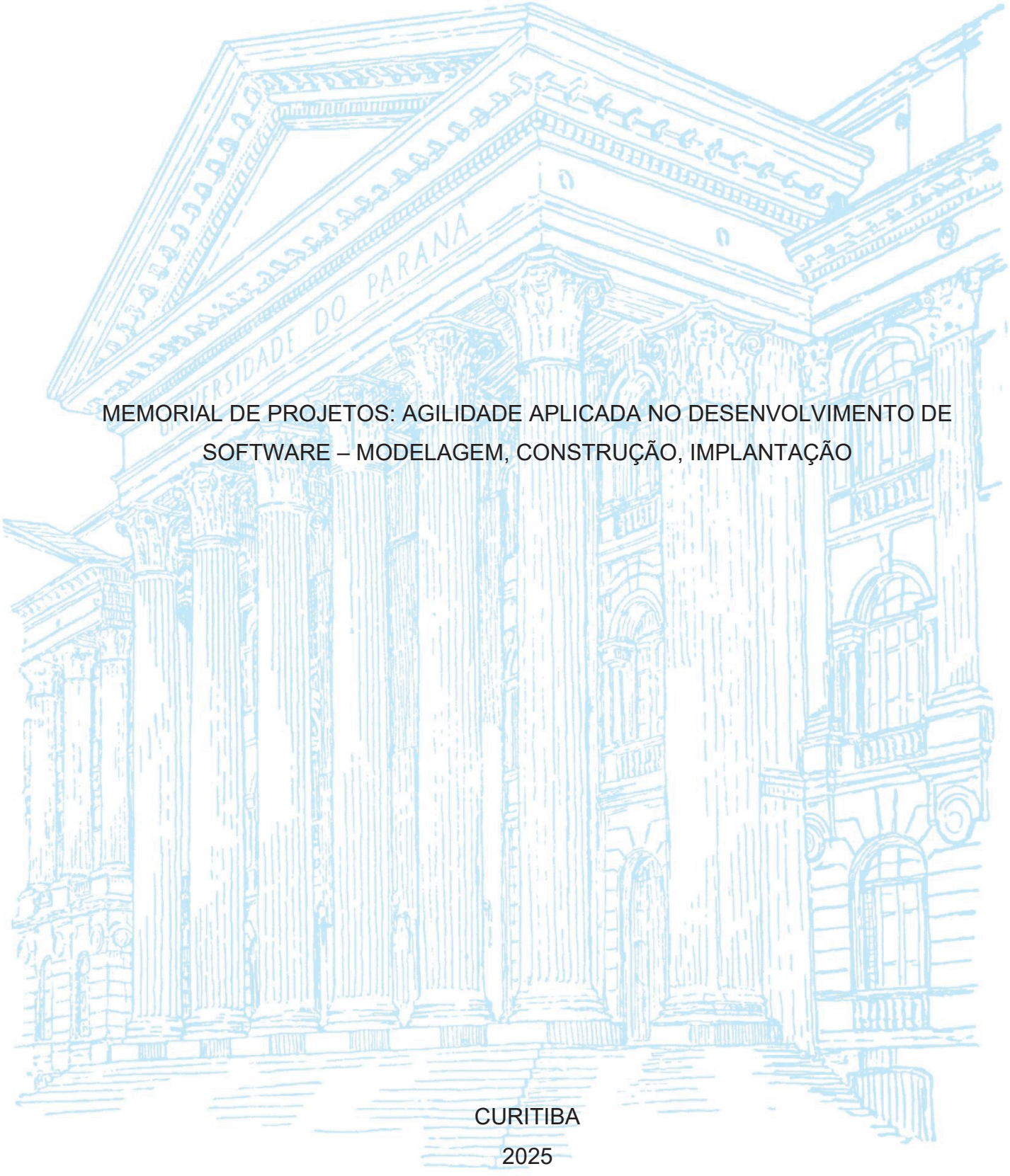
UNIVERSIDADE FEDERAL DO PARANÁ

ALINE RODRIGUES DE LIMA



MEMORIAL DE PROJETOS: AGILIDADE APLICADA NO DESENVOLVIMENTO DE SOFTWARE – MODELAGEM, CONSTRUÇÃO, IMPLANTAÇÃO

CURITIBA

2025

ALINE RODRIGUES DE LIMA

MEMORIAL DE PROJETOS: AGILIDADE APLICADA NO DESENVOLVIMENTO DE SOFTWARE – MODELAGEM, CONSTRUÇÃO, IMPLANTAÇÃO

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. Jaime Wojciechowski

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 **ALINE RODRIGUES DE LIMA**, intitulada: **MEMORIAL DE PROJETOS: AGILIDADE APLICADA NO DESENVOLVIMENTO DE SOFTWARE MODELAGEM, CONSTRUÇÃO, IMPLANTAÇÃO**, 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, 08 de Agosto de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Rua Doutor Alcides Vieira Arco-verde - Curitiba - Paraná - Brasil
CEP 81520-260 - Tel: (41) 98874-0472 - E-mail: jaimewo@ufpr.br

RESUMO

Este memorial apresenta a trajetória de aprendizado e aplicação prática da agilidade no desenvolvimento de software ao longo do curso, por meio de projetos que envolveram desde a modelagem até a implantação de sistemas. Dentre as atividades destacam-se: criação de mapa mental sobre métodos ágeis, produção de diagramas UML (case de uso e classe), design de interfaces, elaboração de plano de *release* com base no Scrum, e simulação de fluxo de trabalho com Kanban. Foram realizados projetos de modelagem de dados e implementação em SQL, desenvolvimento de back-end orientado a testes unitários, e refatoração de algoritmo segundo princípios de *Clean Code*. No front-end, foram construídas aplicações web com Angular e TypeScript, inicialmente usando Local Storage e depois integradas a back-end com Spring Boot e PostgreSQL. As interfaces foram elaboradas com Bootstrap e CSS, garantindo responsividade e consistência visual. A qualidade foi assegurada por testes automatizados com Playwright, simulando uso real da aplicação. Além disso, houve configuração de ambientes com Docker e GitLab para controle de versão e integração contínua, reforçando práticas ágeis e DevOps. Os projetos evidenciam a integração de conhecimentos em modelagem, codificação, testes e implantação, consolidando a compreensão prática do desenvolvimento ágil de software.

Palavras-chave: agilidade, modelagem, desenvolvimento, testes, implantação.

ABSTRACT

This report presents the learning journey and practical application of agility in software development throughout the course, through projects that covered everything from modeling to system deployment. The main activities included creating a mind map on agile methods, producing UML diagrams (use case and class), designing interfaces, preparing a Scrum-based *release* plan, and simulating a Kanban workflow. Projects involved data modeling and SQL implementation, test-driven backend development, and algorithm refactoring following *Clean Code* principles. On the frontend, web applications were developed using Angular and TypeScript, initially with Local Storage and later integrated with a backend built with Spring Boot and PostgreSQL. Interfaces were created with Bootstrap and CSS, ensuring responsiveness and visual consistency. Software quality was ensured through automated testing with Playwright, simulating real application usage. Additionally, environments were configured using Docker and GitLab for version control and continuous integration, reinforcing agile and DevOps practices. These projects demonstrate the integration of knowledge in modeling, coding, testing, and deployment, consolidating practical understanding of agile software development.

Keywords: agility, modeling, development, testing, deployment.

LISTA DE FIGURAS

SUMÁRIO

1 PARECER TÉCNICO

Este parecer demonstra como a agilidade foi aplicada no desenvolvimento de software ao longo do curso, integrando teoria e prática nas etapas de modelagem, construção e implantação.

A disciplina Métodos Ágeis para Desenvolvimento de Software (MADS), descrita no Tópico 2, consolidou a evolução dos processos de software, desde os modelos tradicionais até as abordagens ágeis, conforme Beck et al. (2001, não p.), ao afirmar que “os processos ágeis promovem desenvolvimento sustentável”. Conceitualmente, fundamentaram a flexibilidade em Banco de Dados e a automação na Infraestrutura para Desenvolvimento e Implantação de Software. Na prática, orientaram por meio de simulações com Scrum e Kanban em Gerenciamento Ágil de Projetos de Software, entregas incrementais em Desenvolvimento Web, refatoração e *Clean Code* em Aspectos Ágeis de Software, desenvolvimento orientado a testes unitários em Introdução à Programação, foco em qualidade e produtividade em Testes Automatizados e participação ativa do cliente na prototipação em UX no Desenvolvimento Ágil de Software.

As disciplinas Modelagem Ágil de Software 1 e 2 (MAG1 e MAG2), descritas no Tópico 3, enfatizaram a modelagem funcional e estrutural com foco em documentação leve, conforme Sommerville (2011, p. 16) de que “software não é apenas um programa ou programas; ele inclui também a documentação.” Conceitualmente, a modelagem sustentou regras de negócio e diagramas para guiar uma implementação clara em Aspectos Ágeis de Software, interfaces e lógica de negócios em Desenvolvimento Web, modelos de dados em Banco de Dados, pipelines em Infraestrutura para Desenvolvimento e Implantação de Software e estrutura de testes em Testes Automatizados. Na prática, orientou histórias de usuário e iterações em Métodos Ágeis para Desenvolvimento de Software, organização de requisitos em Gerenciamento Ágil de Software e prototipação com validações contínuas em UX no Desenvolvimento Ágil de Software.

O desenvolvimento em Gerenciamento Ágil de Projetos de Software 1 e 2 (GAP1 e GAP2), detalhado no Tópico 4, focou na aplicação prática de Scrum e Kanban para planejar, monitorar e adaptar o trabalho. Conceitualmente, articulou entrega incremental e priorização em Desenvolvimento Web, simplicidade em Aspectos Ágeis de Software, soluções testáveis em Introdução à Programação,

automação para qualidade em Testes Automatizados, integração contínua em Infraestrutura para Desenvolvimento e Implantação de Software. Na prática, alinhou histórias ao cliente em Métodos Ágeis para Desenvolvimento de Software, guiou ajustes de protótipos e melhorou a experiência do usuário em UX no Desenvolvimento Ágil de Software.

A disciplina Introdução à Programação (INTRO), descrita no Tópico 5, focou no desenvolvimento orientado a testes unitários. Como afirma Martin (2020, não p.): “O Desenvolvimento Orientado a Testes é a segurança que a equipe técnica usa com objetivo de avançar rapidamente, mantendo a mais alta qualidade.” Conceitualmente, relacionou-se à manutenção e refatoração em Métodos Ágeis para Desenvolvimento de Software, qualidade e evolução em Gerenciamento Ágil de Projetos de Software, *Clean Code* em Aspectos Ágeis de Programação, modelagem orientada a objetos em Modelagem Ágil de Software e validação com JUnit em Testes Automatizados. Na prática, envolveu manipulação de dados com scripts SQL e uso de diagramas de classes em Banco de Dados.

