UNIVERSIDADE FEDERAL DO PARANÁ

KAREN ANDRESSA DE CARVALHO

MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO DESENVOLVIMENTO DE SOLUÇÕES WEB E MOBILE

CURITIBA

2025

KAREN ANDRESSA DE CARVALHO

MEMORIAL DE PROJETOS: INTEGRAÇÃO DE PRÁTICAS ÁGEIS NO DESENVOLVIMENTO DE SOLUÇÕES WEB E MOBILE

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 **KAREN ANDRESSA DE CARVALHO**, intitulada: **MEMORIAL DE PROJETOS: Integração de Práticas Ágeis no Desenvolvimento de Soluções Web e Mobile**, que após terem inquirido a aluna 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, 30 de Maio de 2025.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

O presente memorial tem como objetivo apresentar uma análise técnica dos projetos desenvolvidos ao longo do curso de Especialização em Desenvolvimento Ágil de Software da Universidade Federal do Paraná (UFPR). Os artefatos produzidos nas diferentes disciplinas evidenciam a aplicação prática dos princípios e métodos ágeis, com destaque para a integração entre modelagem, programação, gestão de projetos, testes e experiência do usuário. Por meio da realização de projetos como o sistema de gestão de condomínio, o aplicativo FlexSênior e a implementação de testes automatizados, foi possível vivenciar as etapas fundamentais do ciclo de desenvolvimento ágil. O uso de práticas como Kanban, elaboração de histórias de usuário, modelagem com UML, planejamento de releases, construção de interfaces responsivas e testes de software reforçam a aprendizagem incremental e colaborativa promovida pelo curso. Este memorial sintetiza, portanto, uma trajetória de aprendizagem baseada na consolidação de habilidades técnicas e conceituais para o desenvolvimento de software de acordo com os preceitos ágeis.

Palavras-chave: desenvolvimento ágil de software, métodos ágeis, experiência do usuário

ABSTRACT

This memorial aims to present a technical analysis of the projects developed throughout the Specialization Course in Agile Software Development at the Federal University of Paraná (UFPR). The artifacts produced in the different disciplines demonstrate the practical application of agile principles and methods, with emphasis on the integration of modeling, programming, project management, testing, and user experience. Through the execution of projects such as the condominium management system, the FlexSênior application, and the implementation of automated tests, it was possible to experience the fundamental stages of the agile development cycle. The use of practices such as Kanban, user story creation, UML modeling, release planning, responsive interface development, and software testing reinforces the incremental and collaborative learning promoted by the course. This memorial, therefore, synthesizes a learning journey based on the consolidation of technical and conceptual skills for software development aligned with agile principles.

Keywords: agile software development, agile methods, user experience

SUMÁRIO

1 PARECER TÉCNICO.....7
2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE.....10
2.1 ARTEFATOS DO PROJETO.....10
3 DISCIPLINA: MAG1 E MAG2 – MODELAGEM ÁGIL DE SOFTWARE 1 E 2.....12
3.1 ARTEFATOS DO PROJETO.....12
4 DISCIPLINA: GAP1 E GAP2 – GERENCIAMENTO ÀGIL DE PROJETOS DE SOFTWARE 1 E 2 17
4.1 ARTEFATOS DO PROJETO.....17
5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO.....20
5.1 ARTEFATOS DO PROJETO.....20
6 DISCIPLINA: BD – BANCO DE DADOS.....22
6.1 ARTEFATOS DO PROJETO.....22
7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO 27
7.1 ARTEFATOS DO PROJETO.....27
8 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE.....30
8.1 ARTEFATOS DO PROJETO.....31
9 DISCIPLINA: INFRA - INFRAESTRUTURA PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SOFTWARE (DEVOPS) 34
8.2 ARTEFATOS DO PROJETO.....34
9 DISCIPLINA: TEST – TESTES AUTOMATIZADOS 37
9.1 ARTEFATOS DO PROJETO.....37
10 CONCLUSÃO 40
11 REFERÊNCIAS.....41

1 PARECER TÉCNICO

Durante o curso de Especialização em Desenvolvimento Ágil de Software, as disciplinas ofertadas proporcionaram uma imersão prática nos diversos aspectos envolvidos no ciclo de vida do desenvolvimento ágil. Desde a concepção e modelagem de sistemas, passando por planejamento e gestão de projetos, até a construção, testes e implantação de soluções. Os projetos desenvolvidos evidenciam uma progressiva articulação entre teoria e prática com base nos princípios do Manifesto Ágil (BECK et al., 2001).

Um dos destaques do curso foi a ênfase em práticas iterativas e incrementais, vivenciadas por meio da utilização de métodos ágeis como *Kanban*, *Scrum* e *Extreme Programming* (XP), articuladas com práticas técnicas como *Test Driven Development* (TDD), modelagem orientada a objetos, testes automatizados e princípios de *User Experience* (UX). A disciplina de Modelagem Ágil de Software I e II (MAG1 e MAG2) contribuiu fortemente para a compreensão dos requisitos e estrutura de sistemas por meio de artefatos como diagramas de caso de uso, classes e sequência, os quais foram essenciais para a posterior implementação de funcionalidades com clareza e coesão arquitetural.

A disciplina Métodos Ágeis para Desenvolvimento de Software (MADS) foi uma das primeiras do curso e teve como foco a introdução aos conceitos fundamentais das metodologias ágeis. A principal atividade realizada foi a construção de um mapa conceitual, que ajudou a organizar e visualizar os principais frameworks, valores e práticas ágeis. Essa atividade serviu como base para compreender o pensamento ágil e facilitou a conexão com os conteúdos das disciplinas seguintes.

A disciplina de Gerenciamento Ágil de Projetos (GAP1 e GAP2) proporcionou uma vivência realista do planejamento por meio da construção de releases, organização de sprints e estimativas em pontos de história. O desenvolvimento de planos detalhados e a priorização das histórias de usuário fomentaram a visão de entrega de valor contínuo, considerando as limitações de tempo e recursos. Essas práticas, utilizadas em conjunto com os artefatos de modelagem, criaram um fluxo coerente entre planejamento e execução.

Na área de implementação, a disciplina de Introdução à Programação (INTRO) permitiu exercitar fundamentos essenciais da construção de soluções *back-end*, com foco na confiabilidade e qualidade do código. Foram explorados testes automatizados

e boas práticas de codificação, alinhando-se com os princípios de qualidade e *feedback* contínuo do desenvolvimento ágil. Esse conteúdo foi aprofundado em Aspectos Ágeis da Programação (AAP), disciplina responsável por tratar do uso de versionamento com Git, integração contínua, refatoração e princípios SOLID, favorecendo o alinhamento entre arquitetura e manutenção sustentável do *software*.

O componente de Banco de Dados (BD) trouxe solidez ao conhecimento técnico com a aplicação de boas práticas de modelagem conceitual e lógica, construção de scripts SQL e aplicação de restrições de integridade. A clareza na definição dos relacionamentos (1:1, 1:N, N:N), uso de *constraints* e normalização das estruturas de dados foram fundamentais para garantir a consistência entre as regras de negócio e a camada de persistência, integrando-se perfeitamente às demais disciplinas que envolvem implementação e modelagem.

Na dimensão da experiência do usuário, a disciplina de UX permitiu aplicar o design centrado no usuário por meio do desenvolvimento do aplicativo FlexSênior, voltado para pessoas idosas. A proposta considerou princípios de acessibilidade, usabilidade, responsividade e empatia, explorando a psicologia das cores, hierarquia tipográfica, botões com espaçamento adequado e feedbacks visuais. O processo foi enriquecido com a coleta de *feedbacks* reais de usuários, promovendo ajustes iterativos no design e reforçando o princípio ágil de colaboração com o cliente.

As disciplinas de Desenvolvimento Web I e II (WEB1 e WEB2) permitiram consolidar o ciclo de desenvolvimento completo de aplicações web, envolvendo HTML, CSS, *JavaScript* e frameworks modernos como o Angular. Embora os arquivos finais não tenham sido incluídos no memorial, a integração com os demais projetos evidencia a aplicação prática dos conhecimentos adquiridos, especialmente no que se refere à construção de interfaces navegáveis e responsivas em conjunto com o *back-end.*

Na mesma linha, os componentes de Desenvolvimento Mobile I e II (MOB1 e MOB2) potencializaram a criação de soluções voltadas para múltiplos dispositivos, com atenção especial à responsividade e portabilidade. O próprio FlexSênior, apesar de construído inicialmente para navegadores, reflete preocupações típicas do desenvolvimento mobile, como navegação intuitiva, tamanho de botões e uso eficiente do espaço visual.

A disciplina de Infraestrutura e *DevOps* (INFRA) permitiu a aplicação prática de ferramentas essenciais para o ciclo de entrega contínua de *software*, como *Docker*,

Git, GitHub e GitLab com *Jenkins*. Entre as atividades realizadas, destacam-se a criação e execução de *containers Docker* com integração *Jenkins-GitLab*, o versionamento de código com *Git* e *push* para repositórios remotos, além da organização e monitoramento de projetos em ambientes isolados. Essas práticas demonstram domínio sobre automação, rastreabilidade e escalabilidade das soluções desenvolvidas, alinhando-se aos princípios ágeis de integração contínua e entrega incremental. A disciplina se conectou diretamente com disciplinas como AAP e TEST, ao sustentar *pipelines* de testes automatizados e promover uma cultura de qualidade desde as fases iniciais do desenvolvimento, fortalecendo a autonomia técnica e a confiabilidade dos projetos entregues.

