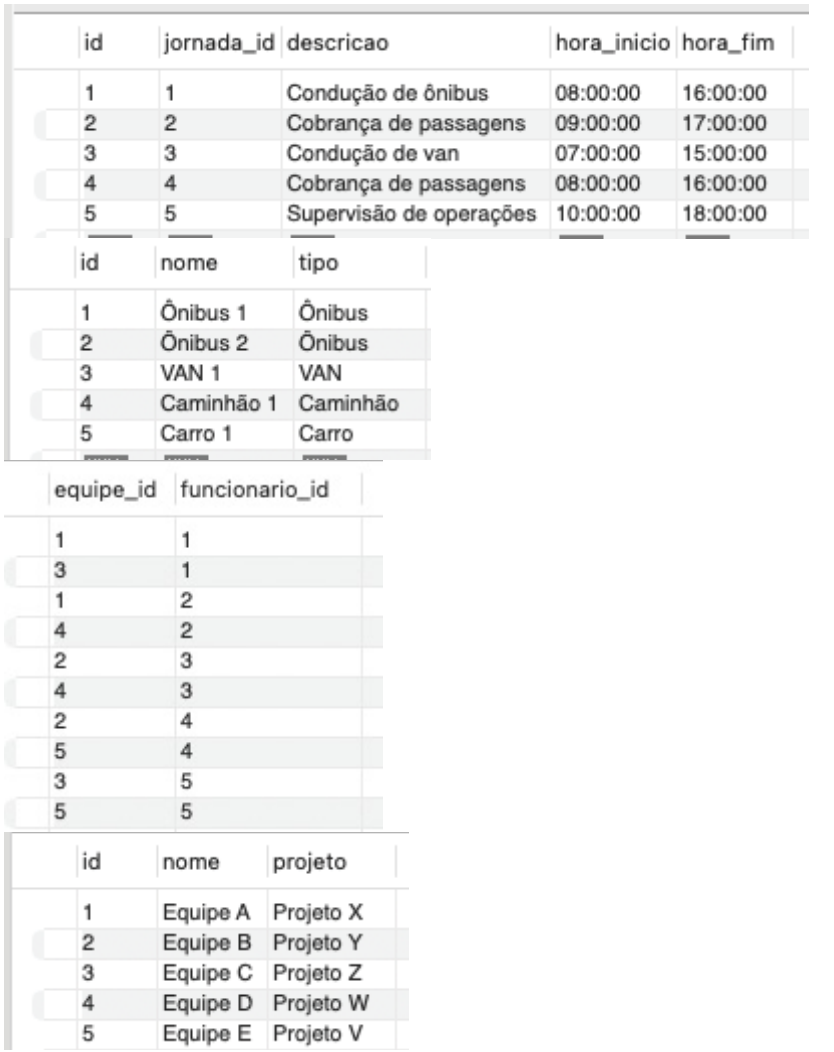
```
-- Criação do banco de dados
CREATE DATABASE ControleJornadas;
USE ControleJornadas;

-- Tabela Funcionários
CREATE TABLE Funcionarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cargo VARCHAR(50) NOT NULL
);

-- Tabela Equipamentos
CREATE TABLE Equipamentos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    tipo VARCHAR(50) NOT NULL
);

-- Tabela Jornadas
CREATE TABLE Jornadas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    funcionario_id INT,
    data_inicio DATETIME NOT NULL,
    data_fim DATETIME NOT NULL,
    turno VARCHAR(20) NOT NULL,
    equipamento_id INT UNIQUE,
    FOREIGN KEY (funcionario_id) REFERENCES Funcionarios(id),
    FOREIGN KEY (equipamento_id) REFERENCES Equipamentos(id)
);

-- Tabela Atividades
CREATE TABLE Atividades (
    id INT AUTO_INCREMENT PRIMARY KEY,
    jornada_id INT,
    descricao VARCHAR(255) NOT NULL,
    hora_inicio TIME NOT NULL,
    hora_fim TIME NOT NULL,
    FOREIGN KEY (jornada_id) REFERENCES Jornadas(id)
);

-- Tabela Equipes
CREATE TABLE Equipes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    projeto VARCHAR(100) NOT NULL
);

-- Tabela de junção para o relacionamento NxN entre Equipes e
Funcionários
```

```
CREATE TABLE EquipeFuncionarios (
    equipe_id INT,
    funcionario_id INT,
    PRIMARY KEY (equipe_id, funcionario_id),
    FOREIGN KEY (equipe_id) REFERENCES Equipes(id),
    FOREIGN KEY (funcionario_id) REFERENCES Funcionarios(id)
);

-- Funcionários
INSERT INTO Funcionarios (nome, cargo) VALUES
('João Silva', 'Motorista'),
('Maria Oliveira', 'Cobradora'),
('Carlos Lima', 'Motorista'),
('Ana Souza', 'Cobradora'),
('Pedro Santos', 'Supervisor');

-- Equipamentos
INSERT INTO Equipamentos (nome, tipo) VALUES
('Ônibus 1', 'Ônibus'),
('Ônibus 2', 'Ônibus'),
('VAN 1', 'VAN'),
('Caminhão 1', 'Caminhão'),
('Carro 1', 'Carro');

-- Jornadas
INSERT INTO Jornadas (funcionario_id, data_inicio, data_fim,
turno, equipamento_id) VALUES
(1, '2023-06-01 08:00:00', '2023-06-01 16:00:00', 'Manhã', 1),
(2, '2023-06-01 09:00:00', '2023-06-01 17:00:00', 'Tarde', 2),
(3, '2023-06-02 07:00:00', '2023-06-02 15:00:00', 'Manhã', 3),
(4, '2023-06-02 08:00:00', '2023-06-02 16:00:00', 'Manhã', 4),
(5, '2023-06-03 10:00:00', '2023-06-03 18:00:00', 'Tarde', 5);

-- Atividades
INSERT INTO Atividades (jornada_id, descricao, hora_inicio,
hora_fim) VALUES
(1, 'Condução de ônibus', '08:00:00', '16:00:00'),
(2, 'Cobrança de passagens', '09:00:00', '17:00:00'),
(3, 'Condução de van', '07:00:00', '15:00:00'),
(4, 'Cobrança de passagens', '08:00:00', '16:00:00'),
(5, 'Supervisão de operações', '10:00:00', '18:00:00');

-- Equipes
INSERT INTO Equipes (nome, projeto) VALUES
('Equipe A', 'Projeto X'),
('Equipe B', 'Projeto Y'),
('Equipe C', 'Projeto Z'),
('Equipe D', 'Projeto W'),
('Equipe E', 'Projeto V');
```

```
-- EquipeFuncionarios
INSERT INTO EquipeFuncionarios (equipe_id, funcionario_id)
VALUES
(1, 1),
(1, 2),
(2, 3),
(2, 4),
(3, 5),
(3, 1),
(4, 2),
(4, 3),
(5, 4),
(5, 5);
```