O projeto de Banco de Dados (BD), detalhado no Tópico 6, envolveu a modelagem e implementação de bancos relacionais com modelos conceituais, lógicos e scripts SQL. Conceitualmente, alinhou-se ao planejamento adaptativo em Métodos Ágeis para Desenvolvimento de Software, modelagem lógica e conceitual em Modelagem Ágil de Software, organização de requisitos em Gerenciamento Ágil de Projetos de Software, legibilidade de scripts em Aspectos Ágeis de Programação, e ambientes de integração em Infraestrutura para Desenvolvimento e Implantação de Software. Na prática, integrou persistência de dados em Desenvolvimento Web e manipulação dos dados via código em Introdução à Programação.

A disciplina Aspectos Ágeis de Programação (AAP), detalhada no Tópico 7, focou na refatoração de um código em Java baseada nos princípios do *Clean Code*. Como enfatiza Martin (2009, p.7), ao destacar que “um programador que escreve um código limpo é um artista que pode pegar uma tela em branco e submetê-la a uma série de transformações até que se torne um sistema graciosamente programado.” Conceitualmente, apoiou a simplicidade e entrega contínua em Métodos Ágeis para Desenvolvimento de Software, iterações e comunicação em Gerenciamento Ágil de Projetos de Software, diagramas objetivos para regras de negócios claras em Modelagem Ágil de Software. Na prática, influenciou o desenvolvimento estruturado em Introdução à Programação, reutilização de código e organização em

Desenvolvimento Web, consultas legíveis em Banco de Dados, integração contínua e em Infraestrutura para Desenvolvimento e Implantação.

As disciplinas Desenvolvimento Web 1 e 2 (WEB1 e WEB2), descritas no Tópico 8, abordaram a construção de uma aplicação web para gestão acadêmica, evoluindo do armazenamento local para integração com back-end e banco de dados. Conceitualmente, refletiram o desenvolvimento interativo e entregas incrementais em Métodos Ágeis para Desenvolvimento de Software, ciclos curtos em Gerenciamento Ágil de Projetos de Software, definição ágil de entidades em Modelagem Ágil de Software, qualidade em Testes Automatizados, automação e integração contínua em Infraestrutura para Desenvolvimento e Implantação. Na prática, integraram PostgreSQL em Banco de Dados, aplicação de *Clean Code* em Aspectos Ágeis de Software e usabilidade em UX no Desenvolvimento Ágil de Software.

A disciplina UX no Desenvolvimento Ágil de Software (UX), detalhada no Tópico 9, consistiu no desenvolvimento de protótipos funcionais para organização de jogos. Conceitualmente, baseou-se em protótipos e histórias de usuário em Modelagem Ágil de Software. Na prática, contribuiu com validações rápidas em Métodos Ágeis para Desenvolvimento de Software, participação ativa do usuário e entregas alinhadas em Gerenciamento Ágil de Projetos de Software, além da construção de interfaces acessíveis e responsivas em Desenvolvimento Web.

O projeto da disciplina Infraestrutura para Desenvolvimento e Implantação de Software – DevOps (INFRA), descrito no Tópico 10, focou na criação e configuração de ambientes com Docker e GitLab. Conceitualmente, apoiou a validação contínua em Modelagem Ágil de Software. Na prática, permitiu testes em ambiente real em Desenvolvimento Web, execução de testes em Testes Automatizados, automação do fluxo de trabalho em Gerenciamento Ágil de Projetos de Software e organização de código em Aspectos Ágeis de Programação.

Na disciplina Testes Automatizados (TEST), detalhada no Tópico 11, foi desenvolvido um script para automatizar a criação de notas no site Anotepad. Conceitualmente, reforçou o foco em qualidade em Métodos Ágeis para Desenvolvimento de Software, validação por fluxos em Modelagem Ágil de Software, controle de riscos em Gerenciamento Ágil de Projetos de Software, execução automatizada em Infraestrutura para Desenvolvimento e Implantação, organização modular do código em Introdução à Programação e interação com interface em Desenvolvimento Web.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

O projeto propõe a elaboração de um mapa mental abrangente sobre o desenvolvimento ágil de software, abordando conceitos essenciais de processo de software, suas atividades e modelos tradicionais — como cascata, incremental, prototipação e espiral —, além do Manifesto Ágil, princípios ágeis, Lean Software Development, Scrum, Extreme Programming, Kanban e entrega contínua de software.

Com bases nos conteúdos explorados ao longo da disciplina, foram desenvolvidos mapas conceituais que consolidam os principais tópicos e práticas relacionados aos métodos ágeis. Entre os temas abordados, destacam-se os modelos iterativos e incrementais, o detalhamento das etapas do processo de software, os princípios do Manifesto Ágil, práticas do Extreme Programming (XP) e estratégias voltadas à entrega contínua.

A construção desses mapas mentais visa proporcionar uma compreensão integrada e acessível dos métodos ágeis, reforçando sua aplicabilidade no contexto do desenvolvimento moderno.

A adoção dessas abordagens representa uma mudança significativa na forma como o software é concebido e mantido, promovendo maior colaboração, entregas frequentes de valor, foco nas necessidades do cliente e capacidade de adaptabilidade a contextos complexos.

> Práticas ágeis, usadas no processo de manutenção em si, provavelmente são mais eficazes, independente de ter sido usada uma abordagem ágil para o desenvolvimento do sistema. Entrega incremental, projeto para mudanças e manutenção da simplicidade — tudo isso faz sentido quando o software está sendo alterado. Na verdade, você pode pensar em um processo ágil de desenvolvimento como um processo de evolução do software. (SOMMERVILLE, 2011, p. 42).

Portanto, a aplicação dos métodos ágeis fortalece a sinergia entre equipes e clientes, possibilitando respostas rápidas às mudanças e assegurando a evolução contínua dos sistemas desenvolvidos.

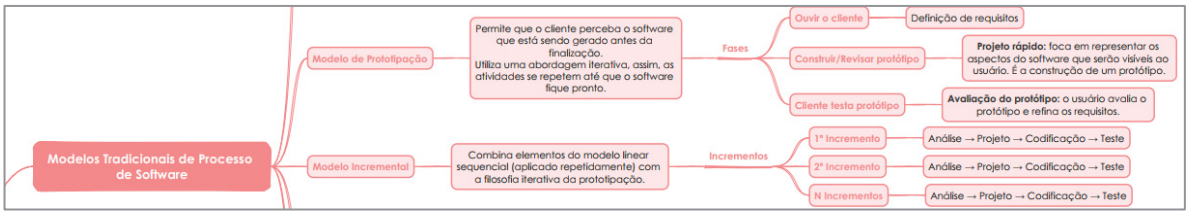
2.1 ARTEFATOS DO PROJETO

A seguir, são apresentados os principais artefatos do projeto, organizados em seções do mapa mental que abordam os modelos de Prototipação e Incremental, o processo de software, os princípios ágeis, o Extreme Programming e as práticas de entrega contínua.

2.1.1 Mapa mental: Modelos Ágeis – Prototipação e Incremental

A figura a seguir representa a seção do mapa mental dedicada aos modelos ágeis de Prototipação e Incremental, evidenciando suas principais fases, características e contribuições para o desenvolvimento iterativo.

FIGURA 1 – FASES E CARACTERÍSTICAS DOS MODELOS DE PROTOTIPAÇÃO E INCREMENTAL

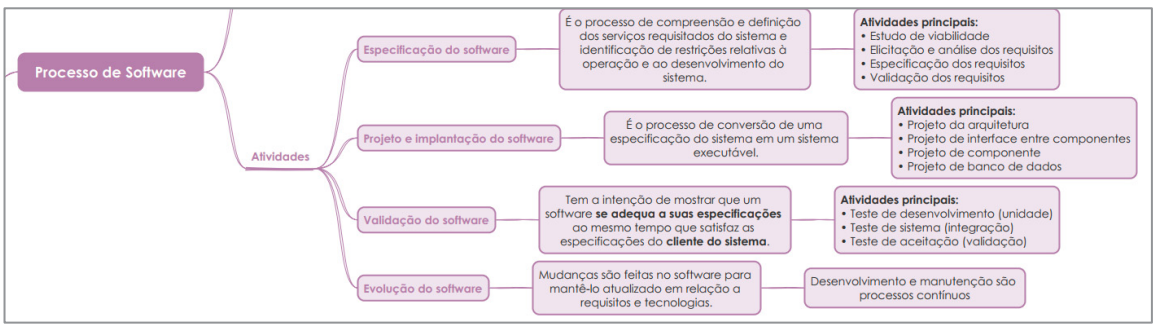


FONTE: A autora (2024).

2.1.2 Mapa mental: Processo de Software – Da Especificação à Evolução

A próxima figura ilustra as principais atividades do processo de software, destacando etapas fundamentais para assegurar a qualidade, a adaptabilidade e a manutenção contínua do sistema ao longo do tempo.