Por fim, a disciplina de Testes Automatizados (TEST) foi essencial para consolidar a cultura de qualidade desde as fases iniciais do desenvolvimento. O projeto apresentado utilizou a ferramenta *Playwright* para automatizar a simulação de interações com o sistema, evidenciando o domínio de testes de interface e validações automatizadas. Essa abordagem garante ciclos de entrega mais curtos e seguros, promovendo confiabilidade e facilitando o trabalho colaborativo em equipes ágeis.

Dessa forma, o conjunto dos projetos e disciplinas deste curso de especialização demonstram uma trajetória de evolução técnica e metodológica da autora, fortemente ancorada nos valores do desenvolvimento ágil. A integração entre teoria e prática foi facilitada por uma abordagem orientada a projetos, que permitiu vivenciar situações reais, resolver problemas concretos e consolidar habilidades alinhadas às necessidades do mercado contemporâneo de *software*.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

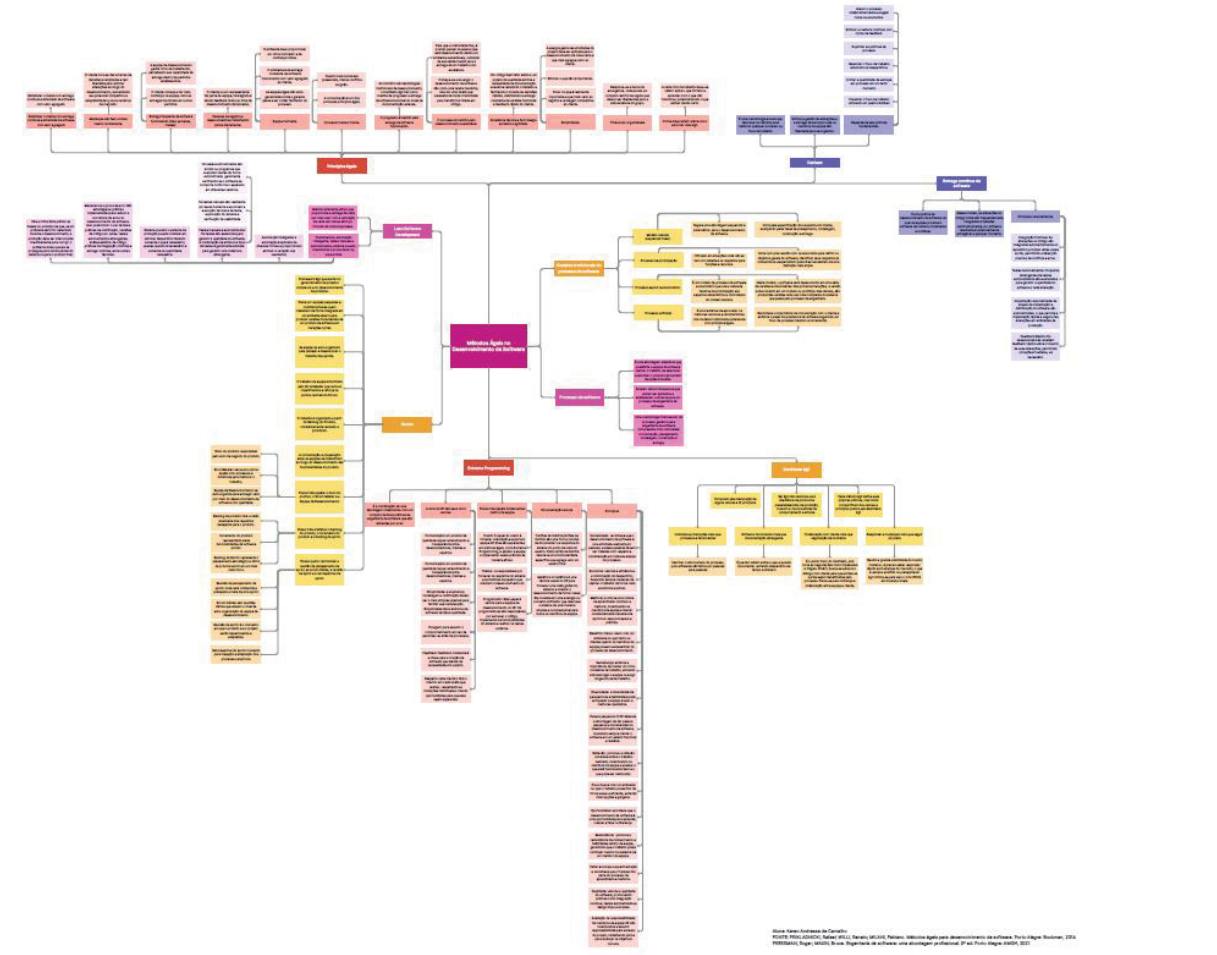
A disciplina de Métodos Ágeis para Desenvolvimento de Software (MADS) teve como foco promover uma compreensão conceitual estruturada sobre as principais abordagens e práticas utilizadas em metodologias ágeis, por meio da elaboração de um mapa mental sintético e analítico. O projeto final consistiu na construção de um material visual que organiza os fundamentos das metodologias ágeis, explorando elementos como princípios do Manifesto Ágil, frameworks como Scrum, Kanban e XP, papéis, cerimônias, artefatos e boas práticas do desenvolvimento incremental e iterativo.

O mapa mental, elaborado com base em diferentes fontes teóricas, apresenta uma visão abrangente e interconectada dos conceitos fundamentais da agilidade. Sua estrutura gráfica favorece o raciocínio visual e a memorização de conteúdos, permitindo compreender as relações entre práticas como planejamento de sprints, gestão de backlog, métricas (lead time, throughput, WIP), refatoração contínua, integração contínua e testes automatizados. O exercício de síntese e categorização dos temas foi essencial para construir uma base sólida sobre a cultura ágil, sua aplicabilidade e impacto no ciclo de desenvolvimento de software.

A importância dessa disciplina se manifesta na articulação direta com todas as demais do curso, servindo como ponto de partida para o entendimento e aplicação das práticas ágeis nos projetos desenvolvidos em modelagem (MAG), gestão de projetos (GAP), testes (TEST), infraestrutura (INFRA), UX e desenvolvimento web/mobile. Ao consolidar os pilares teóricos da agilidade, MADS fortaleceu o embasamento crítico necessário para decisões técnicas, organização do trabalho em equipe e entrega contínua de valor ao longo da especialização.

2.1 ARTEFATOS DO PROJETO

Figura 1 - Mapa conceitual sobre os conceitos dos métodos ágeis



Fonte: A autora, 2025

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

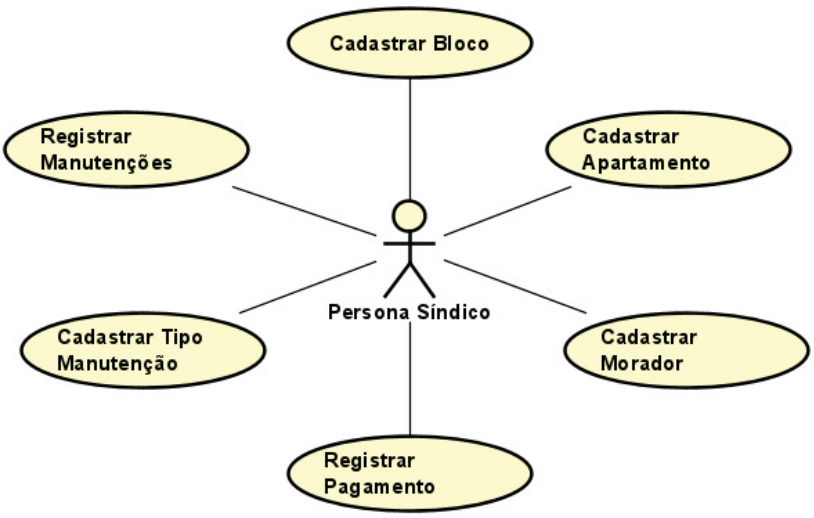
As disciplinas de Modelagem Ágil de *Software* I e II (MAG1 e MAG2) foram fundamentais para a construção da base estrutural dos projetos desenvolvidos ao longo da especialização. Através da aplicação prática de técnicas de modelagem orientada a objetos, foi possível transformar requisitos em representações visuais organizadas, compreensíveis e alinhadas às boas práticas de engenharia de *software*. O projeto central dessas disciplinas foi o desenvolvimento do sistema de Gestão de Condomínio, com foco na modelagem completa das funcionalidades do sistema.

A construção começou com a elaboração de diagramas de caso de uso (nível 1 e 2), que permitiram mapear as interações dos usuários com o sistema e identificar os principais fluxos de funcionalidade. Em seguida, foram produzidos diagramas de classes, que detalham as estruturas de dados e os relacionamentos entre objetos, e diagramas de sequência, que ilustram o comportamento dinâmico das interações no sistema. Também foram desenvolvidas histórias de usuário, com critérios de aceitação e fluxos bem definidos, servindo como ponte entre a visão do usuário e a implementação técnica.

O trabalho desenvolvido em MAG1 e MAG2 teve papel central na integração com outras disciplinas. Serviu de base para o planejamento de entregas em sprints, trabalhado em GAP1 e GAP2, forneceu insumos para a implementação na disciplina de Introdução à Programação (INTRO) e orientou a modelagem do banco de dados (BD). Além disso, os fluxos definidos nos diagramas facilitaram o desenvolvimento das interfaces em UX e os testes automatizados realizados em TEST. Dessa forma, as disciplinas de modelagem cumpriram seu papel estruturante, permitindo que os projetos tivessem coesão, clareza e alinhamento com os princípios do desenvolvimento ágil, como colaboração contínua, documentação leve e design orientado a mudanças.