FONTE: O autor (2025)

Figura 17 – Resultados de script SQL



id	jornada_id	descricao	hora_inicio	hora_fim
1	1	Condução de ônibus	08:00:00	16:00:00
2	2	Cobrança de passagens	09:00:00	17:00:00
3	3	Condução de van	07:00:00	15:00:00
4	4	Cobrança de passagens	08:00:00	16:00:00
5	5	Supervisão de operações	10:00:00	18:00:00

id	nome	tipo
1	Ônibus 1	Ônibus
2	Ônibus 2	Ônibus
3	VAN 1	VAN
4	Caminhão 1	Caminhão
5	Carro 1	Carro

equipe_id	funcionario_id
1	1
3	1
1	2
4	2
2	3
4	3
2	4
5	4
3	5
5	5

id	nome	projeto
1	Equipe A	Projeto X
2	Equipe B	Projeto Y
3	Equipe C	Projeto Z
4	Equipe D	Projeto W
5	Equipe E	Projeto V

id	nome	cargo
1	João Silva	Motorista
2	Maria Oliveira	Cobradora
3	Carlos Lima	Motorista
4	Ana Souza	Cobradora
5	Pedro Santos	Supervisor

id	funcionario_id	data_inicio	data_fim	turno	equipamento_id
1	1	2023-06-01 08:00:00	2023-06-01 16:00:00	Manhã	1
2	2	2023-06-01 09:00:00	2023-06-01 17:00:00	Tarde	2
3	3	2023-06-02 07:00:00	2023-06-02 15:00:00	Manhã	3
4	4	2023-06-02 08:00:00	2023-06-02 16:00:00	Manhã	4
5	5	2023-06-03 10:00:00	2023-06-03 18:00:00	Tarde	5

FONTE: O autor (2025)

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação teve como objetivo apresentar e exercitar práticas essenciais para a escrita de código limpo, legível e sustentável, alinhadas aos princípios do desenvolvimento ágil. Foram abordados temas como nomes significativos, funções com responsabilidade única, formatação consistente, uso adequado de comentários, tratamento de erros, testes automatizados, coesão, acoplamento, abstração, refatoração contínua, e práticas colaborativas como *pair* e *mob programming* (Martin, 2002).

O trabalho final da disciplina consistiu na refatoração de um código Java que implementava o algoritmo *Bubble Sort*, aplicando os conceitos de clean code estudados em aula. O objetivo foi tornar o código mais legível, organizado e de fácil manutenção, seguindo boas práticas reconhecidas no desenvolvimento profissional.

Esse projeto teve grande importância para reforçar a compreensão de que, em contextos ágeis, a qualidade do código é um dos pilares fundamentais para a evolução contínua e rápida de sistemas. Um código bem escrito favorece a colaboração entre os membros da equipe, facilita a identificação de erros e permite mudanças frequentes com menor risco. Além disso, a prática de refatoração e melhoria incremental está diretamente conectada ao ciclo de feedback rápido e à entrega contínua de valor, característicos do desenvolvimento ágil.

Dessa forma, a disciplina contribuiu de forma significativa para a formação técnica e profissional, promovendo uma mentalidade de cuidado e responsabilidade com o código desde as etapas iniciais de um projeto.