FIGURA 2 – PROCESSO DE SOFTWARE E SUAS PRINCIPAIS ATIVIDADES

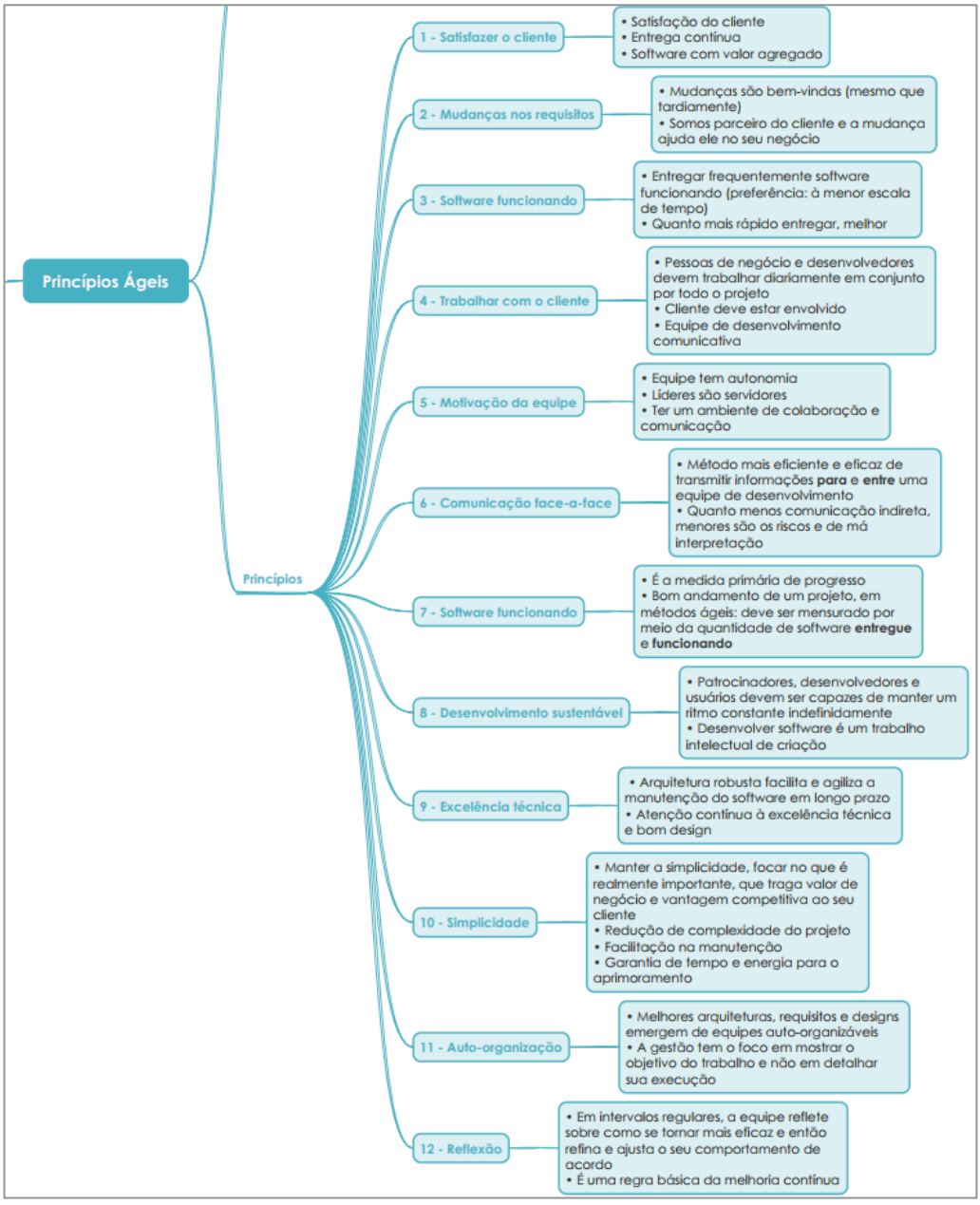


FONTE: A autora (2024).

2.1.3 Mapa mental: Detalhamento dos princípios ágeis

Esta seção apresenta os 12 princípios do Manifesto Ágil, com destaque para a satisfação do cliente, adaptação constante, entregas contínuas, qualidade e simplicidade. Esses princípios sustentam as práticas ágeis, promovendo flexibilidade, eficiência e foco na entrega contínua de valor.

FIGURA 3 – DETALHAMENTO DOS 12 PRINCÍPIOS ÁGEIS, COM ÊNFASE NA SATISFAÇÃO DO CLIENTE, ADAPTAÇÃO, ENTREGA CONTÍNUA, COLABORAÇÃO, QUALIDADE E SIMPLICIDADE

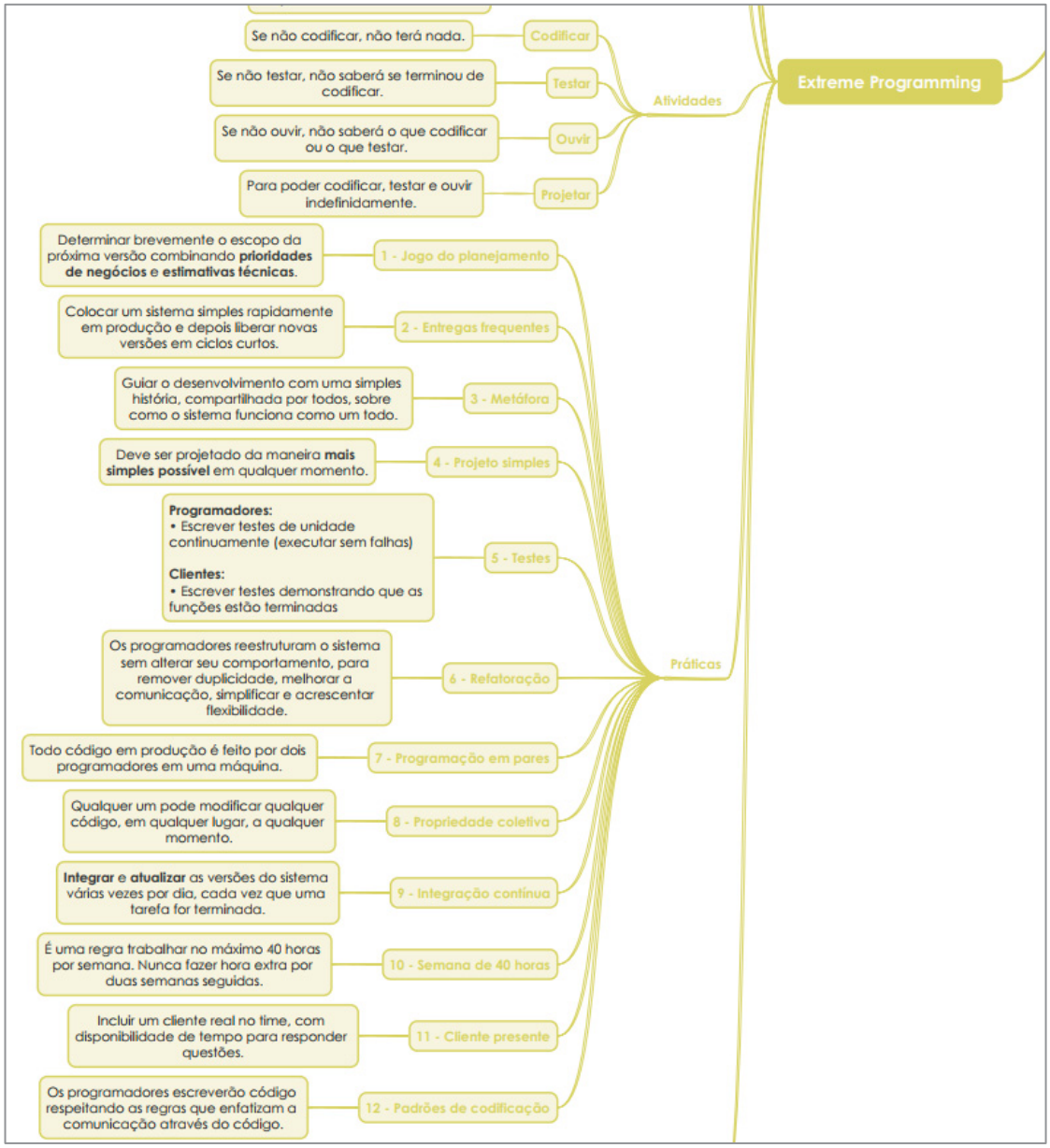


FONTE: A autora (2024).

2.1.4 Mapa mental: Extreme Programming (XP)

O mapa mental detalha as principais práticas do Extreme Programming (XP), com ênfase em entregas frequentes, testes, refatoração de código, integração contínua e a participação ativa do cliente no processo de desenvolvimento. Essas práticas são fundamentais para garantir a qualidade do software e sua evolução contínua, alinhada às necessidades do cliente.

FIGURA 4 – PRINCIPAIS PRÁTICAS DO EXTREME PROGRAMMING (XP), DESTACANDO ENTREGAS FREQUENTES, TESTES, REFATORAÇÃO, INTEGRAÇÃO CONTÍNUA E A PARTICIPAÇÃO DO CLIENTE

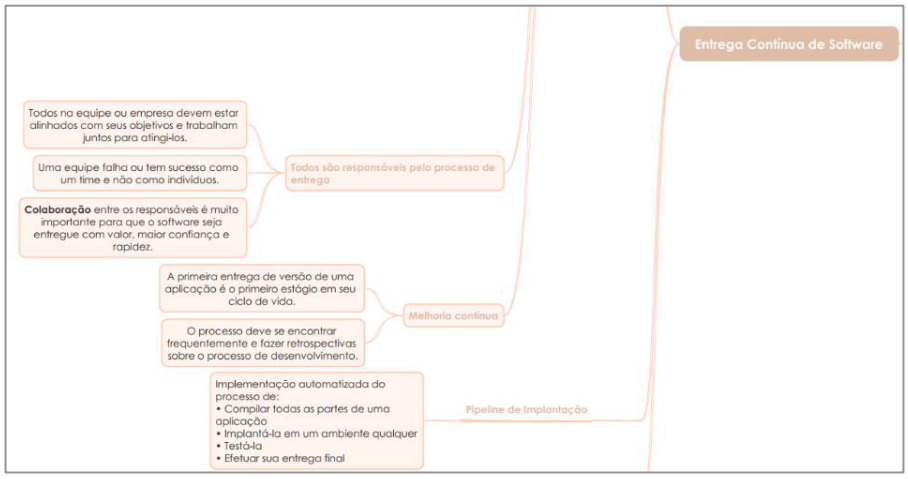


FONTE: A autora (2024).

2.1.5 Mapa mental: Práticas e automação da entrega contínua de software

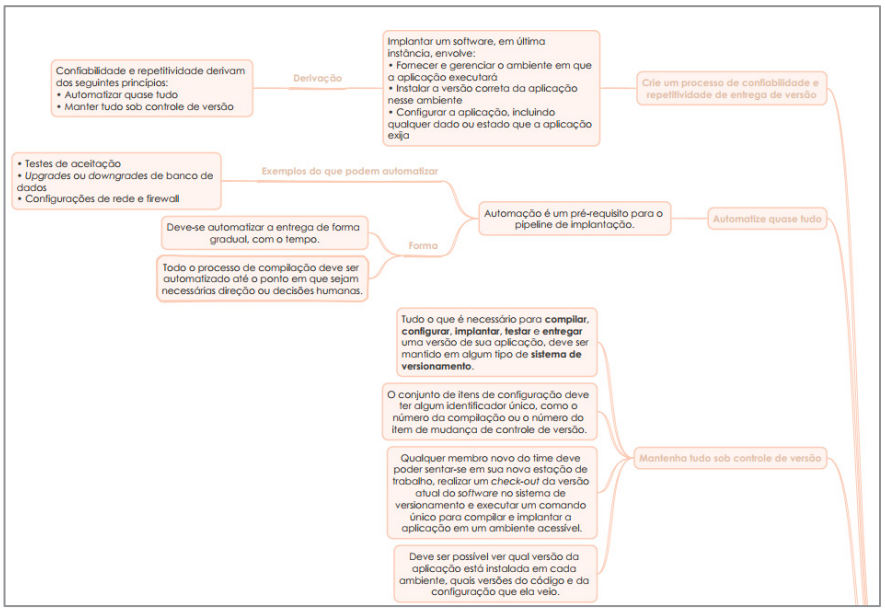
A entrega contínua é uma prática essencial no desenvolvimento ágil, que integra equipes e automatiza o processo de implantação, garantindo entregas rápidas e confiáveis. As figuras 5 e 6 destacam elementos fundamentais dessa prática, como a responsabilidade compartilhada da equipe, a busca por melhoria contínua, a automação de pipelines de implantação e o uso de controle de versão para manter a integridade do software e assegurar processos consistentes e auditáveis.

FIGURA 5 – DESTAQUES DA ENTREGA CONTÍNUA: AUTOMAÇÃO, MELHORIA CONTÍNUA E QUALIDADE



FONTE: A autora (2024).

