

UNIVERSIDADE FEDERAL DO PARANÁ

JESSICA PIMENTEL

MEMORIAL DE PROJETOS: DA CONCEPÇÃO À ENTREGA CONTÍNUA:
CONSTRUÇÃO DE UM SOFTWARE COM METODOLOGIAS ÁGEIS

CURITIBA

2025

JESSICA PIMENTEL

MEMORIAL DE PROJETOS: DA CONCEPÇÃO À ENTREGA CONTÍNUA:
CONSTRUÇÃO DE UM SOFTWARE COM 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: Prof. Dr. 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 **JESSICA PIMENTEL**, intitulada: **MEMORIAL DE PROJETOS: DA CONCEPÇÃO À ENTREGA CONTÍNUA: CONSTRUÇÃO DE UM SOFTWARE COM METODOLOGIAS ÁGEIS**, 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, 23 de Maio de 2025.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

O memorial de projetos tem como objetivo documentar a integração das disciplinas relacionadas ao Desenvolvimento Ágil de Software, evidenciando seu impacto na criação de soluções eficientes, desde sua concepção conceitual. As disciplinas abordadas forneceram conhecimentos teóricos e práticos para estruturação de sistemas baseados em metodologias e práticas de gerenciamento ágil, com foco em abordagens iterativas que contribuem para a coerência dos requisitos e uma implementação robusta. Esses fundamentos garantem a produção de um código limpo e sustentável, além de uma gestão eficiente das informações de dados, possibilitando o desenvolvimento moderno e escalável. Destaca-se a aplicação de testes que asseguram a qualidade do produto e uma infraestrutura que favorece a entrega contínua. Ressalta-se a atenção à experiência do usuário, por meio da criação de interfaces intuitivas que promovem acessibilidade e usabilidade. A integração entre as disciplinas possibilitou a consolidação do conhecimento ágil, promovendo mais eficiência, colaboração e adaptação constante ao longo do processo de desenvolvimento.

Palavras-chave: Ágil, Sustentabilidade, Software, Acessibilidade

ABSTRACT

This project report aimed to document the integration of subjects related to Agile Software Development, highlighting their impact on the creation of efficient solutions from conceptual design to systemic completion. The subjects studied provided both theoretical and practical knowledge for structuring systems based on agile methodologies and management practices, with a focus on iterative approaches that contributed to coherent requirements and robust implementation. These principles enabled the development of clean and sustainable code, efficient data management, and modern, scalable applications. Software testing ensured product quality, while the infrastructure supported continuous delivery processes. Additionally, user experience was prioritized through the design of intuitive interfaces that promoted accessibility and usability. The integration of these subjects consolidated knowledge in agile development, fostering greater efficiency, collaboration, and adaptability throughout the software development lifecycle.

Keywords: Agile, Sustainability, Software, Accessibility

SUMÁRIO

1	PARECER TÉCNICO.....	6
2	DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE.....	9
2.1	ARTEFATOS DO PROJETO	9
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 ÁGIL DE PROJETOS DE SOFTWARE 1 E 2	16
4.1	ARTEFATOS DO PROJETO	16
5	DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO.....	19
5.1	ARTEFATOS DO PROJETO	19
6	DISCIPLINA: BD – BANCO DE DADOS.....	21
6.1	ARTEFATOS DO PROJETO	21
7	DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO	23
7.1	ARTEFATOS DO PROJETO	23
8	DISCIPLINA: WEB1 E WEB2 – DESENVOLVIMENTO WEB 1 E 2	25
8.1	ARTEFATOS DO PROJETO	25
9	DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE.....	27
9.1	ARTEFATOS DO PROJETO	27
10	DISCIPLINA: MOB1 E MOB2 – DESENVOLVIMENTO MOBILE 1 E 2.....	29
10.1	ARTEFATOS DO PROJETO	29
11	DISCIPLINA: INFRA - INFRAESTRUTURA PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SOFTWARE (DEVOPS)	31
11.1	ARTEFATOS DO PROJETO	31
12	DISCIPLINA: TEST – TESTES AUTOMATIZADOS	32
12.1	ARTEFATOS DO PROJETO	32
13	CONCLUSÃO	33
	REFERÊNCIAS.....	34

1 PARECER TÉCNICO

Neste parecer técnico, serão abordados os projetos desenvolvidos como conclusão das disciplinas da Especialização em Desenvolvimento Ágil de Software, desde suas concepções teóricas até suas práticas e abordagens aplicadas ao desenvolvimento ágil de software.

O aprendizado em Desenvolvimento Ágil teve início com a disciplina de Métodos Ágeis de Desenvolvimento de Software (MADS, Capítulo 2), que apresentou desde métodos tradicionais de processo até os princípios e práticas ágeis contemporâneos. Segundo Pressman (2016), as abordagens ágeis representam uma evolução significativa na engenharia de software, priorizando a comunicação contínua e a entrega incremental de valor ao cliente. Foram explorados fundamentos da engenharia de software, o Manifesto Ágil, seus princípios, além de abordagens como *Lean Software Development*. Conforme Poppendieck (2003), o desenvolvimento de software deve eliminar tudo que não agrega valor ao cliente, promovendo entregas contínuas, com qualidade e sem desperdícios. Também foram estudados métodos como as cerimônias do Scrum, Kanban e *Extreme Programming* (XP), todos voltados à construção de um processo iterativo, ágil e focado na entrega contínua.

Após a consolidação da visualização prática do ciclo de desenvolvimento ágil, foi apresentada a disciplina de Modelagem Ágil de Software (MAG 1 e MAG 2, Capítulo 3), que teve foco na documentação dos requisitos por meio da UML (*Unified Modeling Language*). A disciplina abordou a elaboração de diagramas de caso de uso e histórias de usuário, ferramentas fundamentais para representar graficamente as funcionalidades e interações entre o usuário e o sistema. Conforme Booch, Rumbaugh e Jacobson (2005), os diagramas de caso de uso permitem capturar o comportamento do sistema sob o ponto de vista do usuário, facilitando o entendimento dos requisitos e o alinhamento entre equipe técnica e cliente. A modelagem contribuiu para tornar o desenvolvimento mais alinhado aos objetivos do produto.

O próximo passo na concepção de um software se consolidou na disciplina de Gerenciamento Ágil de Projetos de Software (GAP1 e GAP2, Capítulo 4), aprofundando o entendimento entre a gestão tradicional, com base no PMBOK 6ª edição, e as abordagens ágeis. Segundo o PMI (2017), o gerenciamento de projetos consiste na aplicação de conhecimentos e técnicas para atender aos requisitos do projeto. Foram exploradas ferramentas como *Design Thinking*, *Design Sprint*, *Lean*

Inception e práticas Scrum, incluindo estimativas com *Planning Poker* e planejamento de releases, promovendo uma gestão iterativa, colaborativa e orientada à entrega de valor.

A etapa prática da entrega contínua foi consolidada por meio das disciplinas de Introdução à Programação (Capítulo 5), Banco de Dados (Capítulo 6) e Aspectos Ágeis (Capítulo 7), que de forma complementar forneceram a base técnica e metodológica para o desenvolvimento de software de qualidade. A aplicação de TDD (*Test-Driven Development*) foi essencial para estruturar o código desde os testes, conforme Beck (2003, p.4), que defende que “o código é mais confiável quando os testes o antecedem”. A modelagem de dados e o uso de SQL permitiram integrar de forma eficiente a persistência da informação ao sistema. Esses conhecimentos foram potencializados por práticas como *Clean Code* e *Clean Architecture*. Como afirma Martin (2009, p.17), “o código limpo é aquele que foi escrito com clareza, que diz exatamente o que faz”, reforçando a importância de escrever software legível, testável e sustentável.