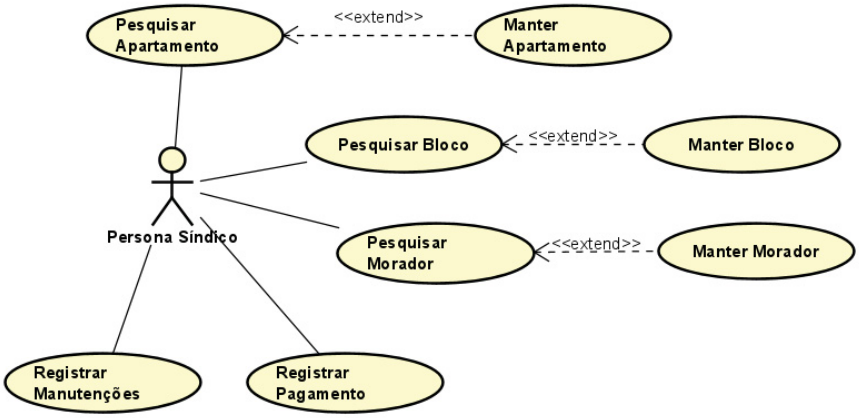
3.1 ARTEFATOS DO PROJETO

Figura 2 - Diagrama de caso de uso nível 1



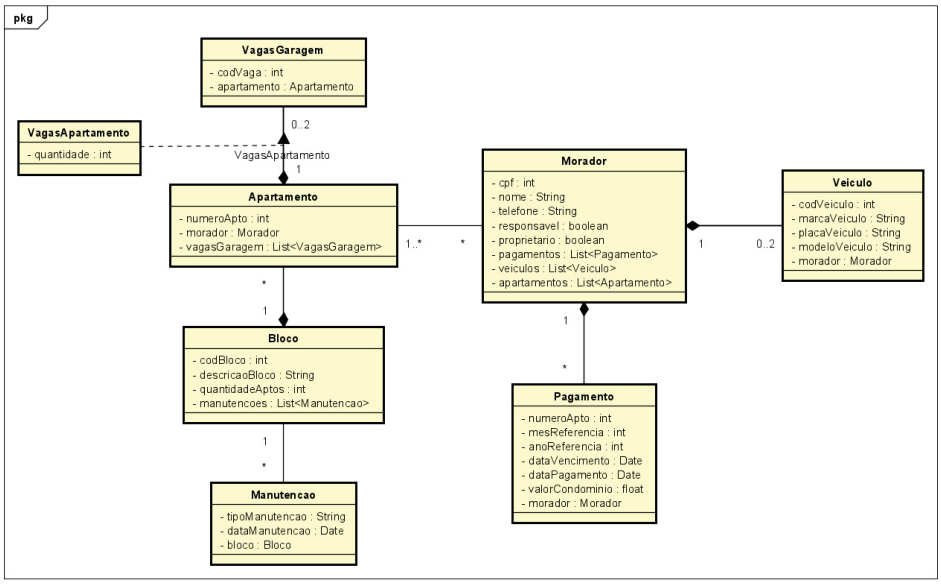
Fonte: A autora, 2025

Figura 3 - Diagrama de caso de uso nível 2



Fonte: A autora, 2025

Figura 4 - Diagrama de classes



Fonte: A autora, 2025

Histórias de usuário

HU001 – Pesquisar Bloco

SENDO	O Síndico
QUERO	Pesquisar os blocos cadastrados no sistema
PARA	Manter os registros atualizados

Desenho da tela:

Figura 5 - Tela desenvolvida para o sistema de gestão de condomínios

Fonte: A autora, 2025

Lista de critérios de aceitação:

1. Deve carregar os dados dos blocos já cadastrados em uma tabela.
2. Deve permitir a pesquisa de blocos cadastrados na tabela da tela.
3. Deve permitir o cadastro de um novo bloco.
4. Deve permitir alterar um bloco cadastrado.
5. Deve permitir excluir um bloco cadastrado.
6. Deve permitir consultar os dados de um bloco cadastrado.
7. Deve voltar a tela anterior.

Critérios de aceitação – detalhamento:

1. Deve carregar os dados dos blocos já cadastrados em uma tabela.

Dado que	
Quando	A tela é acionada.
Então	O sistema carrega os dados dos blocos já cadastrados em uma tabela

2. Deve permitir a pesquisa de blocos cadastrados na tabela da tela.

Dado que	Os blocos estão na tabela
Quando	For informado um argumento de pesquisa
Então	O sistema filtra e apresenta na tela somente os blocos que obedeçam ao critério.

3. Deve permitir o cadastro de um novo bloco.

Dado que	Os blocos estão na tabela
Quando	O botão “Novo bloco” é pressionado
Então	O sistema chama a HU002 – Manter Bloco passando o parâmetro “Novo”

4. Deve permitir alterar um bloco cadastrado.

Dado que	Os blocos estão na tabela
Quando	O botão “Alterar” é pressionado em uma determinada linha

Então	O sistema chama a HU002 – Manter Bloco passando o parâmetro “Alterar” e os dados do bloco da linha.

5. Deve permitir excluir um bloco cadastrado.

Dado que	Os blocos estão na tabela
Quando	O botão “Excluir” é pressionado em uma determinada linha
Então	O sistema pede uma confirmação, exclui o bloco da base de dados e refaz a tabela da tela lendo novamente o banco de dados.

6. Deve permitir consultar os dados de um bloco cadastrado.

Dado que	Os blocos estão na tabela
Quando	O usuário clica em qualquer dado de um bloco em uma determinada linha
Então	O sistema chama a HU002 – Manter Bloco passando o parâmetro “Consultar” e os dados do bloco da linha.

7. Deve voltar a tela anterior.

Dado que	Os blocos estão na tabela
Quando	O botão “voltar” for pressionado
Então	O sistema volta para a tela anterior

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

As disciplinas de Gerenciamento Ágil de Projetos I e II (GAP1 e GAP2) proporcionaram uma imersão prática nos métodos de planejamento, acompanhamento e entrega de projetos com base nos frameworks ágeis, principalmente o *Scrum*. O projeto desenvolvido consistiu na construção de um Plano de *Release* para o sistema de Gestão de Condomínio, com definição de sprints, datas de entrega, priorização de funcionalidades e estimativas de esforço em pontos de história. Foram estruturadas cinco *sprints* com alocação de histórias de usuário como "Pesquisar bloco", "Cadastrar morador" e "Registrar manutenção", cada uma acompanhada de critérios de aceitação e estimativas baseadas na velocidade da equipe. O planejamento das entregas permitiu simular o trabalho de um time ágil real, reforçando a importância da definição clara de escopo, da medição contínua de desempenho e da adaptação durante o ciclo do projeto.

Além disso, em GAP2 foi realizada a simulação do *Kanban Board Game*, que trouxe uma nova perspectiva sobre a gestão de fluxo. A atividade permitiu observar gargalos no processo, a importância do limite de trabalho em progresso (WIP) e a variação do *throughput* entre os ciclos. O uso do *Cumulative Flow Diagram* (CFD) reforçou a importância da análise de métricas visuais para identificar ineficiências e apoiar a tomada de decisão baseada em dados. Essa experiência foi essencial para compreender como adaptar o processo de trabalho à capacidade real do time, mantendo a entrega contínua de valor.

Este projeto se integrou de maneira direta com os artefatos desenvolvidos em outras disciplinas. As histórias de usuário foram baseadas nos requisitos levantados em MAG1 e MAG2. Assim, GAP1 e GAP2 possibilitaram o desenvolvimento da visão estratégica do projeto, alinhando teoria e prática com foco em entregas incrementais e melhoria contínua, pilares fundamentais do desenvolvimento ágil de *software*.

4.1 ARTEFATOS DO PROJETO

Figura 6 - Plano de *release* (GAP1)

Cálculo da Velocidade:

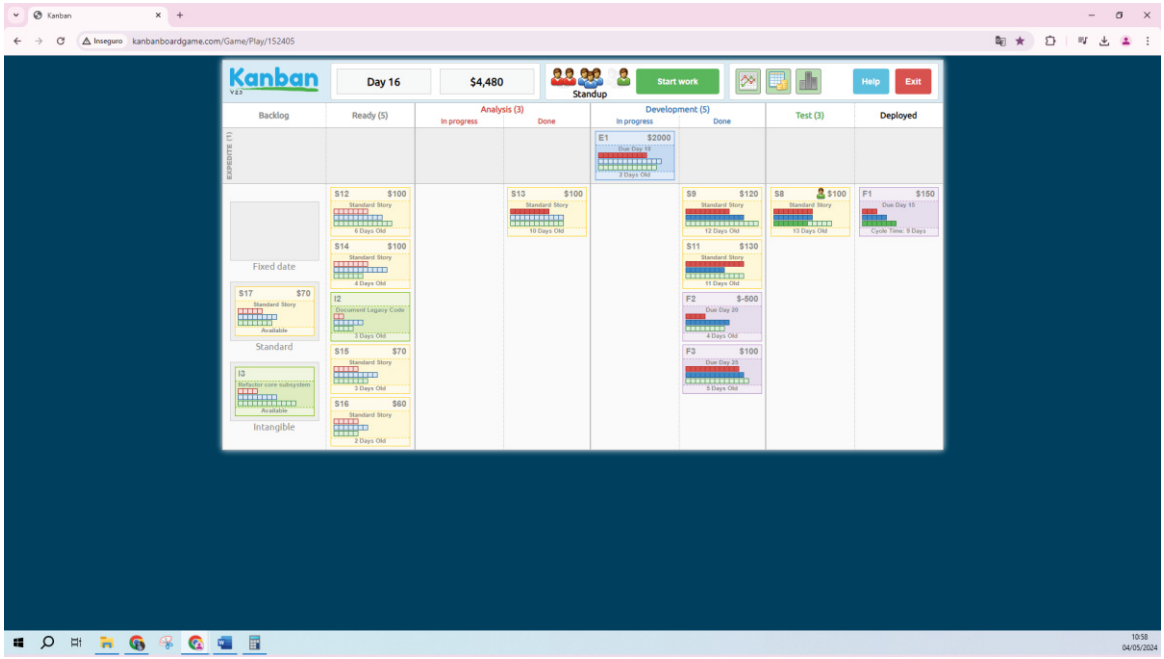
Horas disponíveis por dia: 8 horas		Tamanho da Sprint: 2 semanas	
Horas disponíveis por Sprint: 80h		Velocidade: 10	