FIGURA 6 – DESTAQUES DA ENTREGA CONTÍNUA: PROCESSO CONFIÁVEL, AUTOMAÇÃO E CONTROLE DE VERSÃO



FONTE: A autora (2024).

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

O projeto de **Modelagem Ágil de Software 1** teve foco na análise e modelagem funcional de um Sistema de Gestão de Condomínio, incluindo o levantamento de requisitos, elaboração de histórias de usuário, criação de protótipos e diagramas de caso de uso (níveis 1 e 2), além da definição de critérios de aceitação, regras de negócio e observações técnicas. Essas entregas permitem compreender claramente as necessidades do cliente e as funcionalidades esperadas, orientando o desenvolvimento e garantindo o alinhamento às expectativas.

Já em **Modelagem Ágil de Software 2**, essa visão funcional foi expandida com uma abordagem estrutural, utilizando diagramas de classes e de sequência para representar a arquitetura e o fluxo do sistema. Foram elaborados diagramas de classes, contendo atributos e métodos que ilustram as entidades do sistema e seus comportamentos, bem como diagramas de sequência, que representam a interação entre os componentes durante a execução das funcionalidades.

A modelagem ágil valoriza a produção de documentação concisa, eficiente e colaborativa, facilitando a comunicação entre todos os stakeholders e promovendo a adaptabilidade do projeto às mudanças durante o desenvolvimento. Dessa forma, assegura um processo flexível, transparente e focado na entrega contínua de valor. Como destaca Ambler (2002, p. 156), “Agile documents are sufficiently accurate, consistent, and detailed. […] Agile documents do not need to be perfect, they just need to be good enough.”

3.1 ARTEFATOS DO PROJETO

A seguir, são apresentados os principais artefatos dos projetos, incluindo diagramas de caso de uso, diagrama de sequência, diagramas de classes e desenhos das telas. Esses elementos fornecem uma visão estruturada do sistema, abrangendo sua funcionalidade, interface e arquitetura.

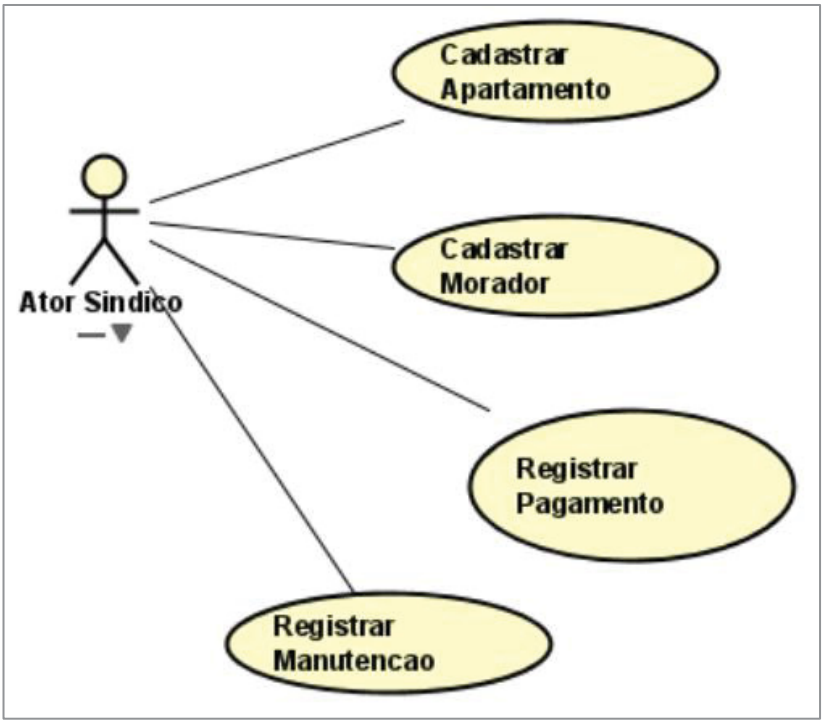
3.1.1 Diagramas de Caso de Uso

Os diagramas de caso foram elaborados para representar as funcionalidades de um sistema sob a perspectiva dos usuários. Eles foram divididos em dois níveis: um que fornece uma visão geral do sistema e o outro que detalha as interações específicas entre o usuário e o sistema.

3.1.1.1 Diagrama de Caso de Uso – Nível 1

O Diagrama de Nível 1 apresenta uma visão ampla do Sistema de Gestão de Condomínio, destacando o ator principal e os casos de uso. Ele proporciona uma compreensão das funcionalidades do sistema, sem detalhar aspectos operacionais.

FIGURA 7 – DIAGRAMA DE CASO DE USO – NÍVEL 1

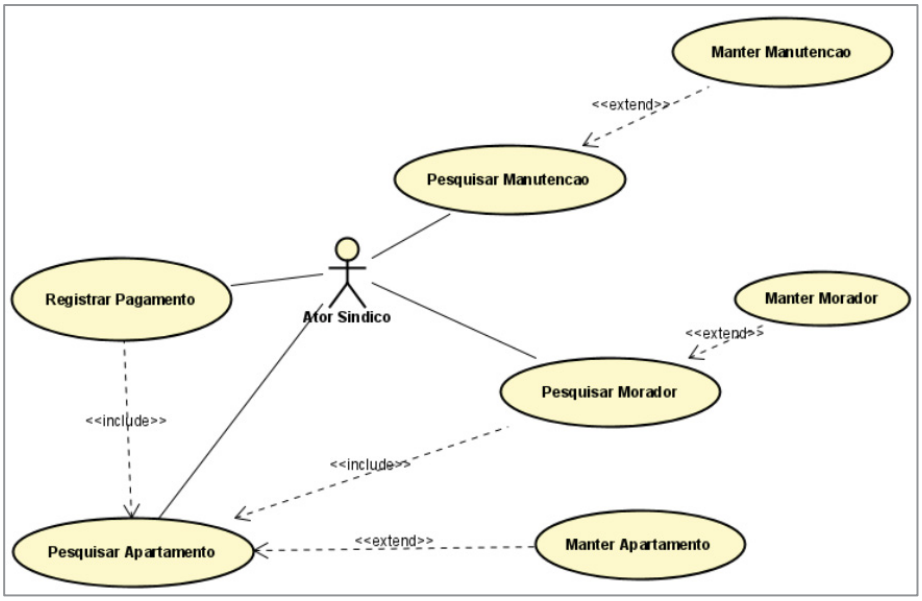


FONTE: A autora (2024).

3.1.1.2 Diagrama de Caso de Uso – Nível 2

O Diagrama de Nível 2 aprofunda-se nas histórias de usuário e nas interações específicas entre o ator e o sistema. Essa visão detalhada permite identificar claramente os fluxos de uso e o comportamento esperado de cada funcionalidade.

FIGURA 8 – DIAGRAMA DE CASO DE USO – NÍVEL 2

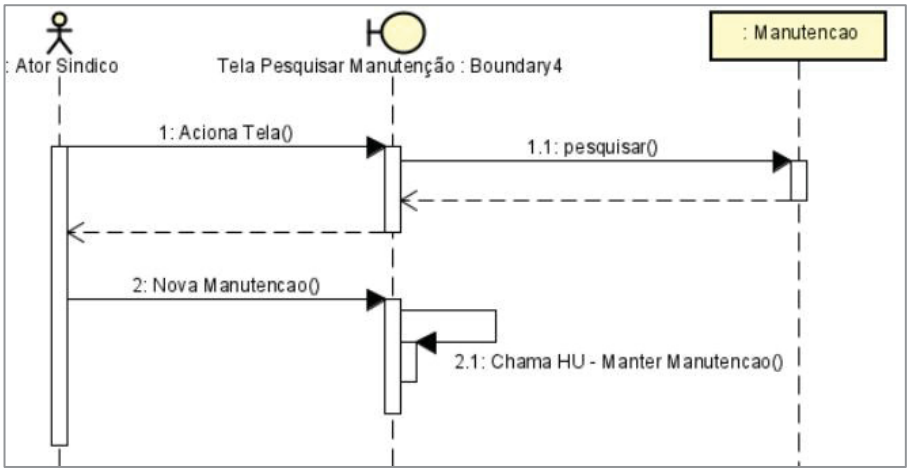


FONTE: A autora (2024).

3.1.2 Diagrama de Sequência – Pesquisa de Manutenção

O diagrama a seguir ilustra a sequência de interações entre os componentes do sistema durante a execução da funcionalidade de pesquisa de manutenção.

FIGURA 9 – DIAGRAMA DE SEQUÊNCIA: PESQUISAR MANUTENÇÃO



FONTE: A autora (2024).

3.1.3 Tela de Pesquisa de Apartamento

A tela de pesquisa de apartamento possibilita uma busca rápida e eficiente pelos imóveis, contando com campo de pesquisa, listagem organizada e botões para criação, edição e exclusão de registros.

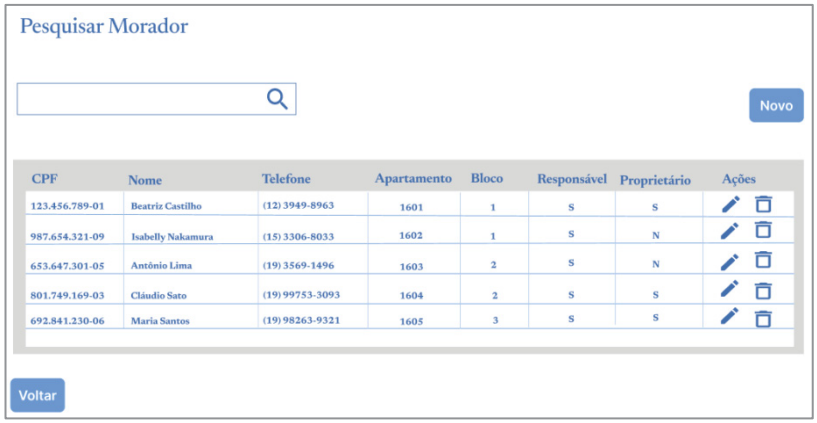
FIGURA 10 – TELA DE PESQUISA DE APARTAMENTO NO SISTEMA DE GESTÃO DE CONDOMÍNIO

FONTE: A autora (2024).

3.1.4 Tela de Pesquisa de Morador

A tela de pesquisa de morador segue o mesmo padrão da anterior, com interface clara e recursos para consulta, editação e gerenciamento ágil de moradores.

FIGURA 11 – TELA DE PESQUISA DE MORADOR NO SISTEMA DE GESTÃO DE CONDOMÍNIO



FONTE: A autora (2024).