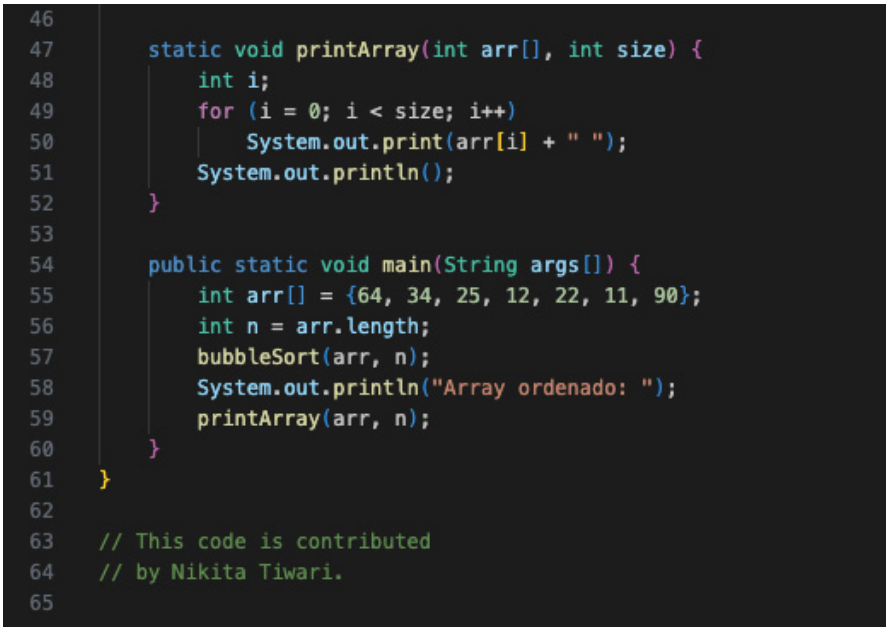
7.1 ARTEFATOS DO PROJETO

Figura 18 – Algoritmo Bubble Sort

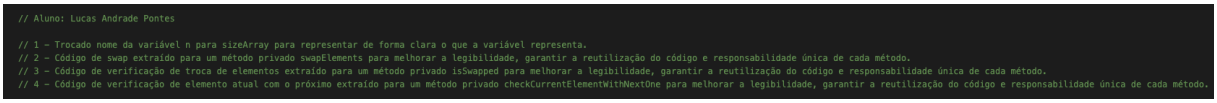
```
class BubbleSort {

    // Aluno: Lucas Andrade Pontes

    // 1 - Trocado nome da variável n para sizeArray para representar de forma clara o que a
    // 2 - Código de swap extraído para um método privado swapElements para melhorar a legibi
    // 3 - Código de verificação de troca de elementos extraído para um método privado isSwap
    // 4 - Código de verificação de elemento atual com o próximo extraído para um método priv

    static void bubbleSort(int[] arr, int arraySize) {
        int outerIndex;

        for (outerIndex = 0; outerIndex < arraySize - 1; outerIndex++) {

            boolean swapped = isSwapped(arr, arraySize, outerIndex);

            if (!swapped)
                break;
        }
    }

    private static boolean isSwapped(int[] arr, int arraySize, int outerIndex) {
        boolean swapped = false;
        int innerIndex;
        for (innerIndex = 0; innerIndex < arraySize - outerIndex - 1; innerIndex++) {
            if (checkCurrentElementWithNextOne(arr, innerIndex)) {
                swapElements(arr, innerIndex);
                swapped = true;
            }
        }
        return swapped;
    }

    private static boolean checkCurrentElementWithNextOne(int[] arr, int innerIndex) {
        return arr[innerIndex] > arr[innerIndex + 1];
    }

    private static void swapElements(int[] arr, int index) {
        int temp = arr[index];
        arr[index] = arr[index + 1];
        arr[index + 1] = temp;
    }
```

FONTE: O autor (2025)

Figura 19 – Código principal



```
46
47      static void printArray(int arr[], int size) {
48          int i;
49          for (i = 0; i < size; i++)
50              System.out.print(arr[i] + " ");
51          System.out.println();
52      }
53
54      public static void main(String args[]) {
55          int arr[] = {64, 34, 25, 12, 22, 11, 90};
56          int n = arr.length;
57          bubbleSort(arr, n);
58          System.out.println("Array ordenado: ");
59          printArray(arr, n);
60      }
61  }
62
63  // This code is contributed
64  // by Nikita Tiwari.
65
```



```
// Aluno: Lucas Andrade Pontes

// 1 - Trocado nome da variável n para sizeArray para representar de forma clara o que a variável representa.
// 2 - Código de swap extraído para um método privado swapElements para melhorar a legibilidade, garantir a reutilização do código e responsabilidade única de cada método.
// 3 - Código de verificação de troca de elementos extraído para um método privado isSwapped para melhorar a legibilidade, garantir a reutilização do código e responsabilidade única de cada método.
// 4 - Código de verificação de elemento atual com o próximo extraído para um método privado checkCurrentElementWithNextOne para melhorar a legibilidade, garantir a reutilização do código e responsabilidade única de cada método.
```

FONTE: O autor (2025)

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

As disciplinas de Desenvolvimento Web 1 e 2 formam um núcleo complementar voltado à construção de aplicações *web* modernas, com foco em tecnologias amplamente utilizadas no mercado. Juntas, permitiram compreender desde os fundamentos do front-end com Angular até a implementação de APIs RESTful (Fielding, 2000) e persistência de dados no *back-end* utilizando Spring Boot. Essa combinação é especialmente relevante em contextos de desenvolvimento ágil, nos quais a entrega contínua, a integração entre camadas e a adaptabilidade técnica são fundamentais.

Em Desenvolvimento Web 1, o foco foi a criação de interfaces *web* utilizando HTML, *TypeScript*, *Bootstrap*, *Material Design* e o *framework* Angular. O conteúdo abordou desde os conceitos básicos da web e a estrutura de projetos até a construção de um CRUD (*Create, Read, Update, Delete*) (Date, 2003) completo para cadastro de pessoas, incluindo os componentes de listagem, inserção, edição e remoção. Também foram trabalhadas as boas práticas com formulários e a organização de código em módulos e serviços. Essa base foi fundamental para o domínio da camada de apresentação e lógica de interface, facilitando protótipos rápidos, testes com usuários e a entrega de valor visual desde os primeiros ciclos de desenvolvimento.

Já em Desenvolvimento Web 2, o foco foi o desenvolvimento da camada de *back-end* com Spring Boot. Foram abordados temas como criação e consumo de APIs REST (Fielding, 2000), programação reativa, autenticação com login, uso do Spring Data JPA, integração com bancos de dados e desenvolvimento de funcionalidades de CRUD (Date, 2003) tanto para usuários quanto para pessoas. Também foi enfatizada a comunicação entre o Angular e o *back-end*, utilizando chamadas HTTP para efetivar a persistência e recuperação de dados. A disciplina forneceu uma visão clara de como estruturar sistemas *web* completos, seguros, escaláveis e integrados

Embora os trabalhos finais dessas disciplinas fossem opcionais e não tenham sido realizados, o conteúdo estudado foi suficiente para capacitar o aluno a desenvolver aplicações *web* robustas, compreendendo o ciclo completo do desenvolvimento — da interface do usuário ao armazenamento de dados. Em ambientes ágeis, essa fluidez entre *front-end* e *back-end* permite entregas mais rápidas, maior colaboração entre equipes e maior adaptabilidade às mudanças de requisitos.

Dessa forma, as disciplinas de Desenvolvimento Web 1 e 2 tiveram papel essencial na formação técnica do aluno, integrando conhecimentos que permitem atuar com eficiência em projetos reais de software sob uma abordagem ágil, centrada em valor, integração contínua e evolução incremental.

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

A disciplina UX no Desenvolvimento Ágil de Software teve como foco principal apresentar os fundamentos da experiência do usuário e como integrá-los aos ciclos iterativos e incrementais de desenvolvimento ágil. Por meio do estudo de prototipagem, acessibilidade, testes com usuários e princípios emocionais no design, a disciplina reforçou a importância de construir soluções digitais centradas nas pessoas desde as fases iniciais do projeto.