Plano de Release:

Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4	Iteração/Sprint 5
Data Início: 06/05/2024	Data Início: 20/05/2024	Data Início: 03/06/2024	Data Início: 17/06/2024	Data Início: 01/07/2024
Data Fim: 17/05/2024	Data Fim: 31/05/2024	Data Fim: 14/06/2024	Data Fim: 28/06/2024	Data Fim: 12/07/2024
HU001 – Pesquisar bloco SENDO o síndico QUERO Pesquisar os blocos cadastrados no sistema PARA manter os registros atualizados ESTIMATIVA 2 pontos	HU003 – Pesquisar apartamento SENDO o síndico QUERO pesquisar os apartamentos cadastrados no sistema PARA manter os registros atualizados ESTIMATIVA 2 pontos	HU005 – Pesquisar morador SENDO o síndico QUERO pesquisar os moradores cadastrados no sistema PARA manter os registros atualizados ESTIMATIVA 2 pontos	HU008 – Cadastrar tipo manutenção SENDO o síndico QUERO cadastrar um novo tipo de manutenção PARA registrar uma manutenção realizada no condomínio ESTIMATIVA 3 pontos	HU007 – Registrar pagamento Parte 2 SENDO o síndico QUERO registrar o pagamento dos condôminos PARA estarem regularizados perante o condomínio ESTIMATIVA 8 pontos
HU002 – Manter bloco SENDO o síndico QUERO manter os dados dos blocos PARA que seus dados fiquem atualizados ESTIMATIVA 8 pontos	HU004 – Manter apartamento SENDO o síndico QUERO cadastrar um novo apartamento PARA que seus dados fiquem atualizados ESTIMATIVA 8 pontos	HU006 – Manter morador SENDO o síndico QUERO cadastrar um novo morador PARA estarem registrados no condomínio ESTIMATIVA 8 pontos	HU009 – Registrar manutenções SENDO o síndico QUERO registrar uma manutenção realizada no condomínio PARA manter o registro atualizado ESTIMATIVA 3 pontos	
			HU007 – Registrar pagamento Parte 1 SENDO o síndico QUERO registrar o pagamento dos condôminos PARA estarem regularizados perante o condomínio ESTIMATIVA 4 pontos	

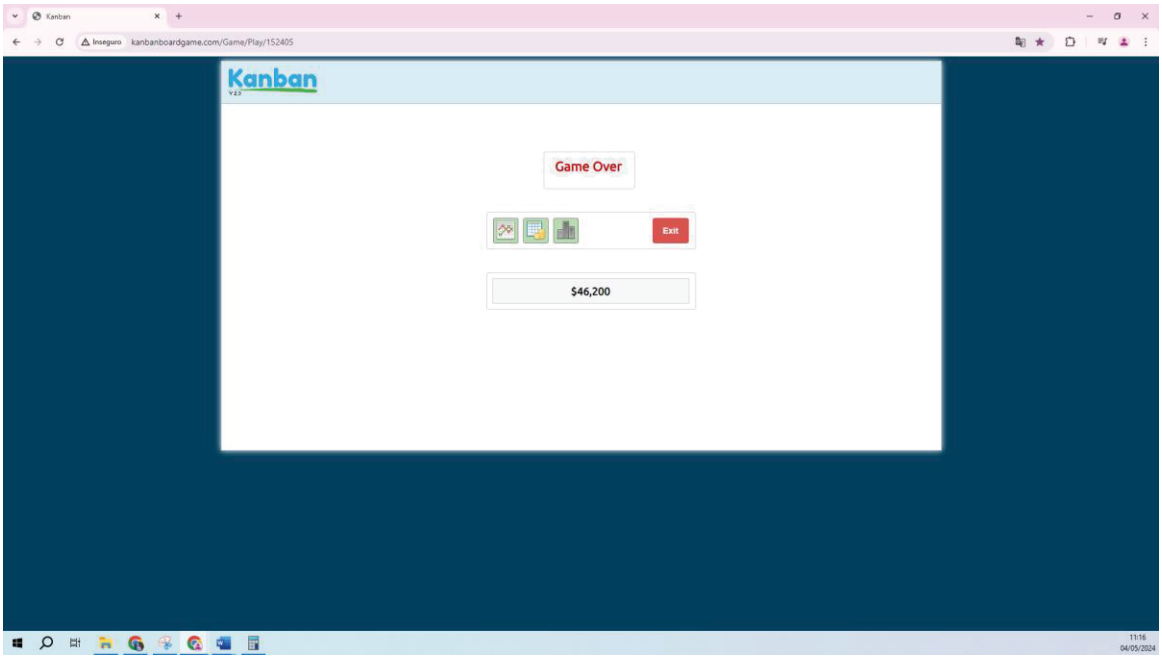
Fonte: A autora, 2025

Figura 7 - Cópia de tela do *Kanban Board Game* dia 15(GAP2)



Fonte: A autora, 2025

Figura 8 - Cópia de tela da conclusão do *Kanban Board Game* dia 35 (GAP2))



Fonte: A autora, 2025

Figura 9 - Cópia de tela do CFD ao final do dia 35 (GAP2)

Fonte: A autora, 2025

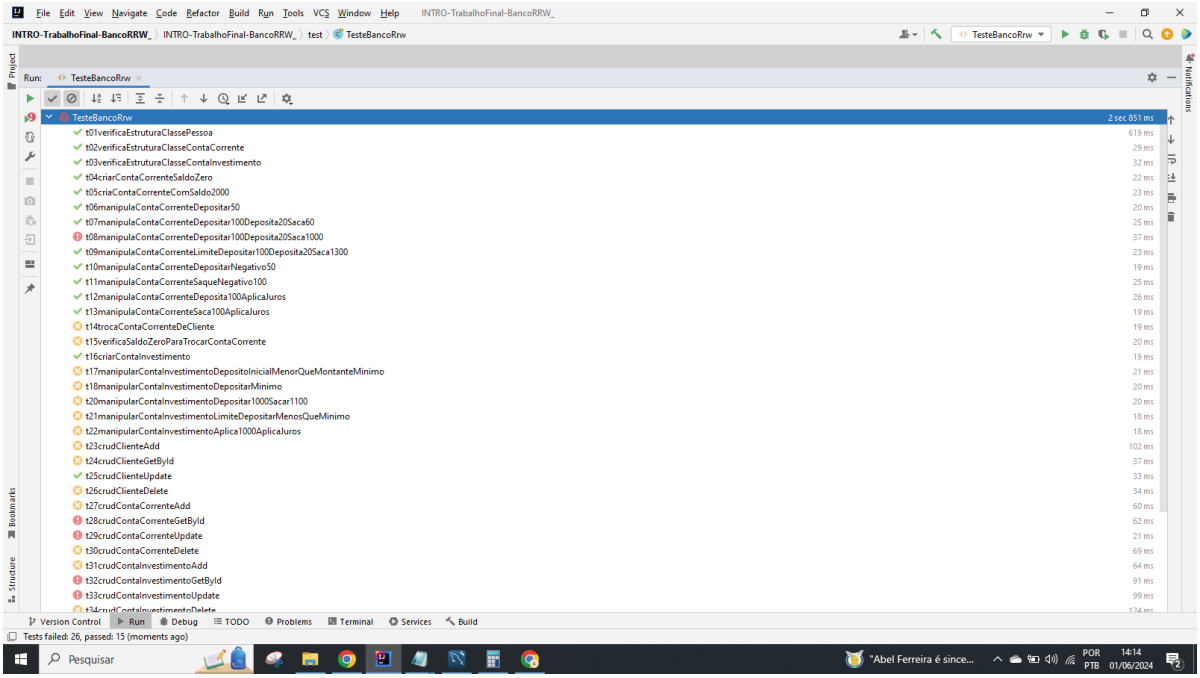
5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação (INTRO) foi responsável por consolidar os fundamentos da lógica de programação e da construção de código funcional, com foco na implementação de soluções orientadas a regras de negócio e validação por testes. O projeto desenvolvido consistiu na criação de funcionalidades de *back-end* com registro de testes automatizados, abordando a importância da escrita de código limpo, compreensível e confiável desde as etapas iniciais do ciclo de desenvolvimento.

Os testes desenvolvidos nesta disciplina desempenharam papel pedagógico fundamental ao promoverem o raciocínio lógico, a modularização de soluções e a detecção precoce de falhas. A prática da automatização de testes contribuiu diretamente para a assimilação de conceitos utilizados nas demais disciplinas, como a integração com banco de dados (BD), o controle de versões com Git (AAP) e a garantia de qualidade reforçada posteriormente em TEST.