As disciplinas de Web (Web 1 e Web2, Capítulo 8), UX (Capítulo 9) e Mobile (MOB1 e MOB2, Capítulo 10) deram continuidade prática à aplicação dos conceitos de *Clean Code*, TDD e arquitetura limpa, promovendo a integração entre back-end, front-end e entrega multiplataforma. A etapa Web (Web 1 e Web2, Capítulo 8) envolveu o desenvolvimento de interfaces com Angular, *Typescript* e HTML, a implementação de operações CRUD, consumo de APIs Rest e persistência em banco de dados, consolidando a comunicação entre as camadas de aplicação. A Disciplina de UX (Capítulo 9) trouxe uma abordagem centrada no usuário, com construção de personas, jornadas, protótipos e princípios de design acessível e responsivo. Como afirma Norman (2004, p.188), “o design centrado no ser humano melhora a usabilidade e aumenta a satisfação do usuário”. Já a etapa Mobile (MOB1 e MOB2, Capítulo 10) permitiu adaptar o projeto para dispositivos móveis, com foco em layouts responsivos, persistência de dados, uso de corrotinas e consumo de serviços, garantindo fluidez e consistência em diferentes plataformas.

As disciplinas de Infraestrutura (Capítulo 11) e Testes Automatizados (Capítulo 12), encerraram o ciclo de desenvolvimento com foco na entrega contínua, confiável e escalável, consolidando a proposta metodológica ágil em sua totalidade. A disciplina de Infraestrutura (Capítulo 8) abordou desde o ciclo de vida do software (SDLC) até práticas modernas de DevOPS com uso de Docker e Kubernetes, construção de

pipelines de integração e entrega contínua (CI/CD), com estratégias de observabilidade que criam um ambiente propício para a automação e estabilidade dos ambientes. Enquanto a disciplina de Testes Automatizados (Capítulo 9) reforçou a importância da qualidade contínua por meio de testes unitários, execução de pipelines e testes de interface o que garantiu confiabilidade nas entregas. Com isso foi possível percorrer todas as etapas do desenvolvimento desde a concepção à prototipação e do código à infraestrutura, aplicando princípios ágeis de forma integrada.

O resultado de todas as etapas acima citadas se consolidou na construção de um software guiado por boas práticas, testes, integração contínua e foco na entrega de valor, alinhando diretamente ao propósito central: da concepção à entrega contínua com metodologias ágeis.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

A disciplina de Métodos Ágeis de Desenvolvimento de Software (MADS) foi essencial para estabelecer as bases teóricas e conceituais que orientaram todo o percurso da especialização e suas disciplinas seguintes. Seu objetivo inicial e principal foi apresentar a evolução dos processos de software e engenharia de software, desde os modelos tradicionais até as abordagens ágeis, esclarecendo a compreensão crítica sobre a necessidade de adaptação às demandas do desenvolvimento de software.

O Projeto da disciplina consistiu na elaboração de um mapa mental, reunindo os principais conceitos estudados ao longo da unidade. O exercício de construção das informações e dos fundamentos da engenharia de software, o ciclo de vida tradicional dos projetos, do conceito de maturidade dos processos e, principalmente, os valores e princípios do Manifesto Ágil permitiu a visualização de forma clara e linear para a abordagem iterativa e incremental. O Lean Software Development também foi abordado como apoio metodológico para eliminação de desperdícios e foco na entrega de valor.

A importância dessa disciplina serve como base fundamental para todas as demais, pois os conhecimentos adquiridos nela são continuamente aplicados nos projetos seguintes, passando pelo gerenciamento colaborativo de tarefas até a construção de soluções que adotam práticas como integração contínua. Assim, a disciplina de MADS não apenas introduziu os conceitos fundamentais do desenvolvimento ágil, mas também plantou a semente da mentalidade adaptativa e do pensamento ágil, essenciais para a continuidade e integração dos projetos desenvolvidos ao longo das disciplinas.

2.1 ARTEFATOS DO PROJETO

O objetivo foi construir um panorama conceitual seguindo um roadmap dos processos de software, iniciando pelo entendimento de sua importância dentro da engenharia de software. O processo de software é compreendido como um conjunto estruturado de atividades essenciais ao desenvolvimento e à manutenção de sistemas. A partir desse ponto, o mapa se ramifica em diversos tópicos fundamentais.

Entre os principais tópicos abordados, destacam-se:

- **Modelos Tradicionais de Processos de Software**, como os modelos Cascata, Incremental, Espiral e Prototipação, que representam abordagens sequenciais ou iterativas utilizadas no desenvolvimento tradicional de sistemas;
- **Manifesto Ágil**, que estabelece os valores e princípios fundamentais das metodologias ágeis, priorizando a interação entre indivíduos, o funcionamento do software, a colaboração com o cliente e a capacidade de adaptação às mudanças;
- **Princípios Ágeis**, que orientam práticas como entregas frequentes, motivação das equipes, comunicação eficaz, simplicidade e melhoria contínua;
- **Scrum**, com suas cerimônias (como *Sprint*, *Planning* e *Daily*), papéis bem definidos (*Product Owner*, *Scrum Master* e Time de Desenvolvimento) e foco na entrega incremental;
- **Extreme Programming (XP)**, que enfatiza práticas como programação em par, testes automatizados, integração contínua e simplicidade no design do software;
- **Lean Software Development**, baseado nos princípios do lean manufacturing, com foco na eliminação de desperdícios, na amplificação do aprendizado e na entrega rápida de valor;
- **Entrega Contínua de Software**, que envolve práticas voltadas à liberação frequente e segura de software, incluindo automação de testes, integração contínua e implantação automatizada.

Este trabalho reforça a importância de adaptar as metodologias ao contexto de cada equipe ou organização, promovendo maior eficiência, flexibilidade e qualidade no processo de desenvolvimento individual de cada time.

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 teve como objetivo aplicar a modelagem funcional de sistemas utilizando técnicas e ferramentas de princípio ágil. O projeto consistiu na modelagem de um Sistema de Gestão de Condomínio, iniciando com a criação de diagrama de caso de uso e da elaboração de histórias de usuário. Esses artefatos representaram as funcionalidades e interações entre os usuários e o sistema, permitindo a construção de uma visão sobre os requisitos priorizados para a conclusão do sistema estipulado.

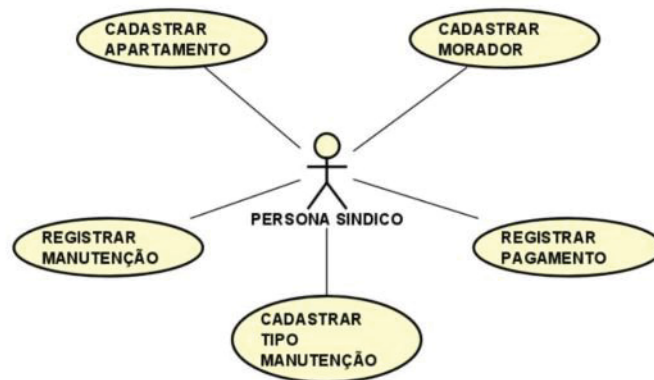
Mais do que uma etapa técnica, a disciplina de Modelagem Ágil conectou a compreensão das necessidades do usuário a sua estruturação lógica do sistema, mostrando a ponte entre a concepção do software com requisitos bem definidos e sua implementação prática. Servindo como base para promover coesão e planejamento funcional para a execução do projeto proporcionando uma visão clara e alinhada com os princípios ágeis, garantindo o valor das entregas ao longo do desenvolvimento.

O trabalho teve como objetivo a modelagem de um Sistema de Gestão de Condomínio, com foco na automação de serviços administrativos e operacionais de um condomínio residencial. A proposta foi desenvolvida em duas etapas, correspondentes às disciplinas de modelagem de software, com a entrega de artefatos fundamentais que descrevem a visão funcional e estrutural do sistema.