Foram abordadas diferentes ferramentas de prototipagem, como Figma, Adobe XD, *Just in Mind* e Bootstrap, além da linguagem visual do *Material Design* (Google, 2025d). Os alunos estudaram protótipos de baixa, média e alta fidelidade e compreenderam suas aplicações práticas nas etapas de ideação, validação e refinamento de interfaces. Também foram discutidos os princípios do *design* acessível (W3C, 2024), ferramentas de verificação de acessibilidade e o papel das emoções na experiência digital — temas cada vez mais relevantes no desenvolvimento de produtos centrados no usuário (Norman, 2004).

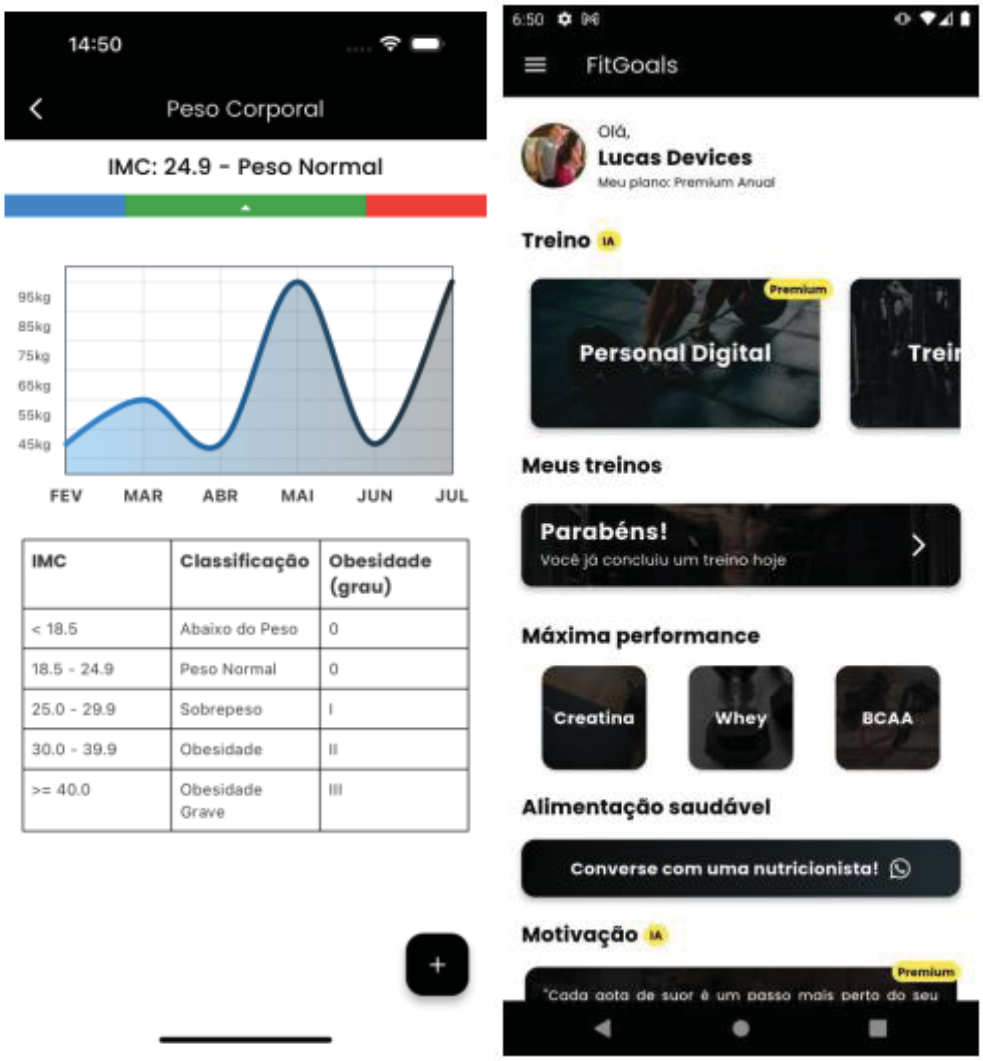
O trabalho final consistiu na entrega de um protótipo funcional de um aplicativo ou site, acompanhado de justificativa do projeto, explicação das decisões de design, simulação da interface e coleta de feedback com um usuário real. Essa atividade promoveu a aplicação prática de todos os conteúdos da disciplina, estimulando a empatia com os usuários e o pensamento iterativo orientado à melhoria contínua.

A integração entre UX e métodos ágeis ficou clara ao longo da disciplina. A realização de testes com protótipos, por exemplo, permite que os times coletem *feedback* rapidamente e ajustem o produto ainda nas fases iniciais do desenvolvimento, reduzindo custos e aumentando a satisfação do usuário final. Além disso, ao incorporar práticas de UX desde o início, equipes ágeis aumentam suas chances de entregar valor de forma contínua e assertiva.

Dessa forma, a disciplina teve papel fundamental na formação de uma mentalidade centrada no usuário dentro do contexto do desenvolvimento ágil, reforçando que tecnologia, *design* e empatia devem caminhar juntos para gerar soluções mais eficazes, inclusivas e desejáveis.