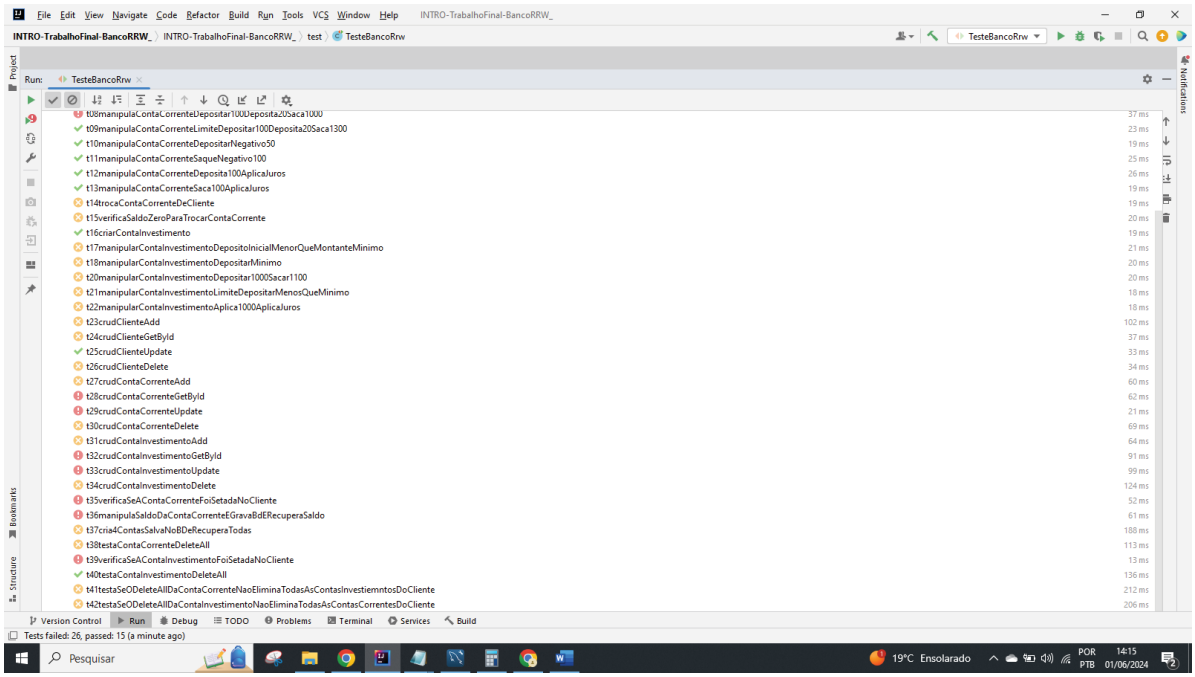
5.1 ARTEFATOS DO PROJETO

Figura 10 - Testes realizados no sistema bancário (INTRO)



Fonte: A autora, 2025

Figura 11 - Testes realizados no sistema bancário (INTRO)



Fonte: A autora, 2025

6 DISCIPLINA: BD – BANCO DE DADOS

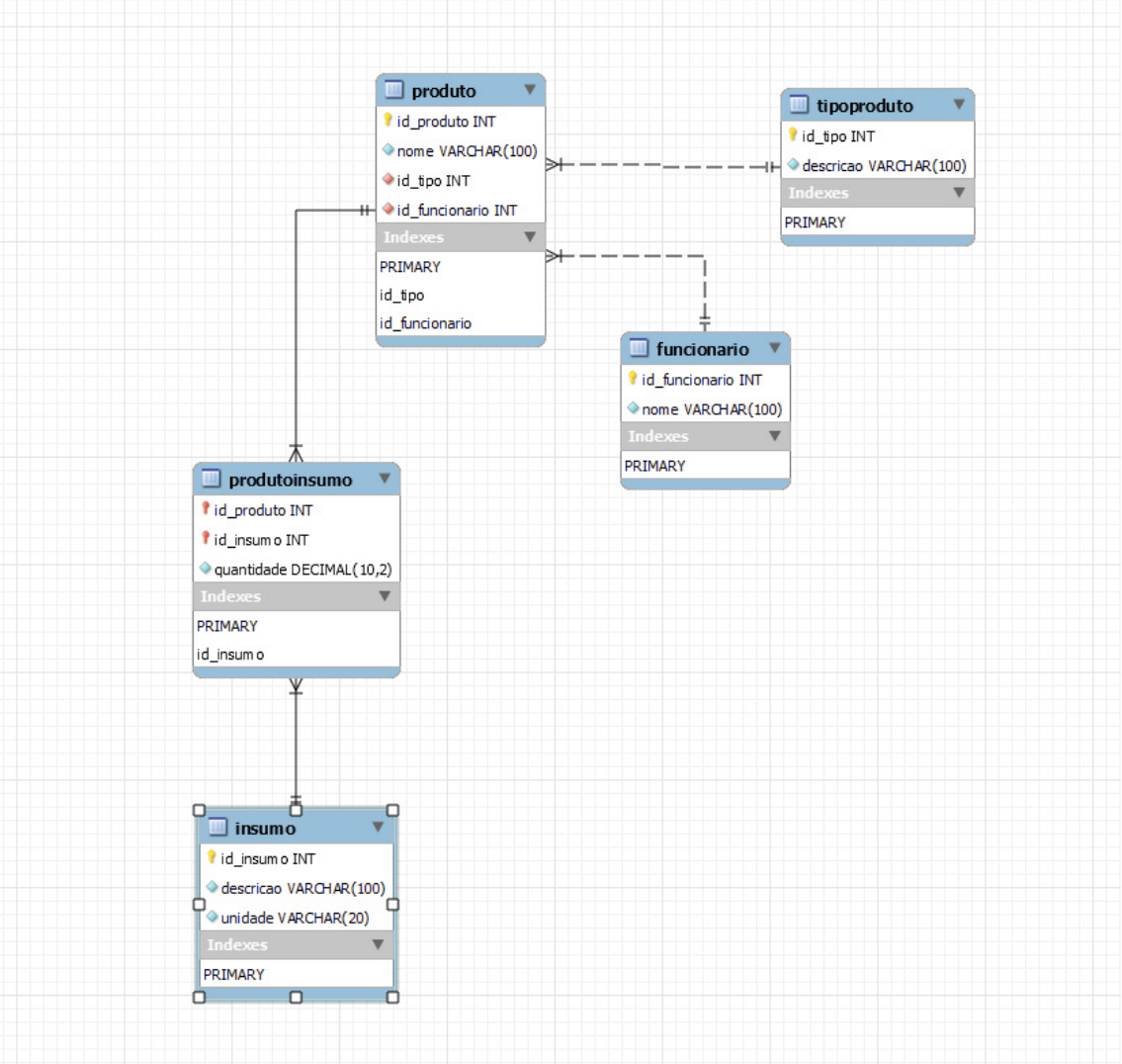
A disciplina de Banco de Dados (BD) foi essencial para a construção de soluções robustas e com integridade na persistência de informações, aspecto indispensável no desenvolvimento ágil de *software*. O projeto realizado envolveu a modelagem e implementação de um sistema de controle de produção em uma fábrica, incluindo desde o modelo entidade-relacionamento (MER) até o modelo lógico relacional, com posterior criação das tabelas e inserção de dados utilizando a linguagem SQL.

A proposta contemplou o uso de restrições de integridade (CHECK, NOT NULL), chaves primárias e estrangeiras, e a definição de relacionamentos 1:1, 1:N e N:N, evidenciando domínio sobre boas práticas de normalização e estruturação de dados. Além disso, a tabela associativa “ProdutoInsumo” implementou com sucesso o controle de quantidade com validações, refletindo preocupações com regras de negócio e integridade referencial. A aplicação desses conceitos garante a consistência e segurança dos dados manipulados pelas aplicações.

A disciplina dialoga fortemente com os conteúdos de MAG1 e MAG2, que forneceram os requisitos e estruturas conceituais das entidades, e com INTRO e AAP, que exigiram interação com o banco de dados por meio de instruções SQL na aplicação. No contexto do desenvolvimento ágil, o domínio sobre banco de dados é fundamental para garantir entregas incrementais e adaptáveis, mantendo a integridade das informações mesmo com mudanças frequentes de requisitos. A clareza e organização do projeto desenvolvido demonstram alinhamento com os princípios de simplicidade, design técnico sólido e entrega contínua de valor.