3.1 ARTEFATOS DO PROJETO

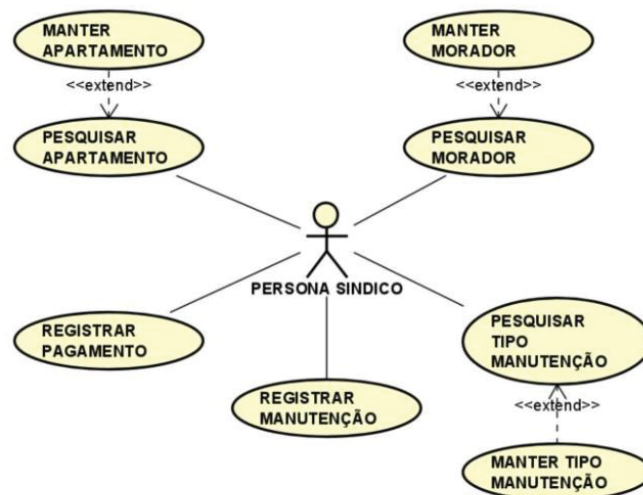
A criação dos diagramas foi essencial para estruturar visualmente os principais fluxos do sistema, detalhando os objetivos do usuário e os limites da aplicação, enquanto as histórias de usuário trouxeram detalhes sobre as funcionalidades esperadas facilitando assim o planejamento iterativo das entregas. Esse alinhamento no contexto garantiu a entrega de valor priorizando as expectativas esperadas pelo cliente já nas funcionalidades necessárias para o sistema.

Figura 2 – Diagrama de Casos de Uso nível 1.



Fonte: O autor (2025).

Figura 3 – Diagrama de Casos de Uso nível 2.



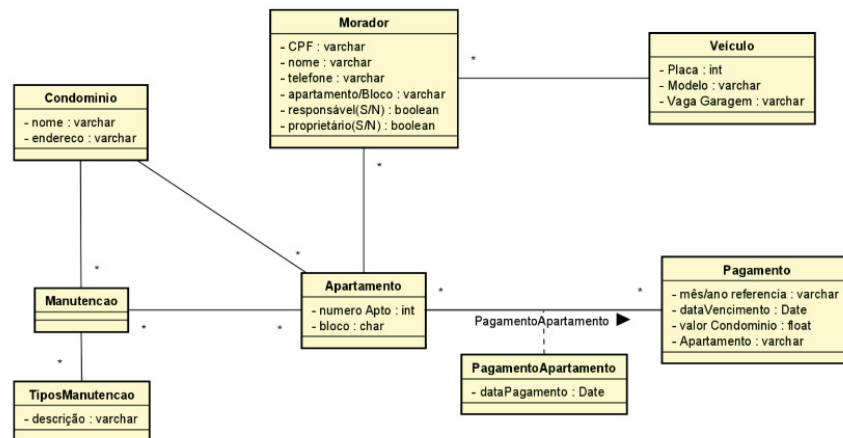
Fonte: O autor (2025).

O sistema proposto buscou atender às necessidades iniciais apresentadas pelo síndico, incluindo as funcionalidades de cadastro de apartamentos, cadastro de moradores, registro de pagamento do condomínio, registro de manutenções e as histórias de usuário foram elaboradas com base nas necessidades práticas do condomínio, buscando representar claramente os requisitos e comportamentos esperados no sistema. Os diagramas de caso de uso permitiram visualizar as funcionalidades em diferentes níveis de abstração e de concepção conceitual.

Na segunda parte do trabalho, voltada à Modelagem Ágil de Software II, foram produzidos artefatos com foco estrutural e comportamental, ampliando a

especificação do sistema com os diagramas de classe contendo as principais entidades do sistema, seus atributos a nível conceitual de tabela e seus relacionamentos.

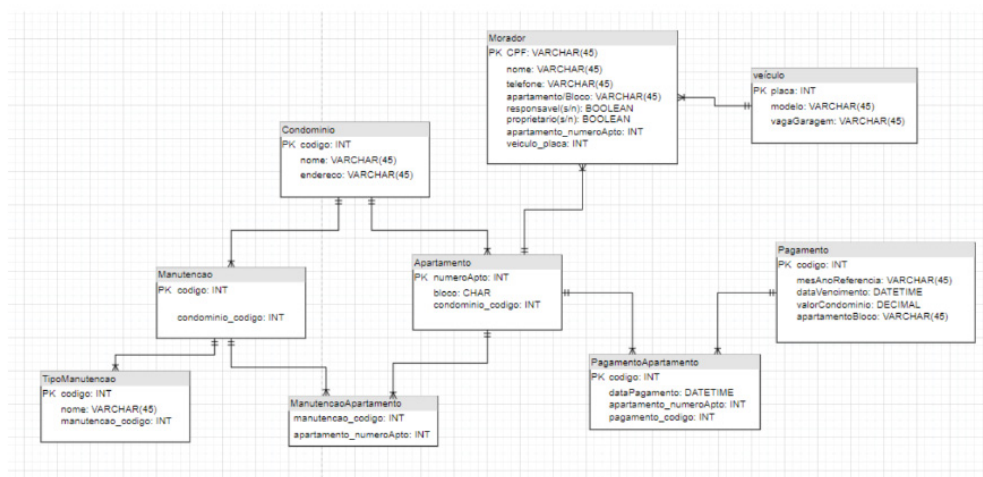
Figura 4 – Diagrama de Classes Com Relacionamentos Entre Entidades.



Fonte: O autor (2025).

A elaboração dos diagramas a nível de banco de dados corresponde ao detalhamento de estrutura lógica de armazenamento para a modelagem de tabelas.

Figura 5 – Tabelas de Banco de Dados Com Estrutura Lógica.

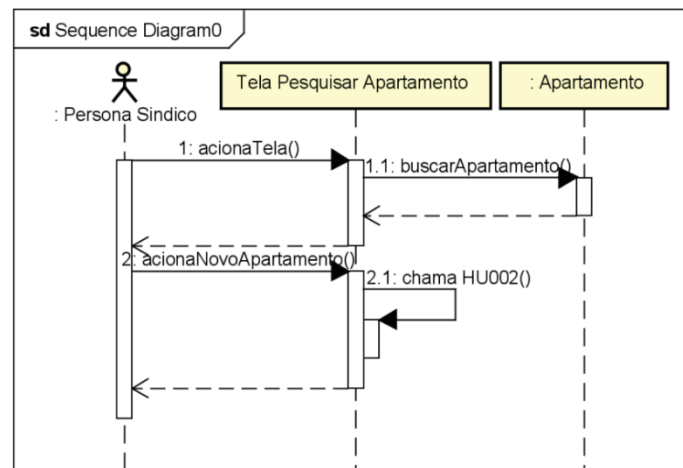


Fonte: O autor (2025).

Finalizando o fluxo de diagramas de modelagem de dados para a resolução do

sistema de condomínio a representação do fluxo de mensagens entre os objetos e a lógica de execução dos processos é feita através do diagrama de sequência, diagrama que representa a sequência de passos realizadas pelo usuário, a nível de tela, para a execução de ações e objetivos do sistema.

Figura 6 - Diagrama de Sequência Com Passos Realizados Pelo Usuário.



Fonte: O autor (2025).

Esses artefatos forneceram uma visão mais detalhada da arquitetura interna do sistema e do comportamento dinâmico de suas funcionalidades, permitindo compreender de forma clara a interação entre as classes e os usuários em tempo de execução. No contexto das metodologias ágeis, essa modelagem detalhada é essencial para garantir a comunicação eficiente entre os membros da equipe, promover entendimento compartilhado sobre o funcionamento do sistema e facilitar ajustes rápidos e iterativos durante o desenvolvimento.

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

As disciplinas de Gerenciamento Ágil de Projetos de Software (GAP1 e GAP2) foram fundamentais para consolidar a visão estratégica da aplicação dos métodos ágeis, unindo teoria e prática na organização, no acompanhamento e na entrega de projetos. Elas proporcionaram uma compreensão aprofundada sobre os modelos tradicionais de gerenciamento com base no PMBOK 6ª e 7ª edições que proporcionam uma visão completa sobre como adaptar o gerenciamento à realidade do desenvolvimento ágil.

O trabalho teve como objetivo simular o planejamento de um projeto ágil, aplicando os conceitos com a elaboração de um plano de release completo para o desenvolvimento de um software baseado em Scrum e uma simulação prática com foco em fluxo contínuo para aplicação de conhecimentos de gestão visual, métricas ágeis e acompanhamento de fluxo de trabalho.