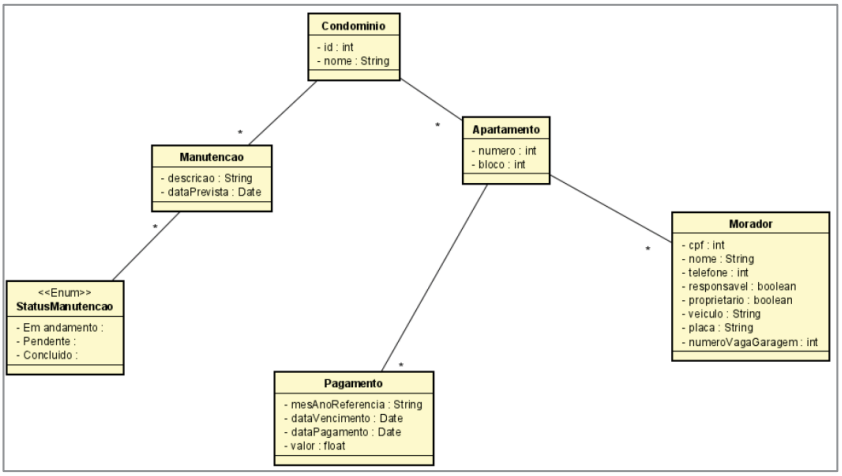
3.1.5 Diagramas de Classes

Esta seção apresenta os diagramas que descrevem a estrutura do sistema, destacando suas entidades, os relacionamento entre elas e seus principais atributos.

3.1.5.1 Diagrama de Classe – Visão geral do Sistema

O diagrama a seguir apresenta as principais entidades e os relacionamentos entre elas, evidenciando a estrutura a geral e interconexão dos componentes.

FIGURA 12 – DIAGRAMA DE CLASSE 1: VISÃO GERAL DAS ENTIDADES E SEUS RELACIONAMENTOS

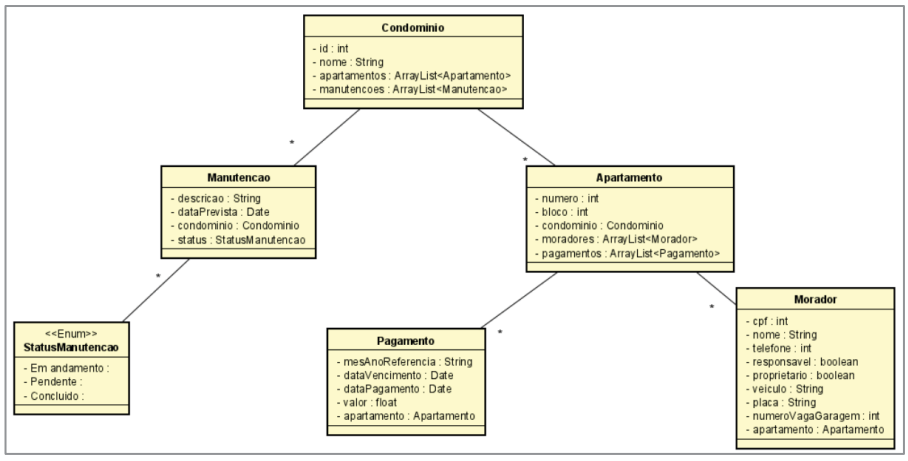


FONTE: A autora (2024).

3.1.5.2 Diagrama de Classe – Visão geral com atributos

Ilustra os atributos das classes, ampliando o entendimento do sistema.

FIGURA 13 – DIAGRAMA DE CLASSE 2: DETALHAMENTO DAS CLASSES COM SEUS ATRIBUTOS



FONTE: A autora (2024).

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

O projeto de **Gerenciamento Ágil de Projetos de Software 1** teve como base a metodologia Scrum para a elaboração de um plano de *release* detalhado. Esse plano incluiu o cálculo da velocidade, considerando as horas disponíveis por dia, por Sprint, o tamanho da Sprint e a velocidade da Sprint. Além disso, contemplou estimativas de histórias de usuário em pontos e a definição das datas de início e fim de cada Sprint. O planejamento assumiu que o desenvolvimento seria realizado por um único profissional.

O projeto de **Gerenciamento Ágil de Software 2** adotou o método Kanban, simulando a execução de um ciclo de 35 dias no site kanbanboardgame.com. A simulação acompanhou o fluxo contínuo de trabalho, no qual as tarefas transitaram por colunas que representam diferentes fases do processo, como *Backlog*, *Ready*, *Analysis*, *Testing* e *Deployed*. Os resultados da simulação foram avaliados por meio do Diagrama de Fluxo Cumulativo (CFD), que permite visualizar o andamento das atividades, controlar o trabalho em progresso (WIP) e identificar gargalos no fluxo.

A aplicação dessas metodologias ágeis proporciona maior flexibilidade e eficiência no desenvolvimento de software. O Scrum favorece entregas iterativas e incrementais, estruturando e priorizando as tarefas de forma organizada. Como destaca o *Scrum Guide* (2020, p.3), “O Scrum torna visível a eficácia relativa da gestão, ambiente e técnicas de trabalho atuais, de modo a que se possam fazer melhorias.” O Kanban, por sua vez, promove maior eficiência por meio do controle visual das atividades e da limitação do WIP, contribuindo para a transparência do progresso e antecipando possíveis problemas que possam impactar o cumprimento dos prazos. Como ressalta o Guia Oficial do Kanban (2021, p.6), “O monitoramento ou medição do fluxo de trabalho resulta em informações importantes que são muito úteis para a gestão das expectativas com os clientes, para a previsão e melhorias.”

4.1 ARTEFATOS DO PROJETO

Esta seção apresenta os artefatos dos projetos, incluindo o plano de *release* elaborado com base no Scrum para o Sistema de Gestão de Condomínio, bem como a simulação do fluxo de trabalho realizada por meio do *Kanban Board Game*.

4.1.1 Plano de Release – Scrum: Sistema de Gestão de Condomínio

O planejamento de *release* do projeto foi desenvolvido com base na metodologia Scrum, dividido em cinco iterações (Sprints) de três semanas cada, com carga diária de 6 horas e uma velocidade média de 11 pontos por Sprint.

A tabela detalha as datas de início e fim de cada Sprint, as histórias de usuário formuladas no padrão “Sendo... Quero... Para...”, as estimativas de esforço em pontos, além do cálculo da velocidade e da distribuição das tarefas ao longo de cada Sprint. Este planejamento foi essencial para organizar o desenvolvimento e garantir uma visão clara das entregas.

A seguir, são apresentados os detalhes do plano elaborado.

QUADRO 1 – CÁLCULO DA VELOCIDADE

Cálculo da velocidade:

Horas disponíveis por dia:	6 horas	**Tamanho da Sprint:**	3 semanas
Horas disponíveis por Sprint:	90 horas	**Velocidade:**	11

FONTE: A autora (2024).

QUADRO 2 – PLANO DE RELEASE: SPRINTS E ITERAÇÕES (CONTINUA)

Plano de *Release:*

Iteração/Sprint 1	Iteração/Sprint 2
Data Início: 01/07/2024	Data Início: 22/07/2024
Data Fim: 19/07/2024	Data Fim: 09/08/2024
< HU001 – Pesquisar Apartamento> **SENDO** o Síndico **QUERO** pesquisar os apartamentos **PARA** fazer manutenções nos seus dados ESTIMATIVA (2)	< HU003 – Pesquisar Morador> **SENDO** o Síndico **QUERO** pesquisar os moradores **PARA** fazer manutenções nos seus dados ESTIMATIVA (3)

QUADRO 2 – PLANO DE RELEASE: SPRINTS E ITERAÇÕES (CONCLUSÃO)

< HU002 – Manter Apartamento> **SENDO** o Síndico **QUERO** manter os dados dos apartamentos **PARA** que seus dados fiquem atualizados ESTIMATIVA (8)	< HU004 – Manter Morador> **SENDO** o Síndico **QUERO** manter os dados dos moradores **PARA** que seus dados fiquem atualizados ESTIMATIVA (8)
Iteração/Sprint 3	**Iteração/Sprint 4**
Data Início: 12/08/2024	Data Início: 02/09/2024
Data Fim: 30/08/2024	Data Fim: 20/09/2024
< HU007 – Manter Manutenção> **SENDO** o Síndico **QUERO** manter os dados das manutenções **PARA** que seus dados fiquem atualizados ESTIMATIVA (8)	< HU005 – Registrar Pagamento> **SENDO** o Síndico **QUERO** registrar o pagamento do condomínio **PARA** que tenha os registros dos moradores que pagaram o condomínio ESTIMATIVA (8)
< HU006 – Pesquisar Manutenção> **SENDO** o Síndico **QUERO** pesquisar as manutenções do prédio **PARA** fazer manutenções nos seus dados ESTIMATIVA (2)	-

Iteração/Sprint 5
Data Início: 23/09/2024
Data Fim: 11/10/2024
< HU009 – Manter Funcionário> **SENDO** o Síndico **QUERO** manter os dados dos funcionários do condomínio **PARA** que seus dados fiquem atualizados ESTIMATIVA (8)
< HU009 – Pesquisar Funcionário> **SENDO** o Síndico **QUERO** pesquisar os funcionários do condomínio **PARA** fazer manutenções nos seus dados ESTIMATIVA (3)

FONTE: A autora (2024).

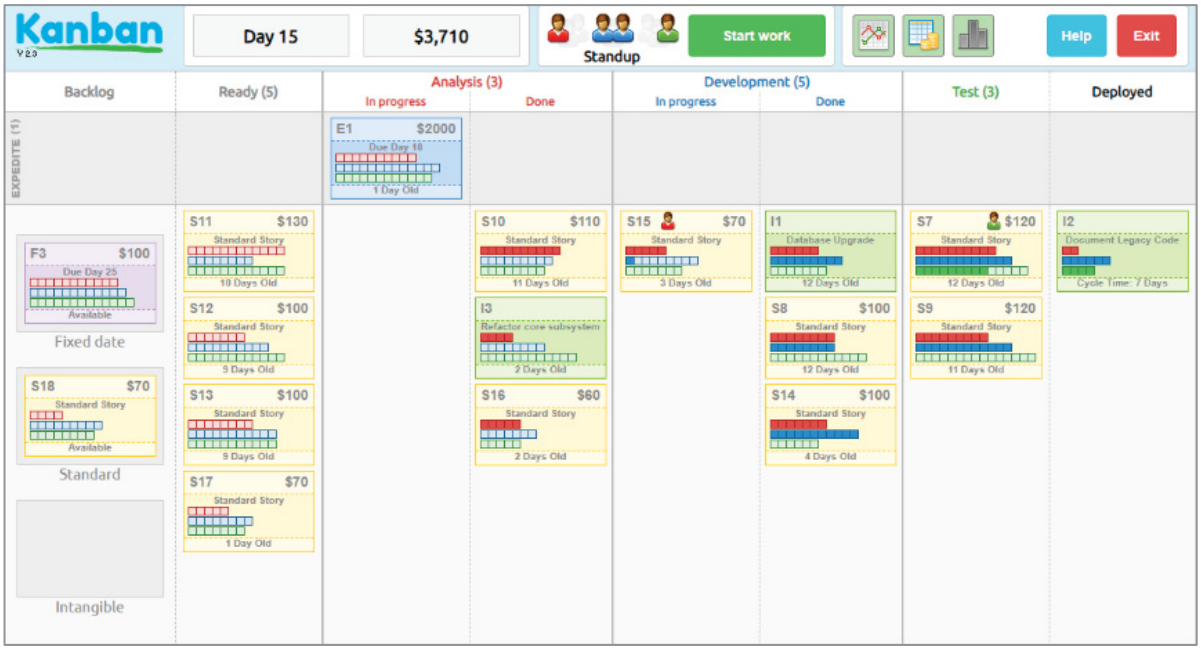
4.1.2 Simulação com Kanban Board Game

No projeto, utilizou-se o *Kanban Board Game* para simular o fluxo contínuo de trabalho, movimentando as histórias entre as colunas *Backlog*, *Ready*, *Analysis*, *Testing* e *Deployed*. A simulação permitiu acompanhar o progresso ao longo de 35 dias, aplicando conceitos fundamentais do Kanban, como a limitação do trabalho em progresso (WIP*)* e a gestão eficiente do fluxo. A cada dia, eventos aleatórios influenciaram o andamento das atividades, exigindo o remanejamento estratégico de analistas, desenvolvedores e testadores para garantir que as entregas fossem realizadas dentro do prazo estimado.