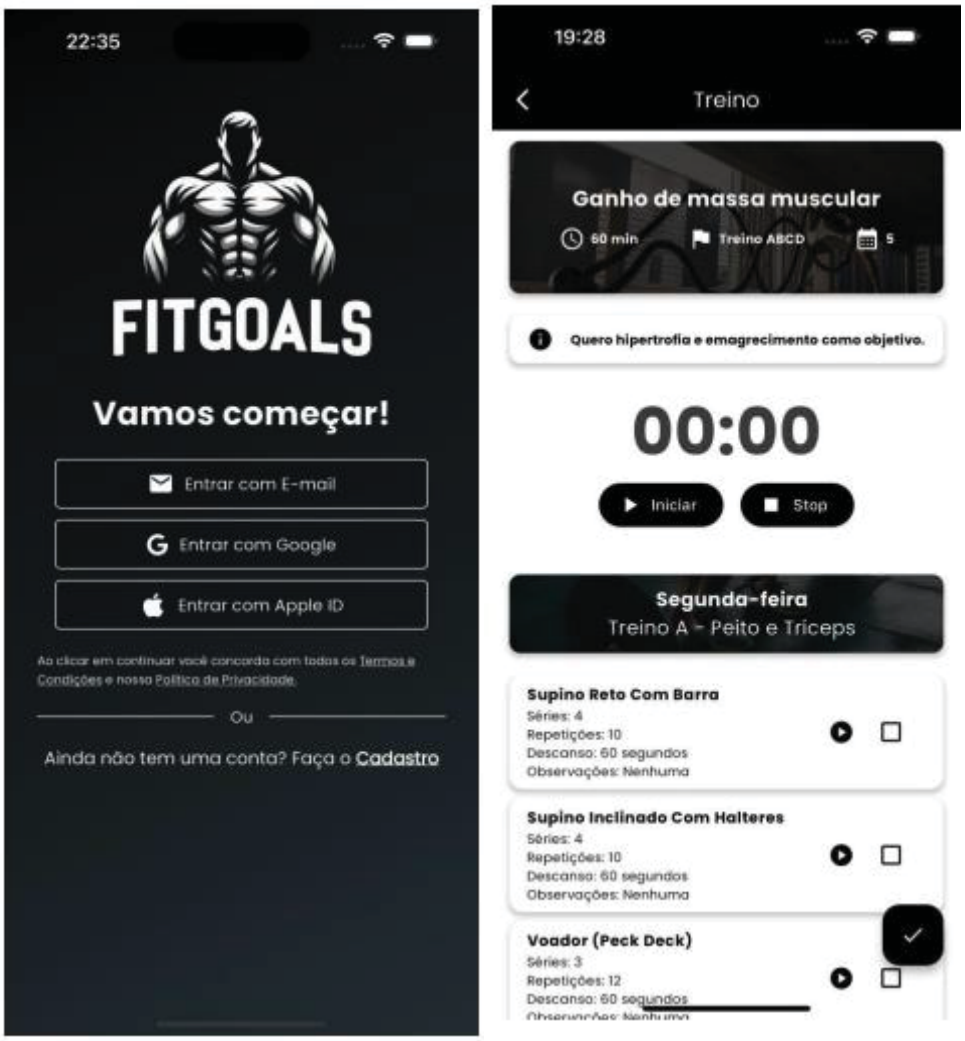
9.1 ARTEFATOS DO PROJETO

Figura 20 – Telas de início e de IMC do aplicativo FitGoals



FONTE: O autor (2025)

Figura 21 – Telas de login e de treino do aplicativo FitGoals



FONTE: O autor (2025)

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

As disciplinas de Desenvolvimento Mobile 1 e 2 forneceram uma base sólida para o desenvolvimento de aplicativos móveis nativos com Android e Kotlin (Google, 2025a), capacitando o aluno a compreender e aplicar os conceitos essenciais da programação mobile em projetos reais. Por serem disciplinas complementares, juntas formaram uma trilha contínua de aprendizado, do básico ao avançado, permitindo que a construção de aplicativos acompanhasse os princípios do desenvolvimento ágil.

Em Desenvolvimento Mobile 1, os conteúdos abordaram a estrutura dos projetos Android, o ciclo de vida das *activities*, a construção de interfaces com *ConstraintLayout*, a navegação entre telas por meio de *intents*, e o uso do *SharedPreferences* para persistência simples de dados. A disciplina também introduziu o uso de eventos, manipulação de componentes visuais e a organização do código com classes e pacotes, permitindo ao aluno construir aplicações funcionais e bem estruturadas desde o início.

Já em Desenvolvimento Mobile 2, os conteúdos avançaram para temas como o uso de listas com *RecyclerView*, personalização de *layouts*, manipulação de cliques com funções lambda, persistência com banco de dados SQLite, e leitura/gravação de arquivos no armazenamento interno do dispositivo. As aulas exploraram ainda a organização do código em camadas, com separação clara entre visualização, controle e dados, aproximando a prática do desenvolvimento profissional de aplicativos móveis robustos.

Apesar de os trabalhos finais das duas disciplinas serem opcionais e não terem sido realizados neste caso, os conteúdos desenvolvidos em aula e os projetos práticos ao longo das disciplinas foram suficientes para consolidar o domínio das ferramentas e técnicas abordadas.

No contexto do desenvolvimento ágil, essas disciplinas desempenharam papel importante ao permitir entregas incrementais de valor por meio da construção de funcionalidades completas, testáveis e integráveis com outras camadas do sistema. A capacidade de construir interfaces, interações e persistência local rapidamente é altamente valorizada em metodologias ágeis, em que iteração, *feedback* e evolução contínua são pilares centrais.

Dessa forma, Desenvolvimento Mobile 1 e 2 contribuíram significativamente para a formação prática e técnica do aluno, integrando-se de forma natural aos demais

aprendizados do curso e reforçando uma atuação ágil, autônoma e alinhada às demandas atuais do mercado.

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

A disciplina Infraestrutura para Desenvolvimento e Implantação de *Software* (DevOps) teve como principal objetivo apresentar os fundamentos, ferramentas e práticas necessárias para sustentar o ciclo moderno de desenvolvimento de *software* com foco em automação, integração contínua, entrega contínua e confiabilidade operacional.