4.1 ARTEFATOS DO PROJETO

A criação do quadro de release como planejamento visual serviu como organização do projeto em sprints, permitindo mapear e simular o que, quando e em qual ordem será feito. Proporcionando uma visão clara do progresso e priorização das atividades para gerenciamento da entrega contínua.

Para o cálculo de pontos e estimativas baseada na atuação dos desenvolvedores e qual a métrica da velocidade média para realização de cada atividade, as histórias de usuário são descritas favorecendo o entendimento do valor que cada funcionalidade que deverá ser entregue e qual a necessidade de realização dela para o cliente final. Com isso é feito o gerenciamento da sprint e qual limite de quantidade de histórias cabem no WIP (*Work in Progress*) do time, visando a entrega e não causando sobrecarga no time de desenvolvimento.

A partir do quadro de release e no decorrer do desenvolvimento, é possível ter uma gestão de velocidade, valor entregue e visão completa do desenvolvimento do projeto em sprints.

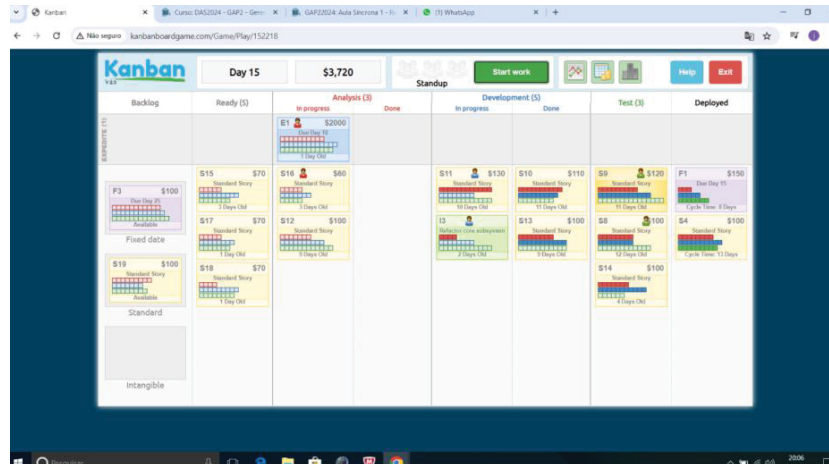
Figura 7 – Plano de Release Com Cálculos de Velocidade

Gerenciamento Ágil de Projetos I Profa. Dra. Rafaela Mantovani Fontana Template para o Plano de Release			
Aluno: Jéssica Pimentel Matéria: Gerenciamento Ágil de Projeto I			
Cálculo da Velocidade:			
Horas disponíveis por dia: 4 horas		Tamanho da Sprint: 2 semana	
Horas disponíveis por Sprint: 40 horas		Velocidade: 5	
Plano de Release:			
Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint 4
Data Início: 29/04/2024	Data Início: 13/05/2024	Data Início: 27/05/2024	Data Início: 10/06/2024
Data Fim: 10/05/2024	Data Fim: 24/05/2024	Data Fim: 07/06/2024	Data Fim: 21/06/2024
<H001 – Pesquisar Apartamento> SENDO um síndico QUERO pesquisar os apartamentos PARA fazer visualizar seus dados ESTIMATIVA (1)	<H004 – Manter Morador> SENDO um síndico QUERO manter os dados do morador PARA ser integrado ao banco de dados ESTIMATIVA (2)	<H007 – Registrar Manutenção> SENDO um síndico QUERO registrar manutenção PARA ser integrado ao banco de registro ESTIMATIVA (2)	<H009 – Pesquisar Vencimento> SENDO um síndico QUERO pesquisar vencimento PARA ser analisado registro pagamento ESTIMATIVA (2)
<H002 – Manter Apartamento> SENDO um síndico QUERO manter os dados do apartamento PARA ele estar regularizado ESTIMATIVA (3)	<H005 – Pesquisar Tipo Manutenção> SENDO um síndico QUERO pesquisar tipo manutenção PARA visualizar os dados ESTIMATIVA (1)	<H008 – Registrar Pagamento> SENDO um síndico QUERO registrar pagamento PARA ser integrado ao banco de registro ESTIMATIVA (3)	
<H003 – Pesquisar Morador> SENDO um síndico QUERO pesquisar os moradores PARA fazer visualizar os seus dados ESTIMATIVA (1)	<H006 – Manter Tipo Manutenção> SENDO um síndico QUERO manter tipo manutenção PARA ser integrado ao banco de dados ESTIMATIVA (2)		

Fonte: O autor (2025).

O foco prático da disciplina foi através da simulação prática sobre fluxo contínuo, utilizando um ambiente visual que representou o andamento real do projeto ágil que durante uma *sprint* são feitos movimentos das histórias de usuário e as mudanças nos estágios (em andamento, testando, concluído), sendo possível visualizar a evolução do trabalho ao longo do desenvolvimento.

Figura 8 – Fluxo de Desenvolvimento da Sprint Com Movimentações.



Fonte: O autor (2025).

Através da ferramenta “*KanbanBoardGame.com*” foi possível visualizar todo o decorrer e capacidade do time em lidar com as entregas, evidenciando a eficiência do fluxo de trabalho e gerando um gráfico CFD (*Cumulative Flow Diagram*) que registra visualmente o acúmulo e movimentações das tarefas, oferecendo uma leitura sobre a estabilidade do time e possíveis melhorias nos gargalos e dificuldades ao decorrer do desenvolvimento.

Figura 9 - Gráfico CFD (*Cumulative Flow Diagram*).



Fonte: O autor (2025).

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

A disciplina de Introdução à Programação teve como objetivo fornecer os conhecimentos fundamentais de lógica computacional, estrutura de códigos e boas práticas na programação orientada a objetos. Foram abordados os conceitos de tipos de dados, operadores, estruturas de controle, tratamento de erros e integração com banco de dados.

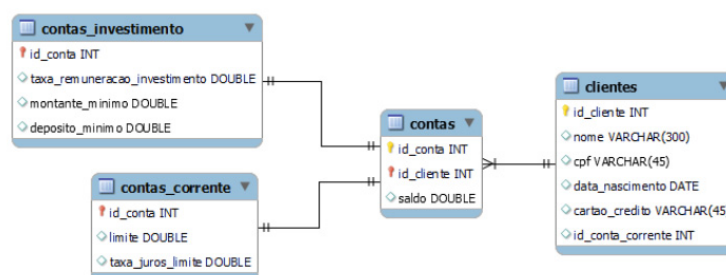
O trabalho realizado consistiu no desenvolvimento do *backend* de um sistema bancário, com foco em funcionalidades de cadastro e movimentações, promovendo a construção de um sistema real, orientado a testes, funcional e ágil.

5.1 ARTEFATOS DO PROJETO

Para a construção do sistema bancário com movimentações para contas correntes e contas de investimento, os requisitos definidos que deveriam ser atendidos precisariam passar em uma série de 42 testes unitários realizados com *JUnits*. Para atingir o objetivo de passar os testes foi necessário a abordagem de implementação de sistema orientada a testes, onde o objetivo é que os testes que inicialmente estão falhando sejam completados com o desenvolvimento do que é necessário para seu sucesso.

Durante o desenvolvimento os conceitos de herança e polimorfismo foram aplicados diretamente nas classes do sistema, buscando a realização de um sistema completo para a realização das movimentações bancárias, promovendo um código reutilizável e alinhado aos princípios de orientação a objetos.