6.1 ARTEFATOS DO PROJETO

Figura 12 - Modelo lógico - Diagrama de Entidade e Relacionamento



Fonte: A autora, 2025

Script SQL para criação das tabelas

```
    -- FUNCIONÁRIOS DA FÁBRICA
CREATE TABLE Funcionario (
    id_funcionario INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL
);

-- TIPOS DE PRODUTO (Ex: Eletrônico, Alimentício etc.)
CREATE TABLE TipoProduto (
    id_tipo INT PRIMARY KEY,
```

```
    descricao VARCHAR(100) NOT NULL
);

-- PRODUTOS FABRICADOS
CREATE TABLE Produto (
    id_produto INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    id_tipo INT NOT NULL,
    id_funcionario INT NOT NULL,
    FOREIGN KEY (id_tipo) REFERENCES TipoProduto(id_tipo)
        ON UPDATE CASCADE ON DELETE RESTRICT,
    FOREIGN KEY (id_funcionario) REFERENCES
Funcionario(id_funcionario)
        ON UPDATE CASCADE ON DELETE RESTRICT
);

-- INSUMOS UTILIZADOS NA PRODUÇÃO
CREATE TABLE Insumo (
    id_insumo INT PRIMARY KEY,
    descricao VARCHAR(100) NOT NULL,
    unidade VARCHAR(20) NOT NULL,
    CHECK (unidade IN ('kg', 'L', 'g', 'mL', 'un'))
);

-- ASSOCIAÇÃO ENTRE PRODUTOS E INSUMOS (N:N)
CREATE TABLE ProdutoInsumo (
    id_produto INT,
    id_insumo INT,
    quantidade DECIMAL(10,2) NOT NULL CHECK (quantidade > 0),
    PRIMARY KEY (id_produto, id_insumo),
    FOREIGN KEY (id_produto) REFERENCES Produto(id_produto)
        ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (id_insumo) REFERENCES Insumo(id_insumo)
```

```
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

Inserção de dados:

```
     Inserção de Dados -- FUNCIONÁRIOS (1:1 com Produto)
INSERT INTO Funcionario (id_funcionario, nome) VALUES
(1, 'Ana Costa'),
(2, 'Carlos Lima'),
(3, 'João Pereira'),
(4, 'Marina Dias'),
(5, 'Paulo Souza');

-- TIPOS DE PRODUTO (1:N com Produto)
INSERT INTO TipoProduto (id_tipo, descricao) VALUES
(1, 'Eletrônico'),
(2, 'Alimentício'),
(3, 'Têxtil'),
(4, 'Químico'),
(5, 'Automotivo');

-- INSUMOS (N:N com Produto via ProdutoInsumo)
INSERT INTO Insumo (id_insumo, descricao, unidade) VALUES
(1, 'Plástico', 'kg'),
(2, 'Açúcar', 'kg'),
(3, 'Algodão', 'kg'),
(4, 'Essência Floral', 'L'),
(5, 'Metal', 'kg');

-- PRODUTOS (relacionando TipoProduto e Funcionario)
INSERT INTO Produto (id_produto, nome, id_tipo,
id_funcionario) VALUES
(1, 'Geladeira FrostX', 1, 1),
(2, 'Chocolate Meio Amargo', 2, 2),
```

```
(3, 'Camiseta Algodão', 3, 3),
(4, 'Shampoo Floral', 4, 4),
(5, 'Freio ABS 5000', 5, 5);

-- PRODUTO x INSUMO (associação N:N)
INSERT INTO ProdutoInsumo (id_produto, id_insumo, quantidade)
VALUES
-- Produto 1: Geladeira usa Plástico e Metal
(1, 1, 4.5),
(1, 5, 2.0),

-- Produto 2: Chocolate usa Açúcar
(2, 2, 3.5),

-- Produto 3: Camiseta usa Algodão
(3, 3, 2.0),

-- Produto 4: Shampoo usa Essência
(4, 4, 1.5),

-- Produto 5: Freio usa Metal e Plástico
(5, 5, 3.0),
(5, 1, 1.0);
```

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis de Programação (AAP) teve como foco o aprimoramento das práticas de codificação sustentáveis e alinhadas aos princípios ágeis, com ênfase na clareza, organização e qualidade do código. O projeto desenvolvido foi a refatoração do algoritmo de ordenação Bubble Sort em Java, com a aplicação de boas práticas de programação, como remoção de parâmetros desnecessários, uso de funções auxiliares para modularização, padronização de nomes de variáveis e substituição de estruturas por versões mais legíveis e seguras, como o uso de for-each no método de impressão do array.

Essa atividade reforçou conceitos fundamentais para o desenvolvimento ágil, como a manutenibilidade do código, a simplicidade na estruturação lógica e a reutilização de componentes, tornando o sistema mais confiável e adaptável. A refatoração, ainda que aplicada a um algoritmo clássico, ilustra a preocupação com o design limpo e com a legibilidade — pilares defendidos tanto pelo Manifesto Ágil quanto por autores como Martin Fowler e Robert C. Martin.

A disciplina também dialogou com outras áreas do curso, como TEST, ao preparar o código para possíveis testes automatizados; INFRA, por incentivar a estruturação de funções que favorecem integração contínua; e INTRO, por consolidar práticas fundamentais de programação. A aplicação prática desses princípios técnicos reforçou o papel da AAP como eixo transversal da especialização, contribuindo para a entrega de software com qualidade e preparada para evolução constante.

7.1 ARTEFATOS DO PROJETO

```
     class BubbleSort {

    //remoção do parâmetro n, pois o tamanho do array pode ser
obtido a partir do método length, conforme linha 7
    static void bubbleSort(int[] arr)
    {
        int n = arr.length;
        boolean swapped; //padronização do nome das variáveis
```

```
para inglês
        for (int i = 0; i < n - 1; i++) {
            swapped = false;

            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    swap(arr, j, j + 1);//aplicação da função
auxiliar de troca
                    swapped = true;
                }
            }
            if (!swapped)
                break;
        }
    }
    //adição de uma função auxiliar para armazenar a lógica da
troca, permitindo sua reutilização, se necessário
    private static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    static void printArray(int[] arr) {
        for (int element : arr) { //uso do for avançado para
eliminar a necessidade de gerenciamento de índices
            System.out.print(element + " ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
        bubbleSort(arr);
```