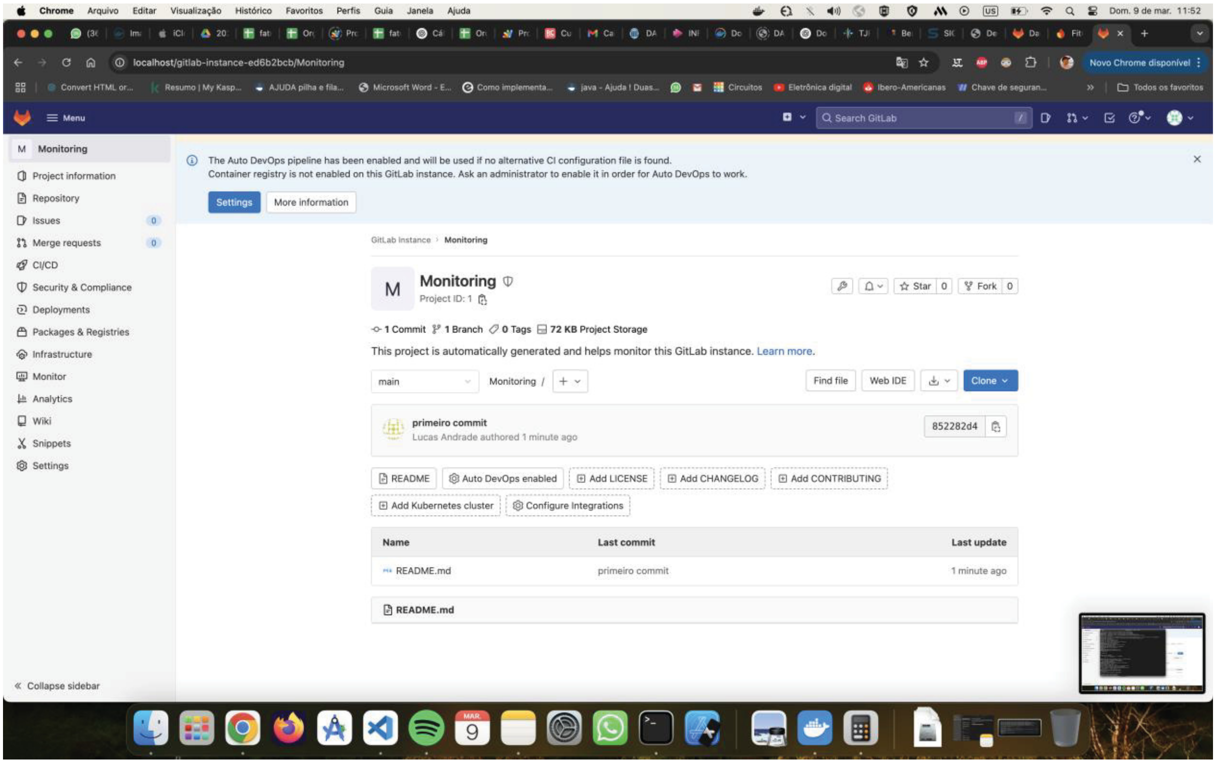
Durante as aulas, foram explorados temas como: ciclo de vida de desenvolvimento de *software* (SDLC), cultura DevOps, DevSecOps, práticas de *Shift Left,* integração contínua (CI), entrega contínua (CD), uso de *containers* com Docker, orquestração com Kubernetes, versionamento com Git, infraestrutura como código, *trunk-based development*, monitoramento e métricas DevOps (como *Deployment Frequency, Lead Time, Change Failure Rate* e MTTR). O conteúdo também incluiu bibliografias fundamentais como *The DevOps Handbook*, Projeto Fênix e Descomplicando o Docker.

O trabalho final da disciplina consistiu em uma aplicação prática dos principais conceitos estudados. Essa atividade permitiu colocar em prática conhecimentos relacionados à criação e gestão de ambientes de desenvolvimento integrados com ferramentas amplamente utilizadas em *pipelines* de CI/CD. Além disso, reforçou a autonomia no uso de *containers*, a navegação em sistemas reais (como GitLab e Jenkins) e a interação entre versionamento de código e automação de processos — pilares fundamentais do DevOps.

A disciplina, portanto, teve impacto direto na formação de competências essenciais para o desenvolvimento ágil de *software*, contribuindo para que o aluno compreenda e aplique práticas modernas de entrega contínua, infraestrutura automatizada e colaboração entre times de desenvolvimento e operações.

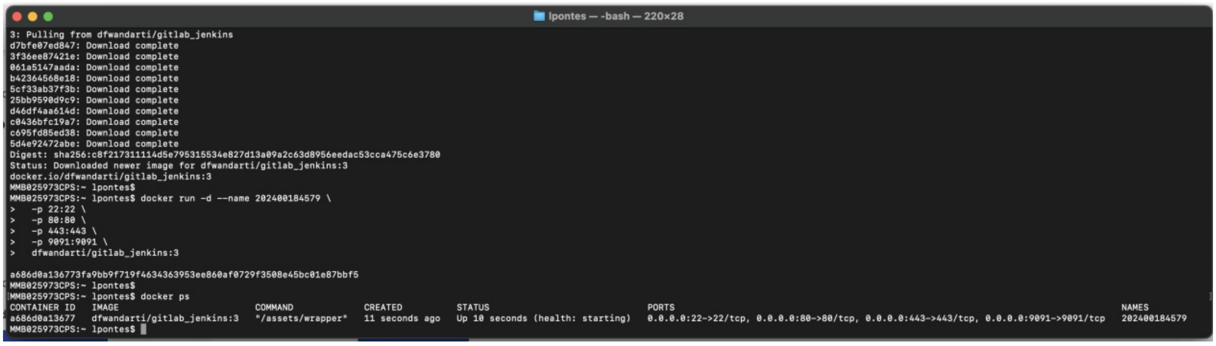
11.1 ARTEFATOS DO PROJETO

Figura 22 - Repositório no GitLab



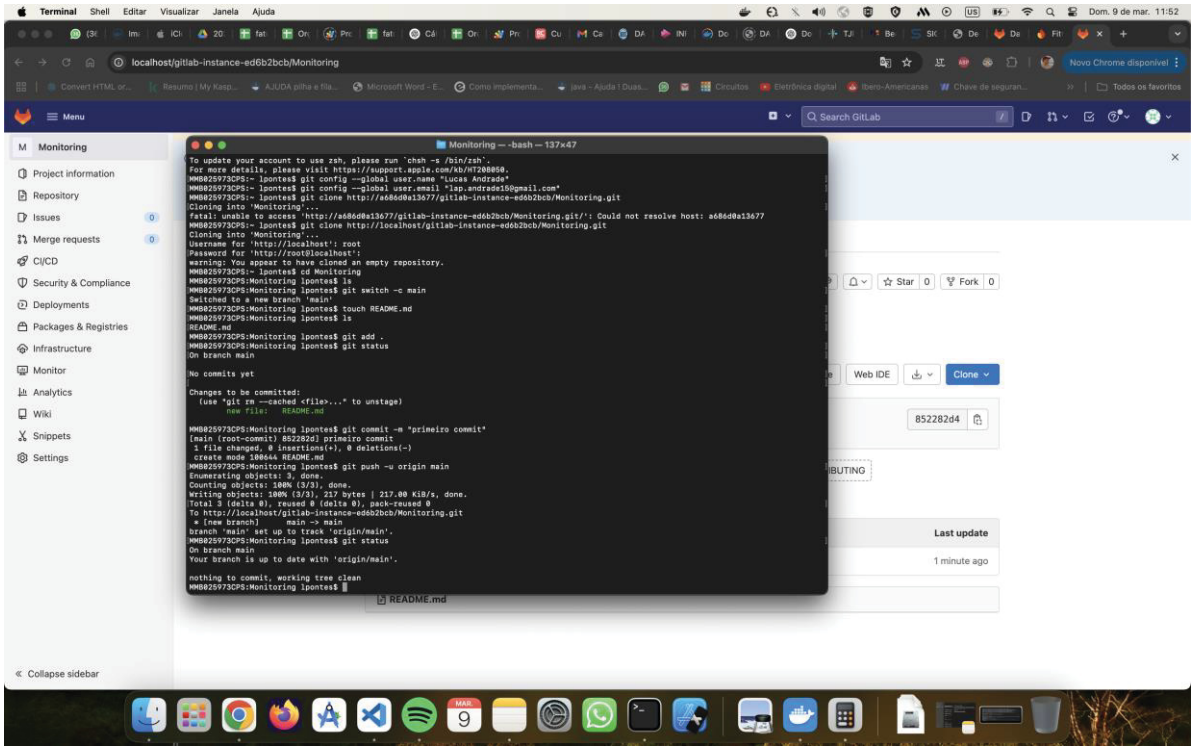
FONTE: O autor (2025)