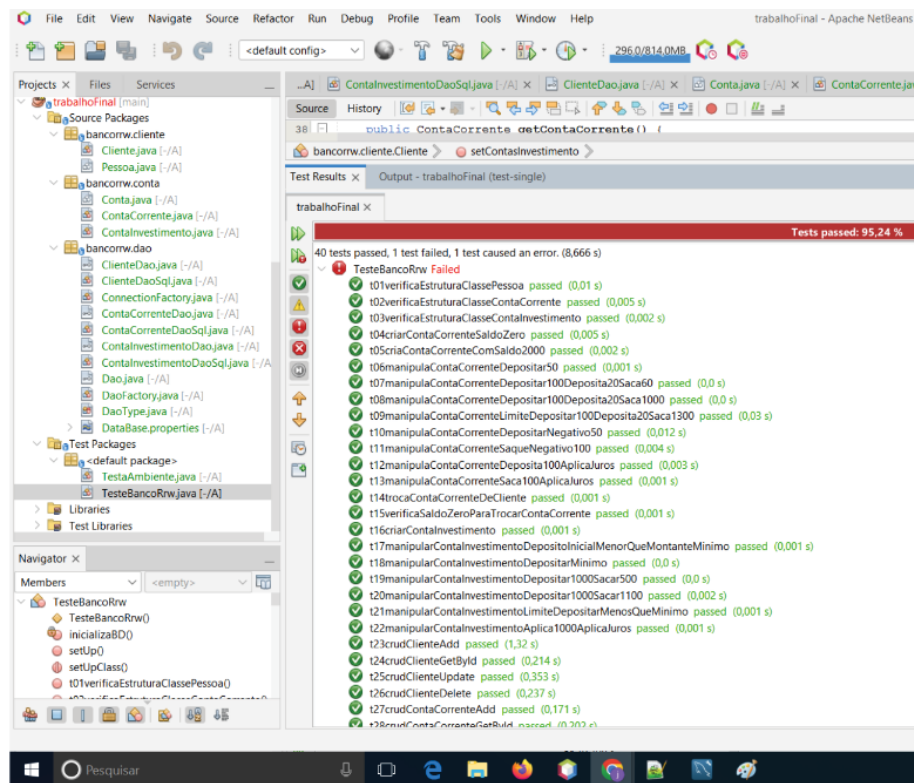
Figura 10 – Diagrama de Relacionamento para Banco de Dados.



Fonte: O autor (2025)

Para a integração com o banco de dados foram utilizados script DDL garantindo que os registros dos clientes, contas e operações fossem devidamente armazenadas e recuperadas conforme os testes eram rodados e evidenciando o fluxo de teste do início ao fim de cada cenário testado.

Figura 11 – Testes Unitários com JUnits e estrutura de projeto bancário.



Fonte: O autor (2025).

6 DISCIPLINA: BD – BANCO DE DADOS

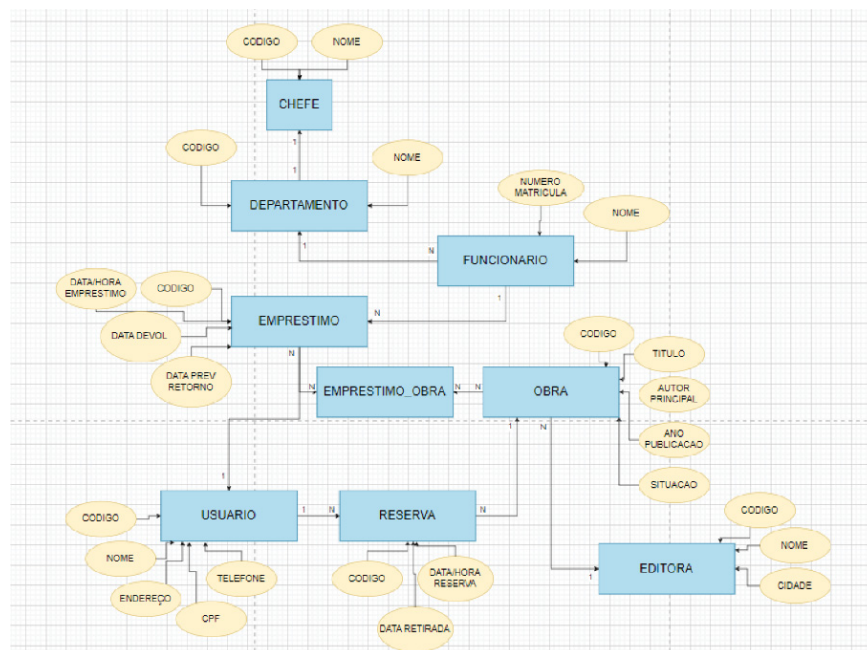
A disciplina de Banco de Dados teve como foco introduzir os fundamentos da modelagem de dados, normalização, linguagem SQL e estruturação de sistemas de persistência de dados. A abordagem prática envolveu a construção conceitual de modelos até a implementação completa em SQL com foco em integridade referencial, tipos de relacionamento e inserção de dados em banco.

O trabalho de Banco de Dados foi uma voltado à modelagem conceitual e lógica de um sistema de biblioteca e na implementação prática com SQL com aplicação dos tipos de relacionamentos (1:1, 1:N e N:N).

6.1 ARTEFATOS DO PROJETO

Para a construção de um banco de dados com pré-requisitos de um sistema de biblioteca foi necessário o desenvolvimento de um modelo de Entidade-Relacionamento para visualização que representasse as principais entidades e relacionamentos existentes para o controle dos livros, departamentos, empréstimos e reservas.

Figura 12 – Modelo Entidade-Relacionamento Conceitual.

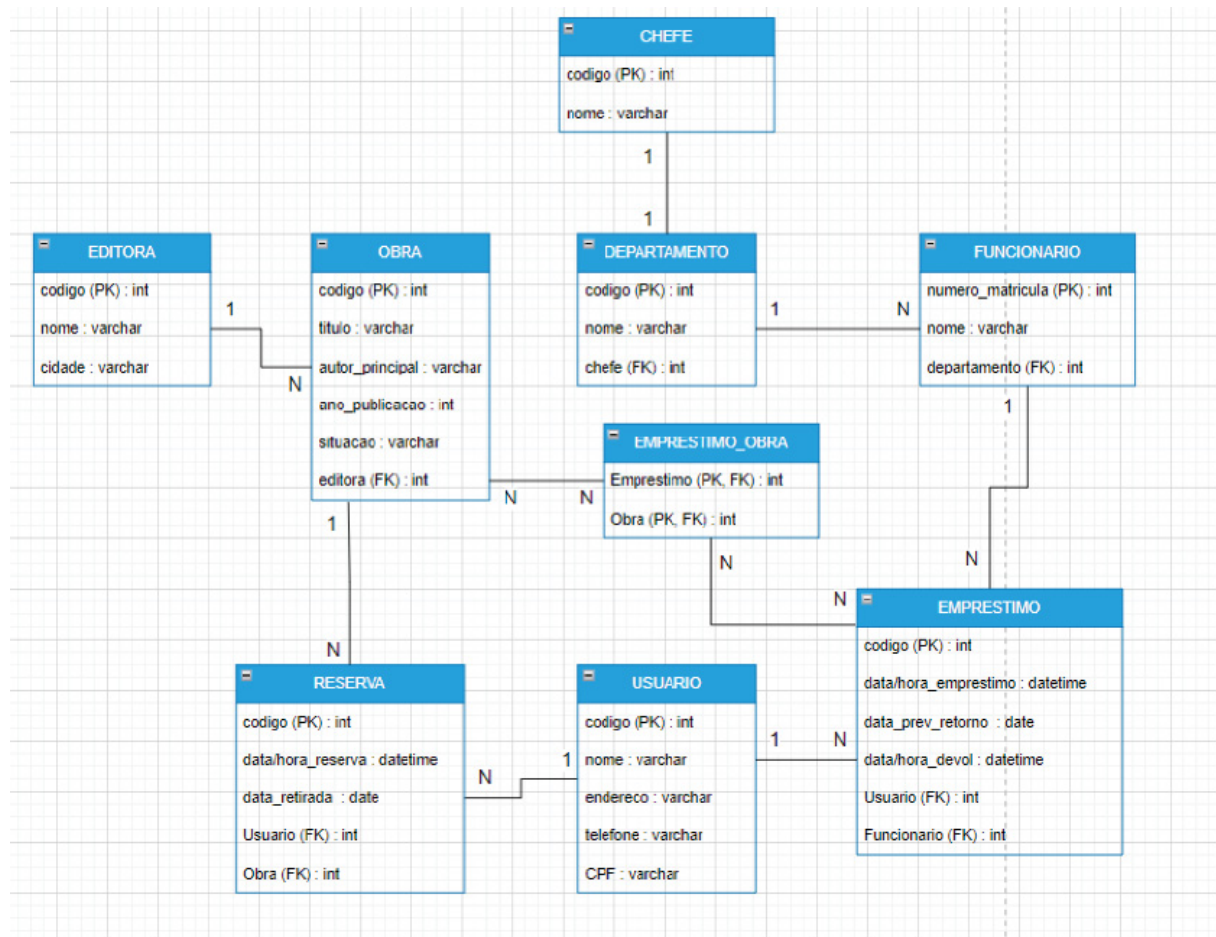


Fonte: O autor (2025).