```
        System.out.println("Array ordenado: ");
        printArray(arr);
    }
}
```

8 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

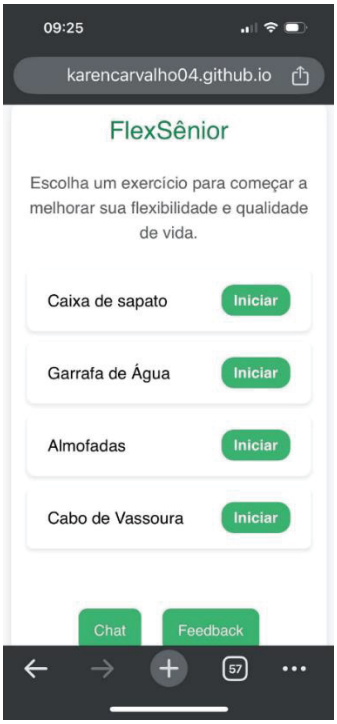
A disciplina de *UX* no Desenvolvimento Ágil de *Software* possibilitou a aplicação de práticas de *design* centrado no usuário, acessibilidade e usabilidade em um contexto real de projeto. O produto desenvolvido foi o FlexSênior, um aplicativo voltado para pessoas idosas com o objetivo de incentivar a prática de exercícios físicos com utensílios domésticos. A proposta atendeu a uma necessidade concreta de saúde e bem-estar desse público, ao mesmo tempo em que seguiu os princípios do desenvolvimento ágil, especialmente a entrega de valor contínuo com foco no usuário.

Durante o desenvolvimento, foram exploradas técnicas de prototipagem, definição de fluxos de navegação e escolha de elementos visuais com base em fundamentos da psicologia das cores, contraste e legibilidade. As telas do sistema foram pensadas para facilitar o uso por idosos, utilizando botões grandes, linguagem acessível, estrutura simples e vídeos demonstrativos. O projeto também contou com *feedback* de três usuários idosos reais, cujas sugestões foram analisadas e integradas ao processo de melhoria contínua do produto.

O FlexSênior demonstrou como a aplicação dos princípios de *UX* pode gerar produtos mais acessíveis, funcionais e acolhedores, especialmente para públicos com maior vulnerabilidade digital. A experiência prática nesta disciplina consolidou a compreensão de que desenvolvimento ágil de *software* não se limita à codificação, mas inclui a escuta ativa do usuário, prototipação rápida, testes de usabilidade e adaptação contínua — pilares essenciais de projetos centrados em pessoas.

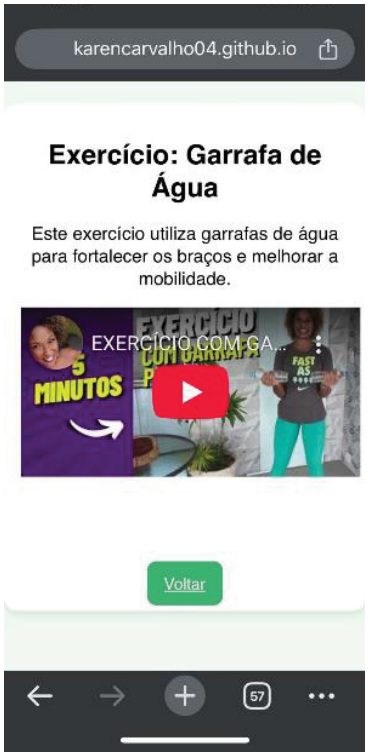
8.1 ARTEFATOS DO PROJETO

Figura 13 - Tela inicial do App FlexSênior



Fonte: A autora, 2025

Figura 14 - Tela de exercícios



Fonte: A autora, 2025

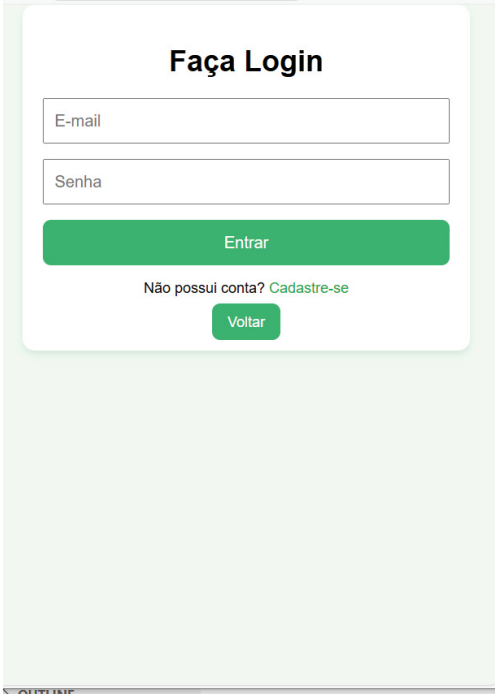
Figura 15 - Tela de feedback

Fonte: A autora, 2025

Figura 16 - Tela de chat

Fonte: A autora, 2025

Figura 17 - Tela de login



Fonte: A autora, 2025

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

A disciplina de Infraestrutura para Desenvolvimento e Implantação de *Software* (INFRA), também conhecida como DevOps, teve como principal objetivo capacitar o estudante na aplicação prática de automação, versionamento e entrega contínua de software. O projeto desenvolvido envolveu a configuração de um ambiente completo de integração contínua, com uso de *Docker Desktop* para a execução de containers com *GitLab* e *Jenkins*, demonstrando conhecimento em infraestrutura como serviço e monitoramento local de instâncias.

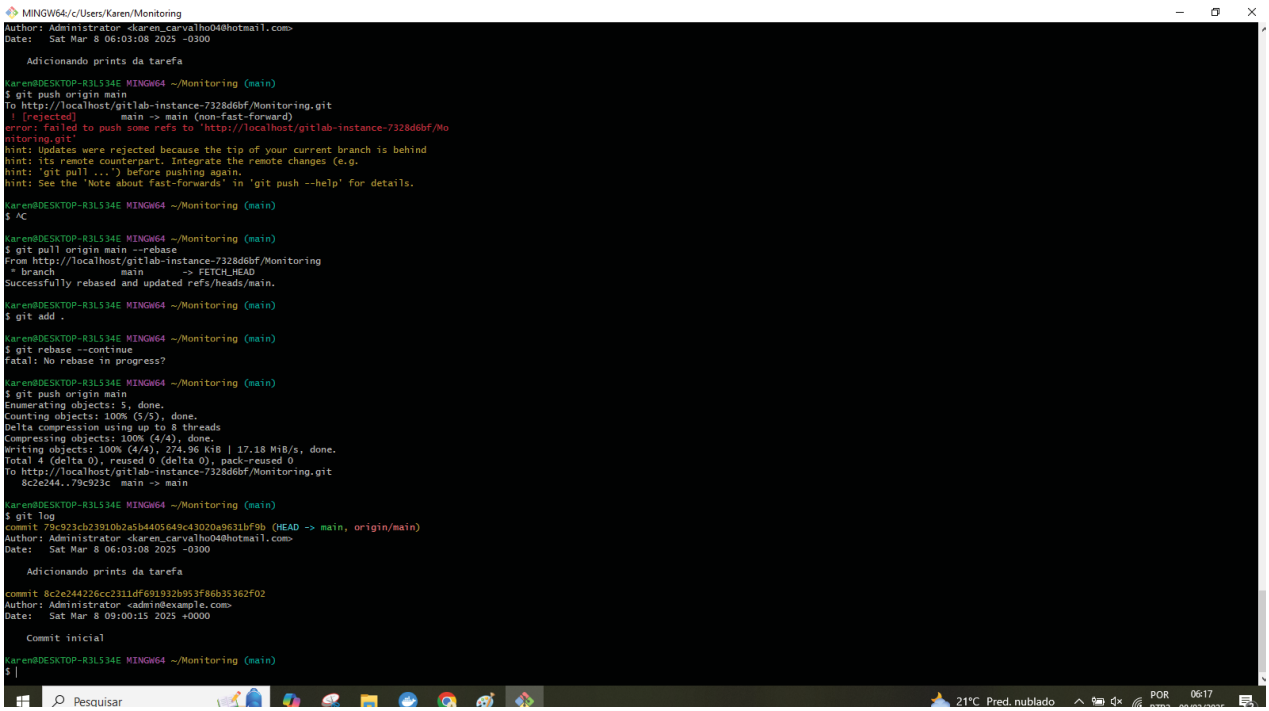
As atividades incluíram a utilização do terminal para executar comandos *Docker*, como *docker run* e *docker ps*, para subir o container da imagem dfwandarti/gitlab_jenkins:3. A integração entre os serviços permitiu observar na prática os fundamentos de ambientes desacoplados, configuração de portas e controle de imagens, além de evidenciar como esses recursos podem ser aplicados no fluxo de desenvolvimento ágil. No GitLab, foi visualizado o projeto de monitoramento da instância, garantindo que o serviço estivesse funcionando corretamente em ambiente local.

Complementando esse cenário, foi utilizado o Git para operações de versionamento como *pull, rebase, add, commit* e *push*, integrando os repositórios locais com o servidor *GitLab* instalado. Esse fluxo refletiu na prática os fundamentos de CI/CD (Integração e Entrega Contínua), reforçando o papel das ferramentas de controle de versão e automação na entrega incremental de valor.

A INFRA se conecta diretamente com disciplinas como AAP (ao aplicar versionamento e estruturação de código reutilizável), TEST (ao preparar o ambiente para a execução de testes automatizados) e WEB/UX (ao garantir a estabilidade e rastreabilidade das versões entregues). Essa disciplina consolidou o entendimento de que o ciclo ágil de *software* vai além da codificação: envolve a automação de processos, o controle dos ambientes e a confiabilidade nas entregas, promovendo a fluidez operacional e a integração entre desenvolvimento e operação.

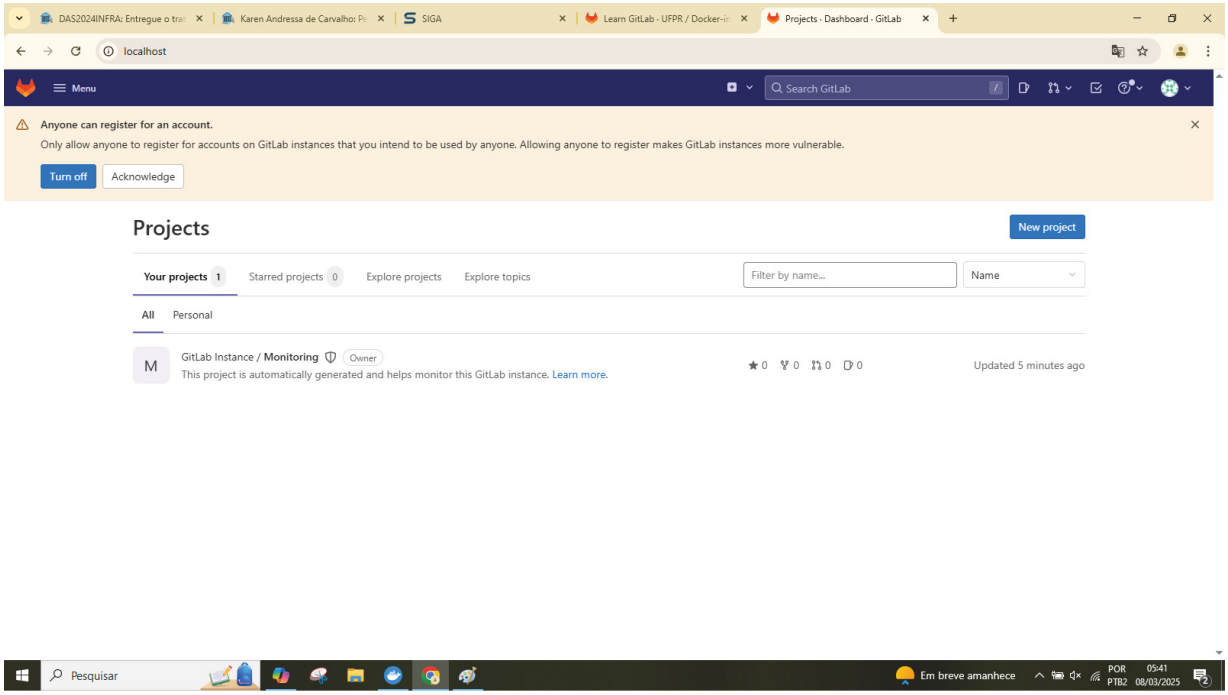
8.2 ARTEFATOS DO PROJETO

Figura 18 - Terminal Git com comandos de rebase e push



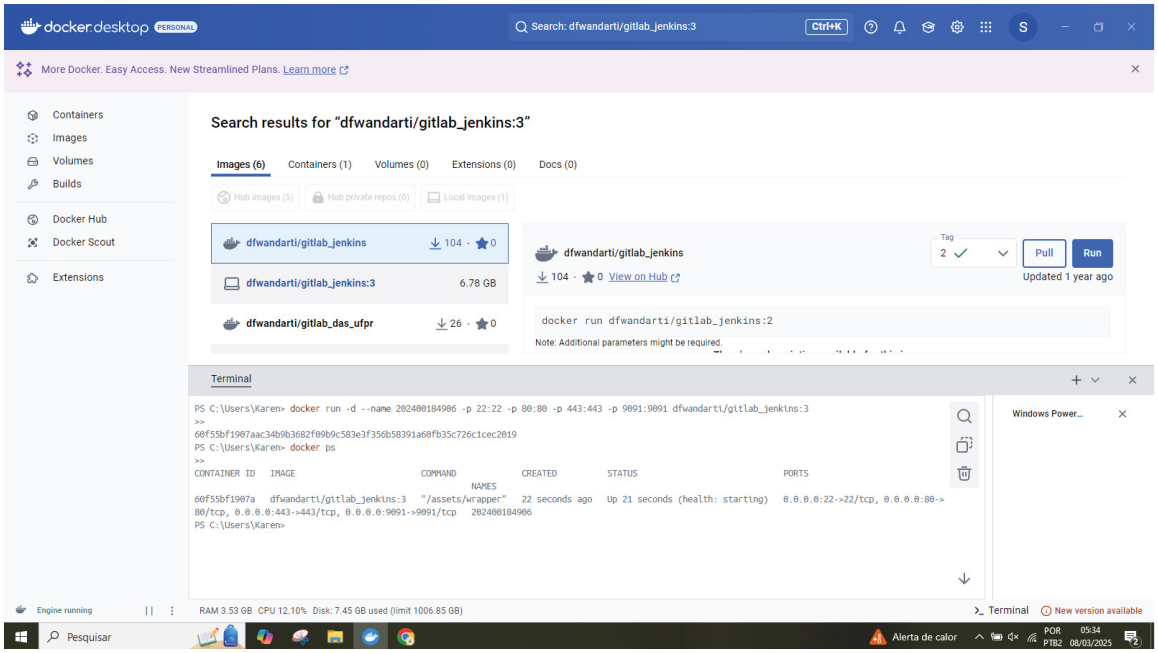
Fonte: A autora, 2025

Figura 19 - Dashboard do GitLab em localhost



Fonte: A autora, 2025

Figura 20 - Docker Desktop com container GitLab + Jenkins rodando



Fonte: A autora, 2025

9 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

A disciplina de Testes Automatizados teve como objetivo principal consolidar práticas de verificação e validação de *software* alinhadas aos princípios da qualidade contínua e do *feedback* rápido, elementos essenciais no contexto do desenvolvimento ágil. O projeto desenvolvido envolveu a criação de um *script* em *Playwright*, uma ferramenta moderna para testes *end-to-end* em aplicações web. A proposta incluiu o acesso a uma página real, preenchimento de campos e execução de ações automatizadas, simulando o comportamento de um usuário real.

A prática reforçou a importância de garantir que as funcionalidades implementadas funcionem como esperado, mesmo em ciclos curtos e frequentes de entrega. Ao automatizar interações com a interface e validar respostas da aplicação, o projeto mostrou como os testes podem prevenir regressões, agilizar o processo de homologação e dar mais segurança ao time de desenvolvimento. A disciplina também abordou princípios como teste de interface, simulação de fluxo de usuário e verificação de comportamentos assíncronos, aspectos cada vez mais relevantes em projetos modernos.

Os conhecimentos adquiridos foram aplicados de forma transversal nos demais projetos da especialização. Os testes criados complementaram a lógica desenvolvida em INTRO e AAP, verificaram fluxos definidos nas modelagens de MAG1 e MAG2, e foram utilizados em aplicações estruturadas em WEB e UX. No caso do aplicativo FlexSênior, por exemplo, as interações do usuário poderiam ser testadas automaticamente para garantir funcionalidade em diferentes dispositivos e navegadores, reforçando o compromisso com a acessibilidade e a qualidade da experiência do usuário.

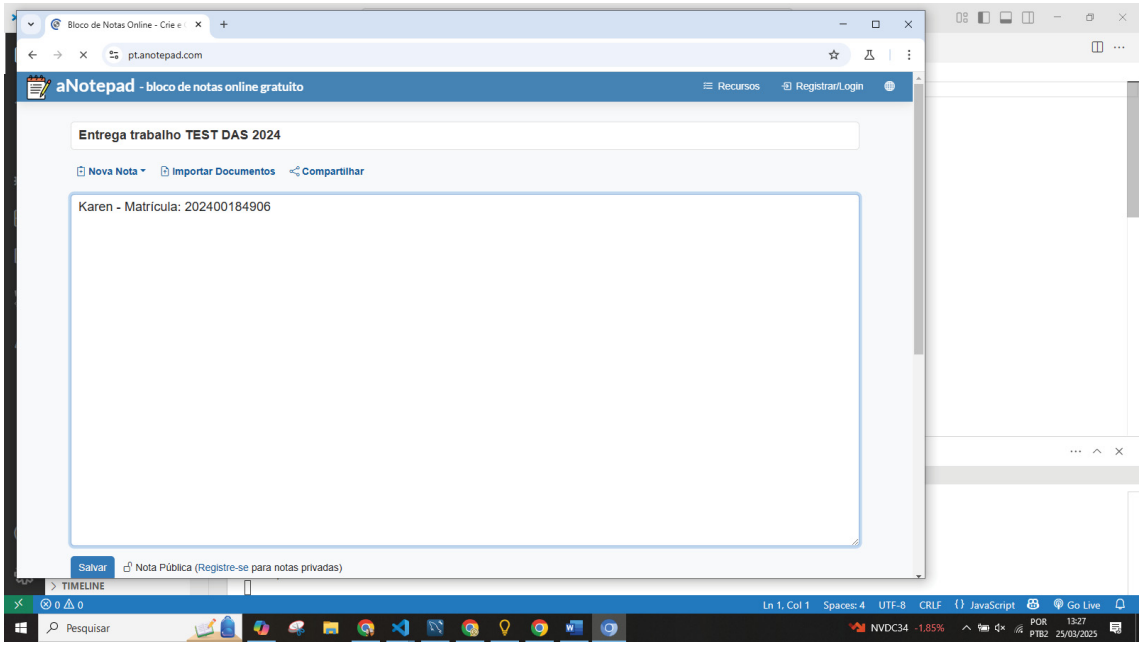
A disciplina TEST ampliou a visão da autora sobre a importância de integrar qualidade desde o início do processo de desenvolvimento, não como uma etapa final, mas como uma prática contínua. Isso está alinhado ao princípio ágil de “excelência técnica contínua” e ao objetivo de manter um ritmo sustentável de entrega com confiança.

9.1 ARTEFATOS DO PROJETO