4.1.2.1 Acompanhamento diário

A figura a seguir mostra a execução do dia 15 no Kanban, destacando o andamento das tarefas, o controle de WIP, o status das atividades, o valor monetário acumulado e a distribuição nas colunas do fluxo de trabalho.

FIGURA 14 – VISÃO DO PROGRESSO DO DIA 15

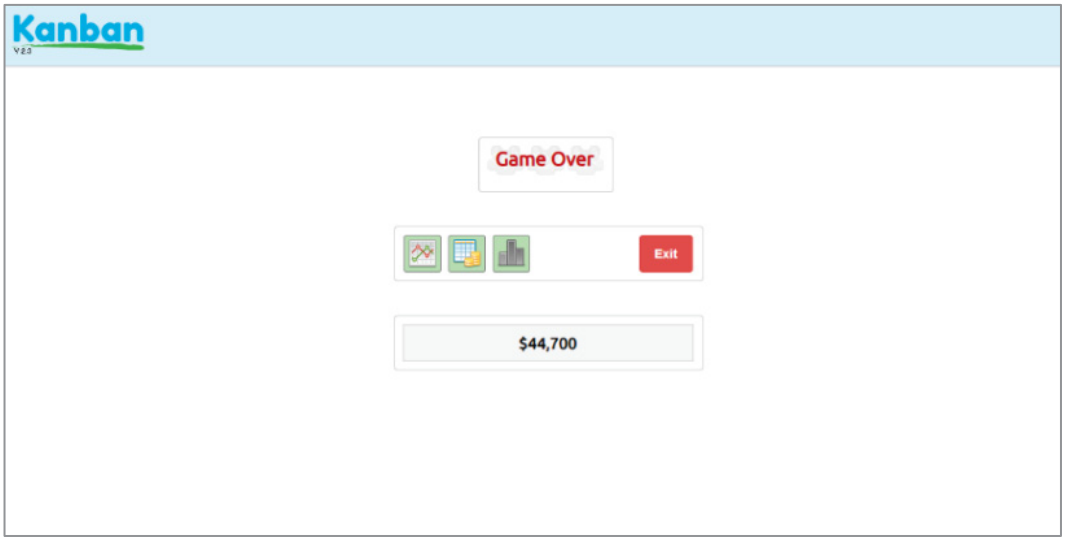


FONTE: A autora (2024).

4.1.2.2 Resultado final

A figura a seguir ilustra a receita total obtida ao final da simulação com o *Kanban Board Game*.

FIGURA 15 – RECEITA TOTAL DA SIMULAÇÃO: $44,700

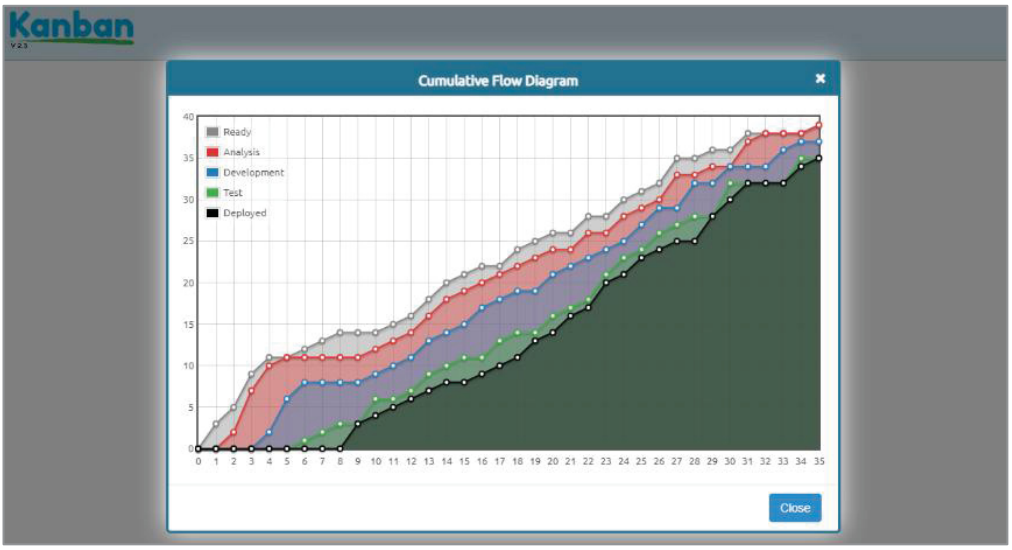


FONTE: A autora (2024).

4.1.2.3 Diagrama de Fluxo Cumulativo (CFD)

Para uma análise abrangente do fluxo de trabalho e do progresso de tarefas, apresenta-se o Diagrama de Fluxo Cumulativo (CFD) da simulação Kanban, referente aos 35 dias de execução.

FIGURA 16 – DIAGRAMA DE FLUXO CUMULATIVO (CFD)



FONTE: A autora (2024).

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

O projeto consistiu na implementação de um back-end de um sistema bancário simplificado, focado no gerenciamento de clientes, contas correntes e contas de investimentos. O desenvolvimento seguiu uma estrutura previamente definida, composta por artefatos fornecidos pelo professor: um diagrama de classes que representa a modelagem orientada a objetos do sistema, um diagrama entidade-relacionamento (ER), um script DDL para criação das tabelas no banco de dados MySQL, além de um projeto no ambiente NetBeans contendo 42 testes unitários já escritos com JUnit.

A principal meta foi implementar as classes do sistema de modo que, ao final, pelo menos 40 dos 42 testes unitários fossem executados com sucesso simultaneamente. Esses testes funcionaram como critério objetivo de validação, avaliando a aderência da implementação aos requisitos estabelecidos.

A persistência de dados foi realizada por meio do banco de dados relacional MySQL, utilizando as tabelas criadas a partir do script DDL disponibilizado.

A adoção de testes unitários desde o início do desenvolvimento reforça práticas essenciais da engenharia de software, facilitando a identificação precoce de erros, promovendo validações contínuas e assegurando que o sistema evolua com qualidade e foco em entregas incrementais. Conforme ressalta Hirama (2011, p.3), “todo projeto de software, além de bem gerenciado e desenvolvido, precisa garantir a qualidade do produto por meio de técnicas de verificação e validação, gerenciamento de configuração e de qualidade”.

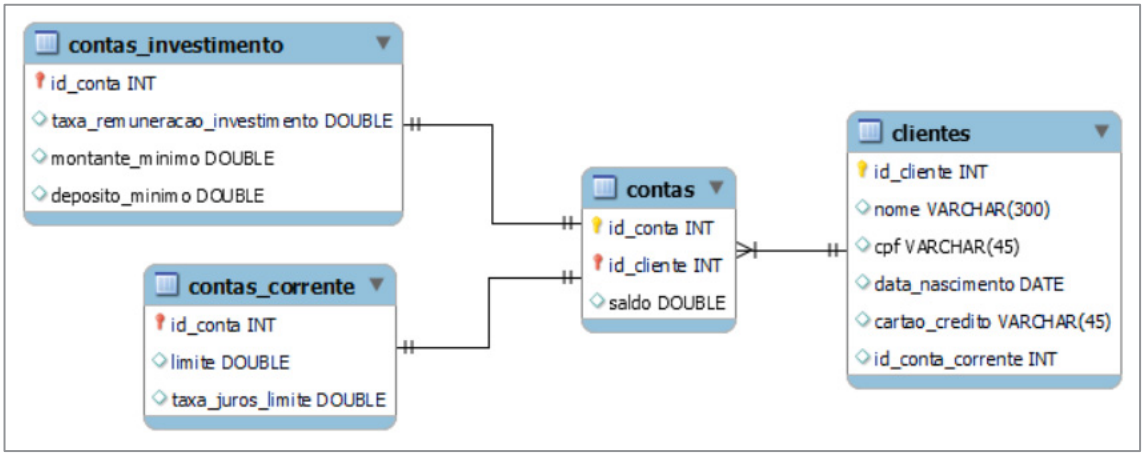
5.1 ARTEFATOS DO PROJETO

Esta seção apresenta os principais artefatos desenvolvidos e utilizados ao longo do projeto, incluindo a modelagem do banco de dados, a organização da estrutura do código, a implementação da classe Pessoa e os resultados dos testes unitários.

5.1.1 Diagrama de Entidade e Relacionamento

O Diagrama Entidade-Relacionamento (ER) apresenta a modelagem do banco de dados do sistema, destacando as principais entidades, como **Clientes**, **Contas Corrente** e **Contas de Investimento**, bem como os relacionamentos entre elas. Esse diagrama, fornecido pelo professor, serviu como referência para a criação das tabelas e definição da estrutura de dados.