Quanto ao entendimento a nível de tabelas necessários para o

desenvolvimento do sistema, são elaborados Modelos Lógicos (Relacional) com os tipos de dados, restrições e relações entre os dados utilizados no sistema para um esquema visual representando as tabelas e seus relacionamentos.

Figura 13 – Modelo Lógico com Diagrama de Entidade Relacionamento



Fonte: O autor (2025).

Com base no modelo lógico, são elaborados *scripts* em SQL que inserem registros nas tabelas do banco de dados, preenchendo-as com informações essenciais para testes, consultas e funcionamento do sistema respeitando os relacionamentos e estruturas dos dados criados nos diagramas conceituais.

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

A disciplina de Aspectos Ágeis aprofundou conceitos técnicos e boas práticas de desenvolvimento de software com foco em qualidade e manutenção de código. Foram abordadas práticas como *Clean Code*, Refatoração e *Test-Driven Development* (TDD), *Behavior-Driven Development* (BDD) e *Clean Architecture* que formam a base para um desenvolvimento sustentável e colaborativo no desenvolvimento ágil, com estrutura de código preparada para mudanças contínuas.

O trabalho da disciplina consistiu na aplicação de *Clean Code* em um código já existente de ordenação de tipo chamado *Bubble Sort*, visando tornar o código mais legível, coeso e de fácil manutenção.

7.1 ARTEFATOS DO PROJETO

O código original da implementação do algoritmo *Bubble Sort* apresenta diversos pontos que vão contra os princípios de *Clean Code*. Um dos principais problemas está na escolha de nomes pouco descritivos para variáveis e métodos não deixando claro o seu papel dentro da lógica do código, o que compromete a legibilidade e dificulta o entendimento na manutenção desse código.

Outro ponto encontrado são as repetições de código, o que fere o princípio de responsabilidade única dos princípios da Arquitetura Limpa, sendo o ideal separar essas ações em métodos distintos, organizando melhor o fluxo do sistema.

Como pontos de melhoria no código, foram aplicadas diversas alterações alinhadas aos princípios do *Clean Code*. A primeira delas foi a renomeação das variáveis para nomes mais descritivos e em inglês, tornando o código autossuficiente em termos de leitura, refletindo de forma clara as funções de cada linha escrita.

Figura 14 - Renomeação e Correções de Projeto com *Clean Code*.

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

static void sortArrayUsingBubbleSort (int[] array) {
    boolean isSorted;
    for (int pass = 0; pass < array.length - 1; pass++) {
        isSorted = true;
        for (int current = 0; current < array.length - pass - 1; current++) {
            if (array[current] > array[current + 1]) {
                swapElements(array, current, current + 1);
                isSorted = false;
            }
        }
        if (isSorted) break;
    }
}
```

Fonte: O autor (2025).

Os métodos do código também foram renomeados para deixar mais claro o que cada um faz, facilitando a leitura. Além disso, foi criado métodos novos deixando o código principal mais simples e organizado. Os comentários do código anterior também foram retirados já que após as mudanças o código era autossuficiente e poderia ser entendido sem a necessidade de comentários, melhorando a legibilidade para futuras manutenções e expansões de forma sustentável.

Figura 15 – Melhorando a Legibilidade do Código

```
static void swapElements(int[] array, int firstIndex, int secondIndex) {
    int temporary = array[firstIndex];
    array[firstIndex] = array[secondIndex];
    array[secondIndex] = temporary;
}

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

Fonte: O autor (2025).

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

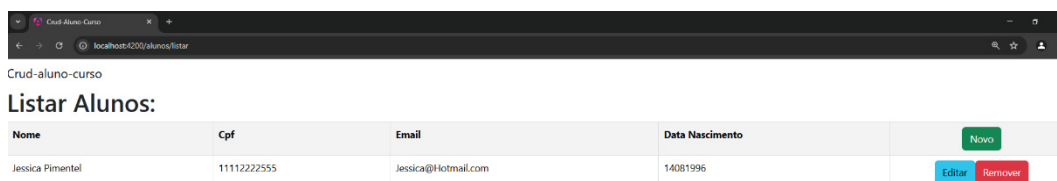
As disciplinas de Desenvolvimento Web 1 e Web 2 permitiram aplicar na prática os fundamentos do desenvolvimento *front-end* e *back-end*. Em Web 1, o foco foi na construção de interfaces com HTML, *TypeScript*, Angular e o uso de *Material Design* para o *layout*. Já em Web 2, o aprendizado foi ampliado com a integração do *back-end* em Java com Spring Boot, persistência em PostgreSQL, e consumo de APIs REST, consolidando a comunicação entre as duas camadas da aplicação.

8.1 ARTEFATOS DO PROJETO

O trabalho da disciplina de Web 1 teve como objetivo a criação de dois *CRUDs*, um para cadastro de alunos e outro para cursos. Todas as informações foram armazenadas localmente, utilizando o recurso de *Local Storage* do navegador. A aplicação contava com telas simples e funcionais para listar, adicionar, editar e excluir registros, respeitando os princípios de criação de interface com o uso de *Bootstrap* ou *Material Design*. As rotas foram organizadas de forma que permitia a navegação direta entre os módulos, o que garantiu uma experiência fluida durante o uso do usuário.

Na disciplina de Web 2, o mesmo sistema foi aprimorado e passou a contar com um terceiro *CRUD*, voltado ao gerenciamento de matrículas. Nessa etapa, o projeto evoluiu para uma estrutura mais robusta, abandonando o armazenamento local para ter seus registros salvos em um banco de dados PostgreSQL, com as tabelas criadas por meio de *scripts* com SQL. O *back-end* foi desenvolvido com *Spring Boot* e *Spring Data JPA*, garantindo integração com o *front-end* feito em Angular.

Figura 16 – Crud de Alunos Feito em Angular.

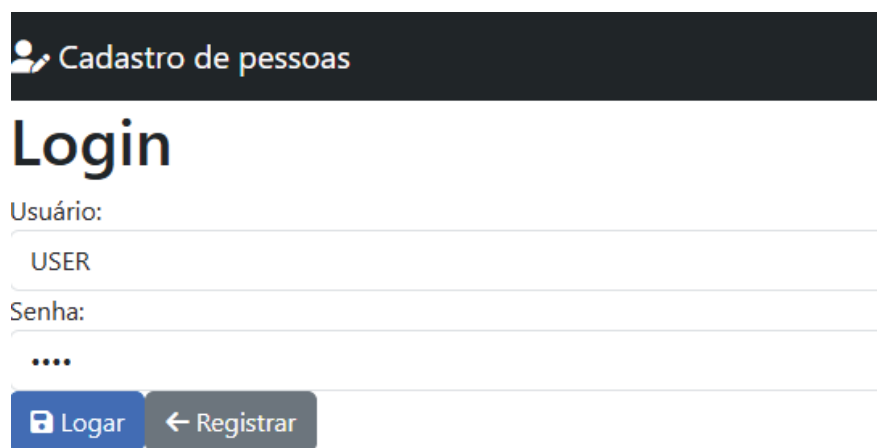


Nome	Cpf	Email	Data Nascimento	
Jessica Pimentel	11112222555	Jessica@hotmail.com	14/08/1996	Novo Edit Remove

Fonte: O autor (2025).

A tela de matrícula foi desenvolvida para permitir relacionamento entre alunos e cursos por meio de campos de seleção (*combobox*), registrando também a data da matrícula e a nota final do aluno. Foram mantidas funcionalidades completas de inserção, edição, visualização por meio de modal e exclusão com confirmação, tornando o sistema mais completo e conectado com as práticas modernas de desenvolvimento web, seguindo práticas de *Clean Code* e *Clean Architecture*.