Figura 23 - Terminal com docker em execução



FONTE: O autor (2025)

Figura 24 – Terminal com git em execução



FONTE: O autor (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes de *Software* teve como foco apresentar os principais conceitos, técnicas e ferramentas relacionadas à garantia da qualidade no desenvolvimento ágil. Dentro de um contexto onde mudanças constantes fazem parte do processo, aprender a testar de forma eficiente e automatizada se tornou essencial para garantir entregas confiáveis, sustentáveis e seguras.

Foram abordados temas como a Pirâmide de Testes (Cohn, 2009), que orienta a proporção ideal entre testes unitários, de integração e *end-to-end*; a abordagem *Shift-Left*, que incentiva a execução de testes desde os estágios iniciais do desenvolvimento; além da importância dos Testes Automatizados, que aumentam a frequência e a segurança nas entregas contínuas. A disciplina também introduziu ferramentas modernas como o *Playwright*, voltado à automação de testes de interface com alto desempenho e cobertura.

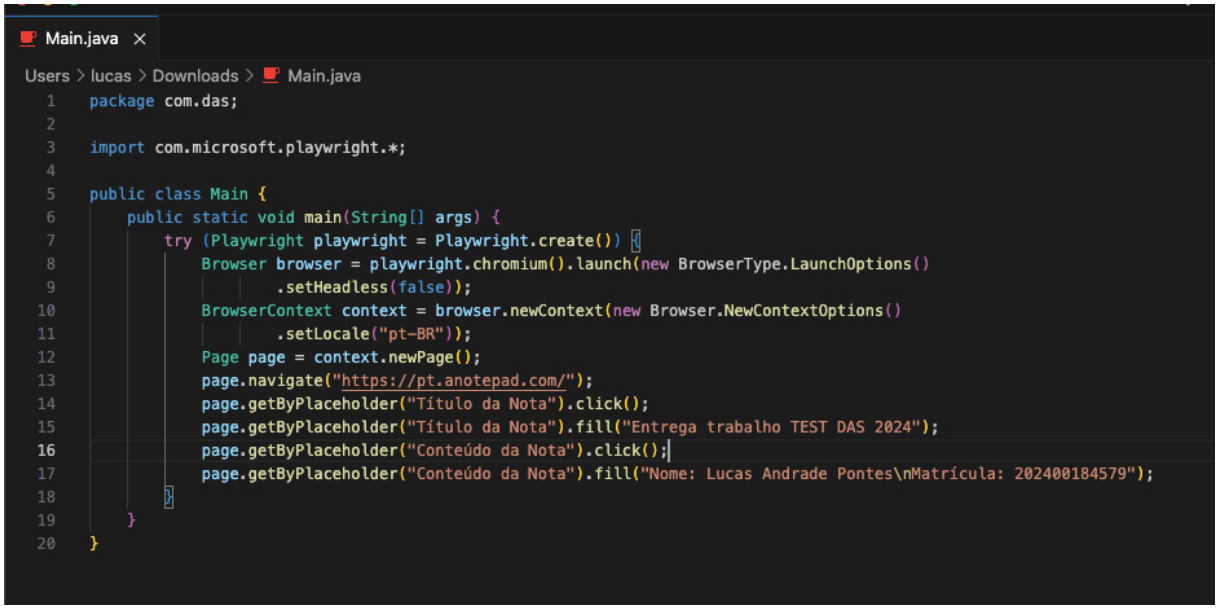
O trabalho final da disciplina consistiu na implementação de um *script* automatizado de teste usando o *Playwright*, cujo objetivo era abrir automaticamente o site https://pt.anotepad.com, criar uma nota com título e corpo personalizados, conforme orientações do professor. Essa atividade permitiu que o aluno aplicasse na prática os conceitos de automação de testes, validação funcional de sistemas e boas práticas de escrita de testes automatizados.

A disciplina reforçou a importância dos testes como parte integrante do ciclo de desenvolvimento ágil, permitindo *feedback* rápido, menor retrabalho e aumento da confiança na evolução do sistema. A automatização, em especial, se mostrou uma ferramenta estratégica para equipes ágeis, que precisam manter velocidade de entrega sem comprometer a qualidade do produto.

Dessa forma, Testes de *Software* teve papel essencial na consolidação de uma cultura de qualidade no curso, complementando os aprendizados técnicos e fortalecendo o compromisso com entregas confiáveis e sustentáveis no desenvolvimento ágil.

12.1 ARTEFATOS DO PROJETO

Figura 25 – Classe Java com código do PlayWright



FONTE: O autor (2025)

13 CONCLUSÃO

O presente memorial apresentou uma jornada de aprendizado que integrou teoria e prática na área de Desenvolvimento Ágil de *Software*, por meio da execução de atividades, projetos e artefatos desenvolvidos ao longo das disciplinas da especialização. Cada módulo contribuiu de maneira específica para a formação de uma mentalidade ágil, colaborativa, técnica e orientada à entrega de valor — características fundamentais para enfrentar os desafios reais do desenvolvimento de *software* no cenário atual.