FIGURA 17 – DIAGRAMA ER FORNECIDO PELO PROFESSOR, BASE PARA A MODELAGEM DO BANCO DE DADOS DO SISTEMA BANCÁRIO



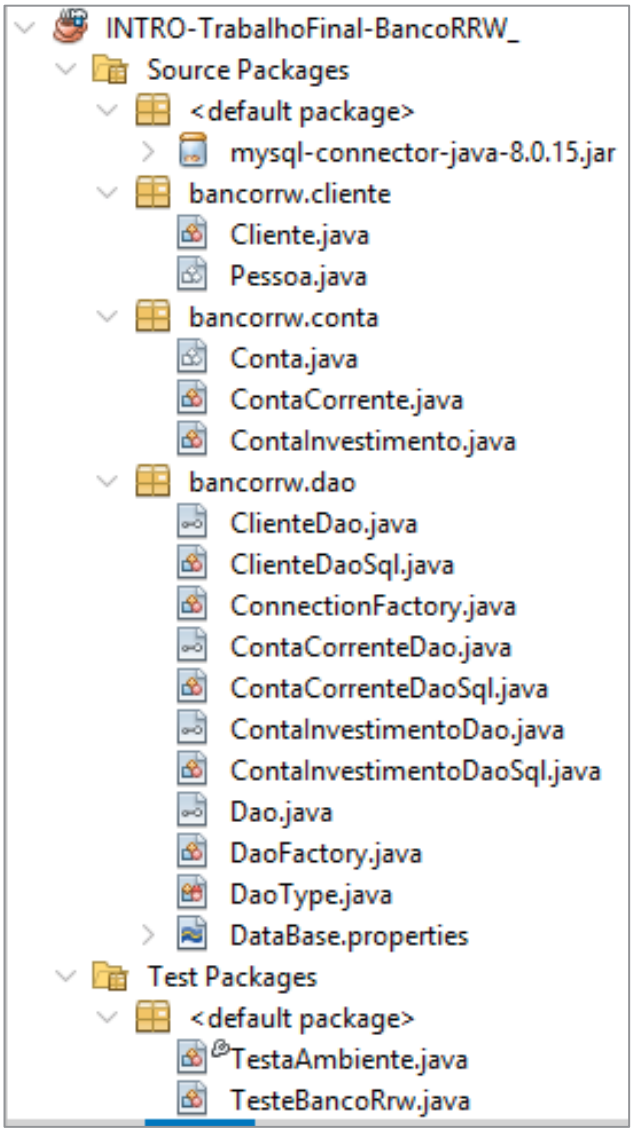
FONTE: Wandresen (2024).

5.1.2 Estrutura do Projeto

O projeto foi dividido em pacotes conforme a função de cada parte, seguindo a organização fornecida pelo professor. As classes de modelo representam os dados principais do sistema, enquanto a persistência é realizada pelo padrão DAO. A conexão com **MySQL** é feita pela classe ConnectionFactory, e os testes estão em um pacote separado para validar o funcionamento das funcionalidades.

A figura a seguir apresenta a estrutura de pastas do projeto.

FIGURA 18 – ESTRUTURA DE PACOTES DO PROJETO DISPONIBILIZADA PELO PROFESSOR COMO BASE PARA O DESENVOLVIMENTO DO SISTEMA



FONTE: Wandresen (2024).

5.1.3 Implementação da Classe Pessoa

A classe **Pessoa** é uma classe abstrata que serve como base para outras entidades, como **Cliente**. Ela define atributos como **id**, **nome**, **cpf** e **dataNascimento**, além de métodos para acesso e modificação desses dados. Dessa forma, facilita o reaproveitamento de código e padroniza as informações do sistema.

A figura a seguir apresenta o código da classe **Pessoa**.

FIGURA 19 – IMPLEMENTAÇÃO DA CLASSE PESSOA, RESPONSÁVEL POR REPRESENTAR E GERENCIAR OS DADOS PESSOAIS DO SISTEMA

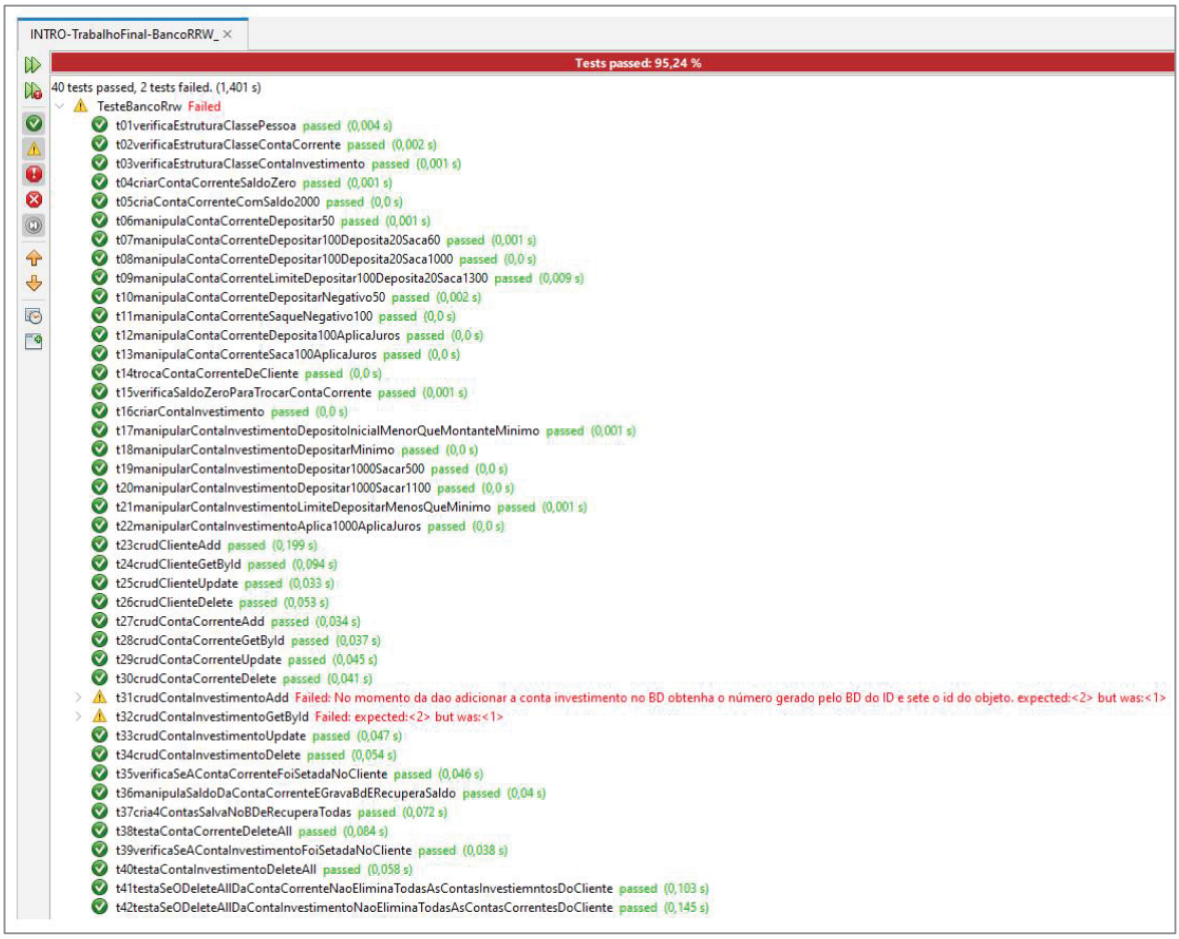
```
public abstract class Pessoa {

    private long id;
    private String nome;
    private String cpf;
    private LocalDate dataNascimento;

    public Pessoa(long id, String nome, String cpf, LocalDate dataNascimento) {
        this.id = id;
        this.nome = nome;
        this.cpf = cpf;
        this.dataNascimento = dataNascimento;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public LocalDate getDataNascimento() {
        return dataNascimento;
    }

    public void setDataNascimento(LocalDate dataNascimento) {
        this.dataNascimento = dataNascimento;
    }
}
```

FONTE: A autora (2024).

5.1.4 Execução de testes unitários do sistema bancário

A seguir, apresenta-se o relatório gerado após a execução dos testes unitários do sistema bancário. Dos 42 testes disponibilizados, 40 foram executados com sucesso, enquanto 2 apresentaram falhas. Esse resultado evidencia o progresso do desenvolvimento e valida aspectos essenciais do funcionamento do sistema.

FIGURA 20 – RELATÓRIO DE EXECUÇÃO DOS TESTES UNITÁRIOS NO NETBEANS



FONTE: A autora (2024).

6 DISCIPLINA: BD – BANCO DE DADOS

O desenvolvimento do projeto foi estruturado em duas etapas principais, com o objetivo de aplicar na prática os conceitos de modelagem e implementação de bancos de dados relacionais.

Na primeira etapa, propôs-se modelar um sistema de controle de biblioteca com base em requisitos fornecidos pelo professor. A partir dessa descrição, foi elaborado o Modelo Entidade-Relacionamento (MER), que representa conceitualmente as entidades envolvidas — como obras, editoras, usuários, funcionários, departamentos, empréstimos e reservas — e seus respectivos relacionamentos. Essa modelagem foi realizada com o uso da ferramenta brModelo, garantindo clareza na estrutura conceitual. Em seguida, o modelo foi transformado em um Modelo Lógico, com a definição das tabelas, atributos, tipos de dados, chaves primárias, estrangeiras e regras de cardinalidade.

Na segunda etapa, o projeto propõe a escolha de um tema livre, no qual a modelagem lógica foi aplicada novamente. Com base nesse modelo, foi elaborado um script SQL completo, contendo os comandos de criação das tabelas, definição de campos, chaves primárias, chaves estrangeiras e restrições de integridade. A estrutura foi validada por meio da inserção de registros de exemplos, demonstrando o comportamento dos diferentes tipos de relacionamentos (1:1, 1:N e N:N) na prática, com foco na consistência dos dados e no funcionamento adequado das relações entre as tabelas.

No contexto do desenvolvimento ágil, a modelagem de dados desempenha um papel essencial ao possibilitar uma estrutura flexível e em constante evolução, capaz de acompanhar as mudanças nos requisitos do sistema.

> O processo de construção de um modelo é um processo incremental, isto é, um modelo de um sistema não é construído em um único passo, mas em muitos passos pequenos, muitas pequenas transformações do modelo inicial até o modelo completo. Gradativamente, o modelo vai sendo enriquecido com novos conceitos e estes vão sendo ligados aos existentes ou os existentes vão sendo aperfeiçoados. (HEUSER, 2004, não.p.).

Essa abordagem incremental favorece a adaptação contínua do modelo, o que está alinhado aos princípios de entregas evolutivas e melhoria constante promovidos pelas metodologias ágeis.