Figura 17 - Tela de Login Feito em Angular.



The image shows a web application interface for user login. At the top, there is a dark header bar with a white user icon and the text 'Cadastro de pessoas'. Below the header, the word 'Login' is displayed in a large, bold, black font. Underneath, there are two input fields. The first is labeled 'Usuário:' and contains the text 'USER'. The second is labeled 'Senha:' and contains four dots, indicating a password field. At the bottom of the form, there are two buttons: a blue button with a white login icon and the text 'Logar', and a grey button with a white left arrow icon and the text 'Registrar'.

Fonte: O autor (2025).

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

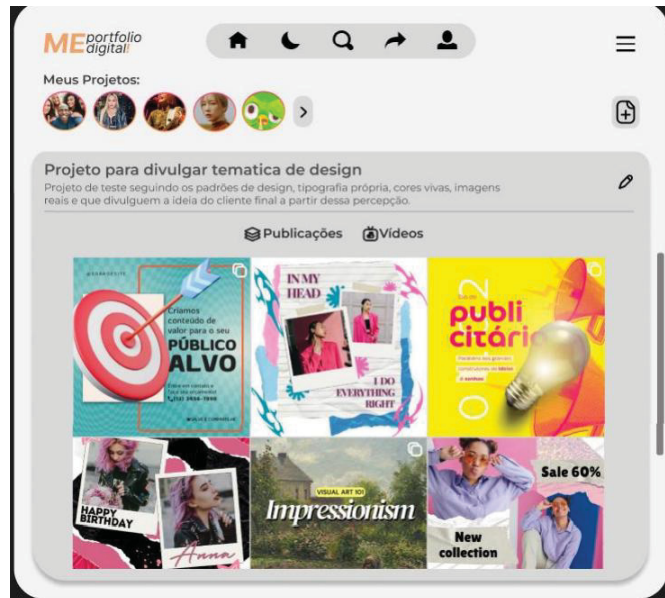
A disciplina de UX abordou os principais conceitos e práticas voltados à criação de interfaces centradas nas necessidades, comportamentos e emoções do usuário voltado ao *UX Design*, compreensão do usuário e arquitetura da informação, com o uso de ferramentas e princípios como personas, jornadas do usuário, hierarquia de informação e *wireframes*. Também foram exploradas técnicas modernas de *design* de interface (UI) como design responsivo, animações, tipografia e fundamentos de *design* visual.

O trabalho da disciplina consistiu no desenvolvimento de um protótipo funcional de telas, com explicação das decisões de *design* adotadas (como cores, fontes e *layout*), e um teste com pelo menos um usuário real, para validação das interações e coleta de *feedback*.

9.1 ARTEFATOS DO PROJETO

O protótipo funcional teve como foco principal o desenvolvimento de um site de portfólio para designers e profissionais da área de social média. A proposta do sistema é permitir que esses profissionais apresentem suas artes (como *posts*, carrosséis e vídeos) em um formato que simula o *feed* de uma rede social. A ideia surgiu ao observar que, durante processos seletivos ou propostas de trabalho, pessoas que trabalham com social media precisam demonstrar como suas criações se encaixam na identidade visual do cliente. Com isso, o sistema possibilita uma visualização prática e realista, valorizando não só o *design* da arte em si, mas também o contexto em que ela será aplicada.

Figura 18 - Protótipo de Portifólio para Social media e Designers



Fonte: O autor (2025).

O protótipo foi desenvolvido com base em conceitos estudados na disciplina, como jornada do usuário, arquitetura da informação, *wireframes*, acessibilidade, responsividade e *design* emocional. Foram criadas seis telas no total, incluindo login, registro, dashboard com feed de projetos, criação e edição de projetos, e a visualização de um projeto pronto para receber as artes. As decisões de *design* priorizaram um visual neutro e limpo para destacar as criações do usuário, mantendo a estética inspirada no *Instagram*, com elementos familiares como os círculos no estilo stories e ícones intuitivos para facilitar a navegação.

O teste de usabilidade realizado trouxe um ponto de vista importante para esse tipo de validação com o usuário, parte essencial dentro do desenvolvimento ágil, pois permite identificar melhorias ainda nas fases iniciais do projeto, promovendo entregas incrementais com base em *feedbacks* reais. Além disso, reforça a importância da colaboração contínua com o usuário final e da flexibilidade para adaptações rápidas.

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

A disciplina de Desenvolvimento Mobile introduziu os fundamentos da criação de aplicativos Android utilizando o Android Studio. O foco esteve na construção de *interfaces* simples, funcionais e organizadas, promovendo boas práticas na estruturação da lógica e organização das *views*, além da aplicação de corrotinas e integração com serviços externos.

Os trabalhos das disciplinas resultaram na criação de dois aplicativos, um voltado à organização financeira pessoal (*FinApp*) e outro para consulta de dados da franquia *Harry Potter* utilizando uma API pública chamada HP-API.

10.1 ARTEFATOS DO PROJETO

O primeiro aplicativo desenvolvido, chamado *FinApp*, teve como objetivo auxiliar o usuário na organização de sua vida financeira, simulando uma demanda real de uma *FinTech*. A aplicação foi estruturada em três principais telas: *Dashboard*, Cadastro de operações e Extrato.

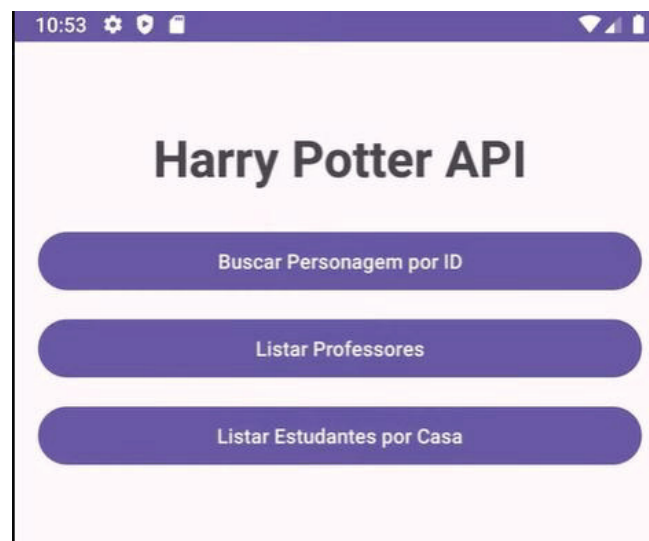
- A tela principal (*Dashboard*) apresenta botões organizados de forma equilibrada para acesso rápido às funcionalidades: cadastrar transações, visualizar extrato e sair do aplicativo.
- Na *activity* de cadastro, o usuário informa se a operação é um débito ou crédito, insere uma descrição e o valor. As transações são armazenadas em estruturas de memória temporária, não havendo persistência após o fechamento do aplicativo.
- A tela de extrato exibe as transações cadastradas em uma lista (*ListView*), com visual estilizado e organizado para destacar o tipo, descrição e valor de cada item.

Esse projeto consolidou os conhecimentos sobre *layouts* em XML, navegação entre *activities*, manipulação de dados em memória e uso de *ListView* padrão para exibição de informações. O segundo aplicativo teve como objetivo a prática de

corrotinas, consumo de Web Services e manipulação de respostas de APIs REST, utilizando a API pública.

A aplicação foi composta por quatro principais funcionalidades, acessadas a partir do *Dashboard*: Buscar personagem por ID: permite que o usuário informe um identificador e receba como resposta o nome e a casa do personagem, exibidos em um *TextView*. Listar professores: exibe os nomes dos professores da escola de *Hogwarts*, consumindo diretamente o *endpoint* da API. Listar estudantes por casa: o usuário escolhe uma casa (ex: *Grifinória*, *Sonserina*) através de *RadioButtons*, e a aplicação retorna uma lista com os estudantes correspondentes. Sair: opção para encerrar a aplicação. O projeto aplicou boas práticas de consumo de APIs REST com *Retrofit*, manipulação de dados com *Json*, e execução assíncrona com *Kotlin Coroutines*, além de reforçar a importância da organização das telas e tratamento adequado das respostas da *API*.