A partir dos fundamentos iniciais sobre métodos ágeis e suas aplicações práticas (como *Scrum, Kanban, XP* e *Lean*), até disciplinas mais técnicas voltadas à programação, modelagem, testes, infraestrutura e experiência do usuário, foi possível perceber como as etapas do ciclo de vida de um sistema se conectam e se beneficiam de uma abordagem iterativa, incremental e centrada no cliente. Projetos como os modelos ágeis de banco de dados, protótipos de interfaces, automações com testes e pipelines DevOps demonstraram, na prática, o alinhamento entre o conteúdo acadêmico e as exigências do mercado.

Entre os principais desafios percebidos durante a construção deste memorial, destaca-se a necessidade de mudança cultural para adoção efetiva das práticas ágeis. Em muitos ambientes corporativos, ainda persiste uma mentalidade tradicional, focada em controle rígido, previsibilidade e documentação extensiva, o que dificulta a aplicação plena de metodologias que valorizam adaptação contínua, colaboração e resposta rápida a mudanças. Além disso, a integração entre times multidisciplinares, o domínio técnico para automação de testes e entregas, e o alinhamento constante com as necessidades do cliente também se mostraram aspectos desafiadores — exigindo não apenas conhecimento técnico, mas também habilidades interpessoais e visão sistêmica.

Conclui-se, portanto, que a especialização em Desenvolvimento Ágil de *Software* proporcionou uma base sólida e atualizada para a atuação profissional, permitindo a construção de soluções reais com foco em agilidade, qualidade e valor entregue ao usuário final. Mais do que um conjunto de ferramentas, o desenvolvimento ágil se consolidou como uma forma de pensar e agir diante da complexidade dos projetos modernos de *software*.

REFERÊNCIAS

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. **Agile Software Development Methods: Review and Analysis.** Espoo: VTT Publications, 2002. Disponível em: https://arxiv.org/pdf/1709.08439. Acesso em: 08 jul. 2025.

AGILE ALLIANCE. **What is Agile Software Development?** 2024. Disponível em: https://www.agilealliance.org/agile101/. Acesso em: 08 jul. 2025.

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

BECK, K.; ANDRES, C. **Extreme Programming Explained: Embrace Change.** 2. ed. Boston: Addison-Wesley, 2004. Disponível em: https://ptgmedia.pearsoncmg.com/images/9780321278654/samplepages/9780321278654.pdf. Acesso em: 08 jul. 2025.

BECK, K.; et al. **Manifesto for Agile Software Development.** 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 06 set. 2025.

CAROLI, P. **Lean Inception: How to Align People and Build the Right Product.** São Paulo: Caroli.org, 2018.

COCKBURN, A. **Agile Software Development: The Cooperative Game.** 2. ed. Boston: Addison-Wesley, 2006. Disponível em: https://www.researchgate.net/publication/235616359_Agile_Software_Development. Acesso em: 08 jul. 2025.

CODD, E. F. A relational model of data for large shared data banks. **Communications of the ACM**, New York, v. 13, n. 6, p. 377–387, jun. 1970.

COHN, M. **Succeeding with Agile: Software Development Using Scrum.** Boston: Addison-Wesley, 2009.

CRISPIN, L.; GREGORY, J. **Agile Testing: A Practical Guide for Testers and Agile Teams.** Boston: Addison-Wesley, 2009. Disponível em: https://scrummalaysia.com/media/attachments/2020/03/18/agile_testing_-_a_practical_guide_for_testers_and_agile_teams.pdf. Acesso em: 08 jul. 2025.

DATE, C. J. **An Introduction to Database Systems.** 8. ed. Boston: Addison-Wesley, 2003.

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

FOWLER, M. **Refactoring: Improving the Design of Existing Code.** 2. ed. Boston: Addison-Wesley, 2018. Disponível em:

https://dl.ebooksworld.ir/motoman/Refactoring.Improving.the.Design.of.Existing.Code.2nd.edition.www.EBooksWorld.ir.pdf. Acesso em: 08 jul. 2025.

GOOGLE. **Android Developers.** 2025a. Disponível em: https://developer.android.com/. Acesso em: 06 set. 2025.

GOOGLE. **Firebase Documentation.** 2025b. Disponível em: https://firebase.google.com/docs. Acesso em: 06 set. 2025.

GOOGLE. **Flutter Documentation.** 2025c. Disponível em: https://docs.flutter.dev/. Acesso em: 06 set. 2025.

GOOGLE. **Material Design.** [S. l.]: Google, 2025d. Disponível em: https://material.io/design. Acesso em: 06 set. 2025.

KANBAN BOARD GAME. **Kanban Board Game.** [S. l.]: Board Game Learning, 2025. Disponível em: https://kanbanboardgame.com/. Acesso em: 06 set. 2025.

MARTIN, R. C. **Agile Software Development: Principles, Patterns, and Practices.** Englewood Cliffs: Prentice Hall, 2002.

NORMAN, D. A. **Emotional Design: Why We Love (or Hate) Everyday Things.** New York: Basic Books, 2004.

OMG – Object Management Group. **Unified Modeling Language (UML)**, Version 2.5.1. 2017. Disponível em: https://www.omg.org/spec/UML/2.5.1/. Acesso em: 06 set. 2025.

PMI – Project Management Institute. **A Guide to the Project Management Body of Knowledge (PMBOK® Guide).** 6. ed. Newtown Square, PA: Project Management Institute, 2017.

POPPENDIECK, M.; POPPENDIECK, T. **Lean Software Development: An Agile Toolkit.** Boston: Addison-Wesley, 2003. Disponível em: https://ptgmedia.pearsoncmg.com/images/9780321150783/samplepages/0321150783.pdf. Acesso em: 08 jul. 2025.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide.** 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-BR.pdf. Acesso em: 06 set. 2025.

W3C. **Web Content Accessibility Guidelines (WCAG).** 2024. Disponível em: https://www.w3.org/WAI/standards-guidelines/wcag/. Acesso em: 06 set. 2025.