```
     const { chromium } = require('playwright');

(async () => {
    try {
        const browser = await chromium.launch({ headless:
false });
        const page = await browser.newPage();

        console.log("Acessando o site...");
        await page.goto('https://pt.anotepad.com', {
waitUntil: 'domcontentloaded' });

        console.log("Aguardando campos de título e
conteúdo...");
        await page.waitForSelector('#edit_title', { timeout:
10000 });
        await page.waitForSelector('#edit_textarea', {
timeout: 10000 });

        console.log("Preenchendo campos...");
        await page.fill('#edit_title', 'Entrega trabalho TEST
DAS 2024');

        const conteudo = `Karen - Matrícula: 202400184906`;
        await page.fill('#edit_textarea', conteudo);

        console.log("Nota preenchida com sucesso!");

        await page.waitForTimeout(5000);

        await browser.close();
    } catch (error) {
        console.error("Ocorreu um erro durante a execução:",
error);
```

```
        }
})();
```

Figura 21 - Cópia de tela do aNotepad, contendo a execução do script



Fonte: A autora, 2025

10 CONCLUSÃO

O presente memorial apresentou os projetos desenvolvidos ao longo da Especialização em Desenvolvimento Ágil de *Software* da UFPR, evidenciando uma trajetória de aprendizagem pautada na aplicação prática de metodologias ágeis em diferentes etapas do ciclo de desenvolvimento. Cada disciplina contribuiu de forma integrada para a formação de uma visão sistêmica, colaborativa e técnica sobre como construir soluções de *software* alinhadas aos valores do Manifesto Ágil.

A construção de artefatos como modelagens UML, histórias de usuário, planejamentos de *release*, protótipos de interface e testes automatizados demonstrou o compromisso com entregas iterativas, com foco no usuário final e na melhoria contínua. Projetos como o sistema de gestão de condomínio e o aplicativo FlexSênior foram fundamentais para consolidar o conhecimento técnico em linguagens, banco de dados, testes e design centrado no usuário, promovendo a articulação entre teoria e prática de forma concreta.

Um dos principais aprendizados obtidos ao longo do curso foi compreender que o desenvolvimento ágil vai além da adoção de ferramentas ou rituais: trata-se de uma mudança cultural, centrada na colaboração entre pessoas, na valorização do *feedback* constante e na construção de soluções de forma adaptativa e incremental. Nesse sentido, o curso da UFPR proporcionou uma vivência realista, coerente com as práticas de mercado, contribuindo de forma decisiva para a qualificação profissional da autora.

Entre os principais desafios enfrentados estão o equilíbrio entre tempo, escopo e qualidade das entregas, a definição clara de critérios de aceitação e a construção de soluções que atendam simultaneamente aos requisitos técnicos e às necessidades dos usuários. Tais desafios, no entanto, reforçaram a importância de ciclos curtos, testes constantes e comunicação eficaz, princípios que certamente continuarão guiando a atuação profissional da autora nos próximos projetos.

11 REFERÊNCIAS

BECK, Kent et al. **Manifesto Ágil para Desenvolvimento de Software**. 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/. Acesso em: 20 maio 2025.

FOWLER, Martin. **Refatoração: aperfeiçoando o design de códigos existentes**. 2. ed. São Paulo: Novatec, 2019.

GIT. **Pro Git Book**. Disponível em: https://git-scm.com/book/pt-br/v2. Acesso em: 20 maio 2025.

KNUTH, Donald E. **A Arte de Programar Computadores**. São Paulo: Pearson, 2013.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de Software: Aprendendo com Exemplos**. Rio de Janeiro: Elsevier, 2007.

LARMAN, Craig. **Utilizando UML e Padrões**. 3. ed. Porto Alegre: Bookman, 2007.

MICROSOFT. **Playwright Documentation**. Disponível em: https://playwright.dev/. Acesso em: 20 maio 2025.

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

SILVA, Rodrigo Branas. **Arquitetura Limpa: O guia do artesão para estrutura e design de software**. São Paulo: Casa do Código, 2020.

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo: Pearson, 2019.