Figura 19 – Aplicativo HPApp em funcionamento.



Fonte: O autor (2025).

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

A disciplina de Infraestrutura abordou as principais práticas e ferramentas utilizadas no ciclo de vida do desenvolvimento moderno de *software*, com foco em automação, versionamento, provisionamento e monitoramento de ambientes. Entre os temas explorados estiveram o ciclo de vida do desenvolvimento (SDLC), práticas de DevOps, controle de versão com Git, definição de métricas, utilização de contêineres com *Docker*, automação de *pipelines*, orquestração com Kubernetes, além de práticas de observabilidade e uso de IaC (*Infrastructure as Code*).

11.1 ARTEFATOS DO PROJETO

O trabalho da disciplina consistiu na execução de um contêiner Docker com uma imagem pública, utilizando o mapeamento de portas e nomeando esse contêiner com a matrícula do aluno.

Com o ambiente em execução, foi acessado um versionador de repositório via navegador, disponibilizado dentro do contêiner, para validar o funcionamento completo de uma imagem pública sendo replicada e utilizada em ambiente isolado. A atividade teve como objetivo aplicar na prática conceitos como execução de contêineres com Docker, acesso a serviços de auto hospedagem, interação com terminal Linux e integração de ferramentas em ambiente DevOps.

Figura 20 – Comandos para utilização de Docker no Terminal.

R: Comando sendo feito:

```
PS C:\Users\jppim> docker run -d -p 22:22 -p 80:80 -p 443:443 -p 9091:9091 --name 40001016398E1-jessica
-pimintel dfwandarti/gitlab_jenkins:3
Unable to find image 'dfwandarti/gitlab_jenkins:3' locally
3: Pulling from dfwandarti/gitlab_jenkins
b42364568e18: Download complete
c0436bfc19a7: Download complete
d46df4aa614d: Download complete
5d4e92472abe: Download complete
c695fd85ed38: Download complete
5cf33ab37f3b: Download complete
d7bfe07ed847: Download complete
061a5147aada: Download complete
3f36ee87421e: Download complete
25bb9590d9c9: Download complete
Digest: sha256:c8f217311114d5e795315534e827d13a09a2c63d8956eedac5cca475c6e3780
Status: Downloaded newer image for dfwandarti/gitlab_jenkins:3
6cb155d3a3857914d433b9772e1441a04979c3d0e83494a6b642860647b8f826
PS C:\Users\jppim>
```

R: Container rodando:

```
PS C:\Users\jppim> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        NAMES
6cb155d3a385   dfwandarti/gitlab_jenkins:3         "/assets/wrapper"       9 minutes ago   Up 9 minutes   healthy
0.0.0.0:22->22/tcp, 0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 0.0.0.0:9091->9091/tcp   40001016398E1
-jessica-pimintel
PS C:\Users\jppim>
```

Fonte: O autor (2025)

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

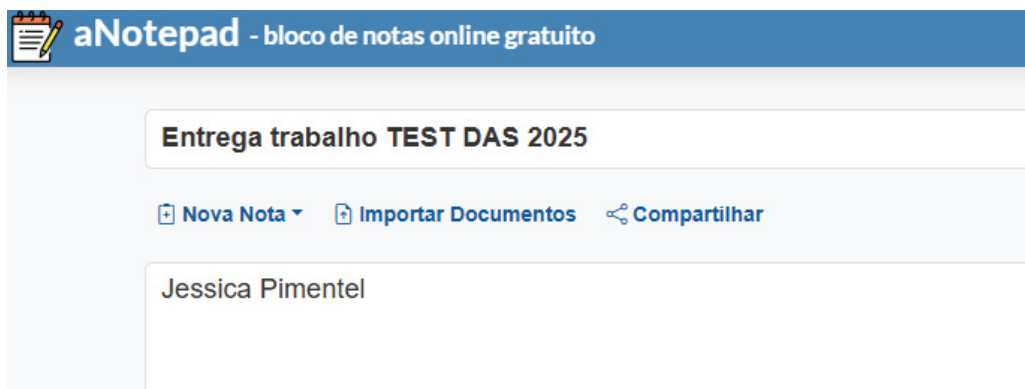
A disciplina de Testes Automatizados apresentou os principais conceitos e práticas relacionados à validação de *software* por meio de testes automáticos, fundamentais para garantir a qualidade e estabilidade de sistemas em ciclos ágeis de desenvolvimento. Foram abordados tanto testes unitários com *JUnit*, quanto testes automatizados de interface com *Selenium* e *Playwright*, promovendo a automação desde os níveis mais baixos até a camada de interação com o usuário final.

12.1 ARTEFATOS DO PROJETO

O trabalho da disciplina consistiu na criação de um *script* automatizado com *Playwright* que acessa um site, cria uma nota e preenche automaticamente o título e o corpo com as informações passadas em código.

A atividade consolidou a aplicação de testes automatizados de interface, explorando comandos de navegação, identificação de elementos por seletores, preenchimento de campos e simulação de ações do usuário como clique e digitação.

Figura 21 - Automação de Preenchimento com PlayWright.



Fonte: O autor (2025).

Essa prática reforçou a importância dos testes automatizados como parte do ciclo ágil de desenvolvimento, garantindo verificações rápidas e confiáveis a cada entrega de código, e permitindo ao desenvolvedor validar fluxos essenciais de forma contínua e escalável.

13 CONCLUSÃO

Este memorial percorreu de maneira sistêmica as disciplinas de Especialização em Desenvolvimento Ágil de Software, reunindo fundamentos teóricos e práticas aplicadas que ofereceram uma visão geral e abrangente de todo o ciclo de vida de um projeto ágil. Iniciamos com os conceitos centrais dos métodos ágeis (Manifesto Ágil, Princípios *Lean* e Modelagem Ágil) que estabeleceram a base para a colaboração eficaz entre equipes e clientes. Em seguida, aprofundamos no Gerenciamento Ágil de projetos, que usou de base as diretrizes do PMBOK com frameworks como *Scrum*, *Kanban*, *Lean Inception* e métricas com planejamento de releases para orientar entregas iterativas e previsíveis.

A incorporação de práticas como *Test-Driven Development*, *Clean Code* e *Clean Architecture* elevou a qualidade do software garantindo a legibilidade, manutenibilidade e sustentabilidade do código. Nos ambientes web e mobile se comprovou a integração fluida entre *front-end* e *back-end*, aliada aos protótipos centrados na experiência do usuário que resultou em soluções responsivas e acessíveis. E por fim, as disciplinas de infraestrutura e Testes Automatizados consolidaram um *pipeline* de CI/CD, uso de Docker, kubernetes e cobertura segura e abrangente de testes que asseguram implantações estáveis de aplicações.

A adoção plena do método ágil em organizações demonstra significativo avanço nas entregas e fluxos de trabalho, porém ainda encontra desafios para sua aceitação definitiva. Dentre esses desafios os principais são a necessidade de fomentar mentalidade ágil junto a liderança e equipes, buscando uma maturidade técnica e comportamental que capacita os colaboradores com uma mudança cultural. O equilíbrio entre documentação e agilidade, definindo junto ao time níveis de documentação e prática ágil para alcançar uma entrega contínua.

Superar essas barreiras culturais é essencial para os conceitos e técnicas ágeis se transformem em eficiência, qualidade e geração de valor para as empresas e time de desenvolvimento, trazendo mais qualidade de trabalho e de produto.

REFERÊNCIAS

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

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário**. Rio de Janeiro: Elsevier, 2005.

MARTIN, Robert C. **Código Limpo: Habilidades Práticas do Agile Software**. Rio de Janeiro: Alta Books, 2009.

NORMAN, Donald A. **O design do dia a dia**. Rio de Janeiro: Rocco, 2004.

PMI – PROJECT MANAGEMENT INSTITUTE. **Um guia do conhecimento em gerenciamento de projetos – Guia PMBOK. 6. ed.** São Paulo: Project Management Institute, 2017.

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

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