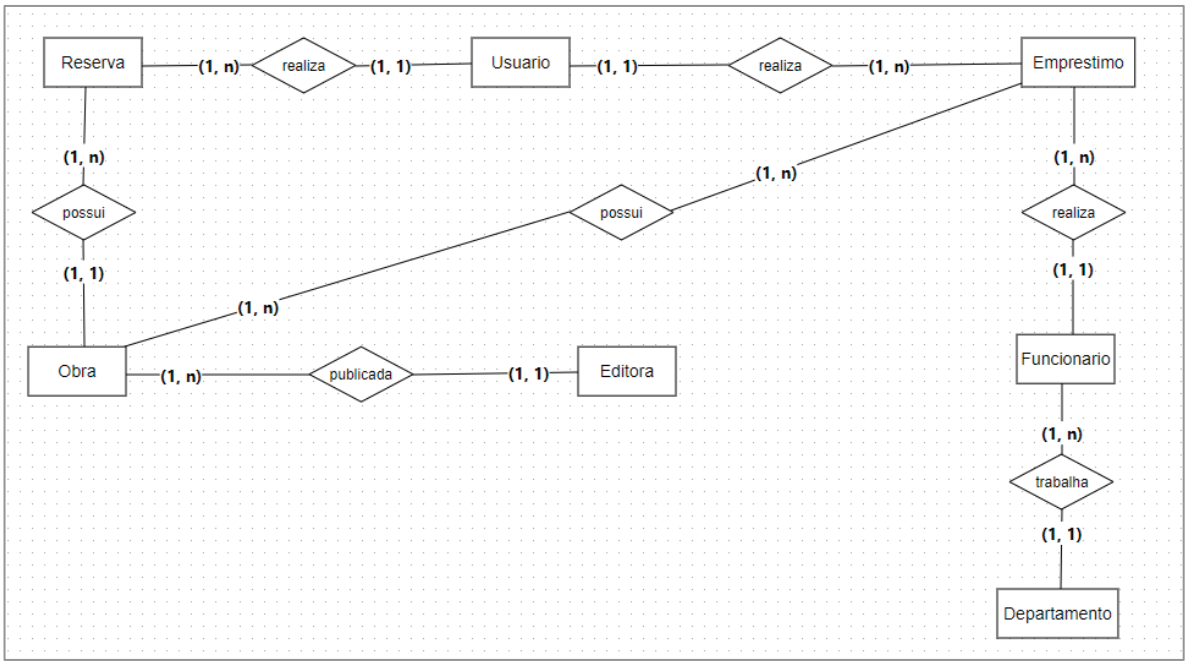
6.1 ARTEFATOS DO PROJETO

A seguir, são apresentados os principais artefatos desenvolvidos no decorrer projeto. Dentre eles, destacam-se o modelo entidade-relacionamento (conceitual e lógico), os scripts SQL implementados e os registros inseridos nas tabelas do banco de dados.

6.1.1 Modelo Entidade-Relacionamento Conceitual – Biblioteca

O modelo conceitual a seguir representa as principais entidades e os relacionamentos identificados no sistema da Biblioteca. Ele foi elaborado com base nos requisitos do sistema e desenvolvido utilizando a ferramenta brModelo, servindo como base para a construção do modelo lógico.

FIGURA 21 – MODELO CONCEITUAL – BIBLIOTECA

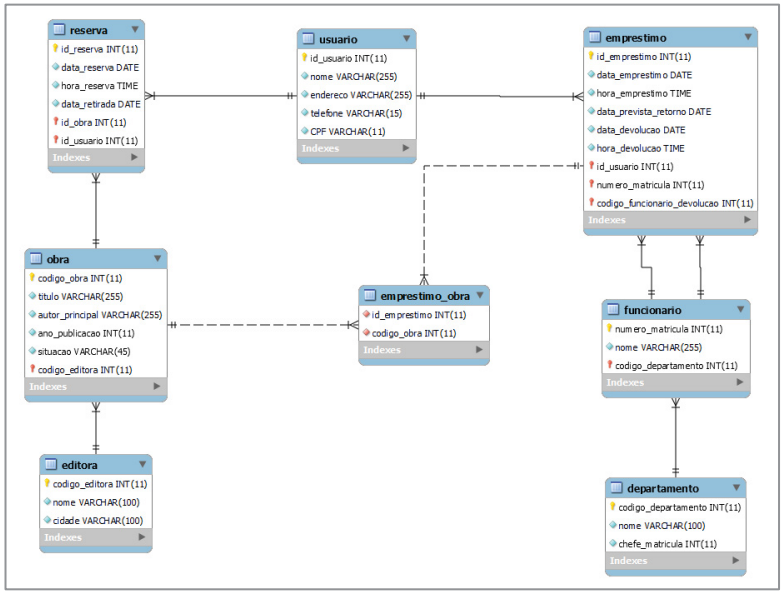


FONTE: A autora (2024).

6.1.2 Modelo Lógico – Biblioteca e Gestão Acadêmica

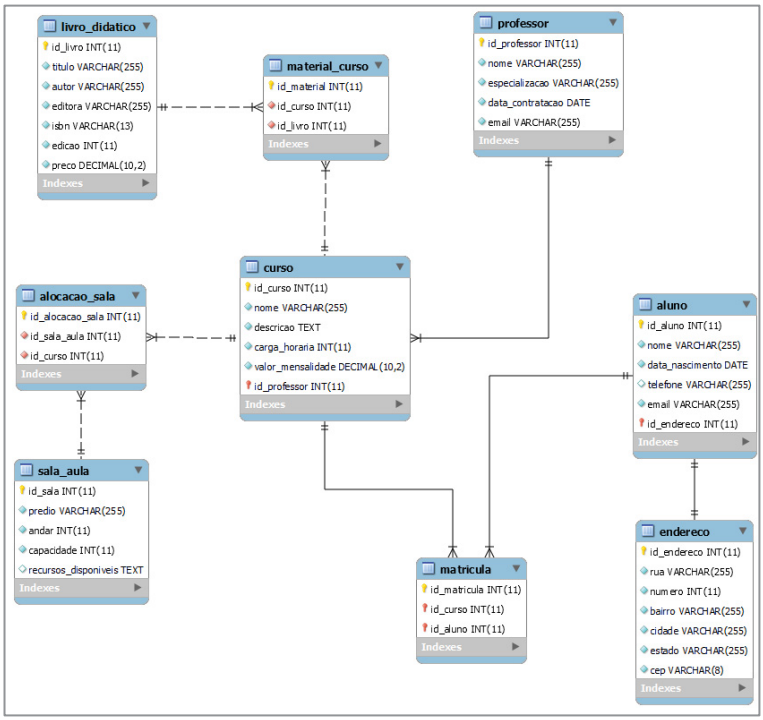
Os diagramas a seguir apresentam as tabelas e os relacionamentos dos sistemas da Biblioteca e da Gestão Acadêmica.

FIGURA 22 – MODELO LÓGICO – BIBLIOTECA



FONTE: A autora (2024).

FIGURA 23 – MODELO LÓGICO – GESTÃO ACADÊMICA



FONTE: A autora (2024).

6.1.3 Scripts SQL

A seguir, apresentam-se os scripts para a criação das tabelas principais do sistema de Gestão Acadêmica, incluindo cursos, alunos, professores e livros didáticos.

FIGURA 24 – SCRIPT SQL DA TABELA CURSO

```sql
CREATE TABLE curso (
  id_curso INT(11) PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  descricao TEXT NOT NULL,
  carga_horaria INT(11) NOT NULL,
  valor_mensalidade DECIMAL(10,2) NOT NULL,
  id_professor INT(11) NOT NULL,
  FOREIGN KEY (id_professor) REFERENCES professor(id_professor)
);
```

FONTE: A autora (2024).

FIGURA 25 – SCRIPT SQL DA TABELA ALUNO

```sql
CREATE TABLE aluno (
  id_aluno INT(11) PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  data_nascimento DATE NOT NULL,
  telefone VARCHAR(255),
  email VARCHAR(255) NOT NULL,
  id_endereco INT(11) NOT NULL,
  FOREIGN KEY (id_endereco) REFERENCES endereco(id_endereco)
);
```

FONTE: A autora (2024).

FIGURA 26 – SCRIPT SQL DA TABELA PROFESSOR

```sql
CREATE TABLE professor (
  id_professor INT(11) PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  especializacao VARCHAR(255) NOT NULL,
  data_contratacao DATE NOT NULL,
  email VARCHAR(255) NOT NULL
);
```

FONTE: A autora (2024).

FIGURA 27 – SCRIPT SQL DA TABELA LIVRO DIDÁTICO

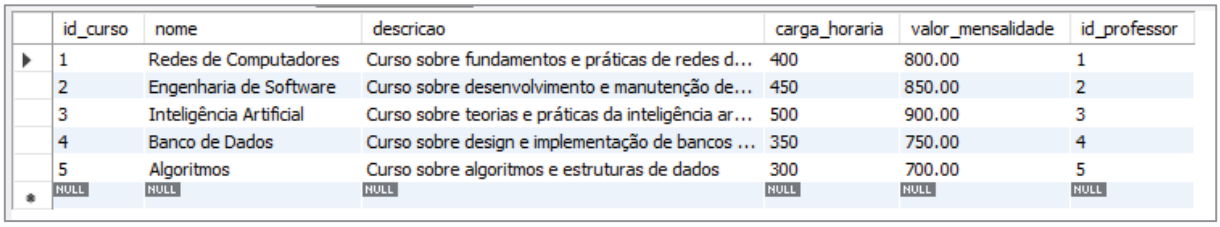
```sql
CREATE TABLE livro_didatico (
  id_livro INT(11) PRIMARY KEY AUTO_INCREMENT,
  titulo VARCHAR(255) NOT NULL,
  autor VARCHAR(255) NOT NULL,
  editora VARCHAR(255) NOT NULL,
  isbn VARCHAR(13) NOT NULL,
  edicao INT(11) NOT NULL,
  preco DECIMAL(10,2) NOT NULL
);
```

FONTE: A autora (2024).

6.1.4 Registros no Banco de Dados

Apresentam-se exemplos dos dados inseridos nas tabelas do sistema de Gestão Acadêmica, ilustrando o preenchimento inicial e o funcionamento do banco.

FIGURA 28 – DADOS DA TABELA CURSO NO BANCO DE DADOS



id_curso	nome	descricao	carga_horaria	valor_mensalidade	id_professor
1	Redes de Computadores	Curso sobre fundamentos e práticas de redes d...	400	800.00	1
2	Engenharia de Software	Curso sobre desenvolvimento e manutenção de...	450	850.00	2
3	Inteligência Artificial	Curso sobre teorias e práticas da inteligência ar...	500	900.00	3
4	Banco de Dados	Curso sobre design e implementação de bancos ...	350	750.00	4
5	Algoritmos	Curso sobre algoritmos e estruturas de dados	300	700.00	5
NULL	NULL	NULL	NULL	NULL	NULL

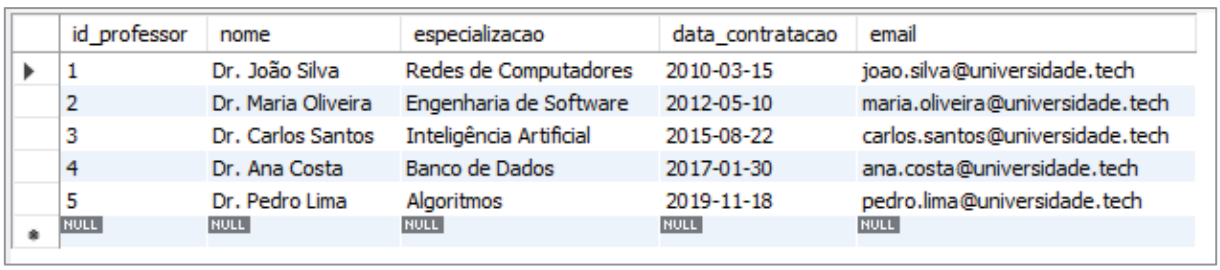
FONTE: A autora (2024).

FIGURA 29 – DADOS DA TABELA ALUNO NO BANCO DE DADOS

id_aluno	nome	data_nascimento	telefone	email	id_endereco
1	Alice Souza	1998-05-10	11987654321	alice.souza@universidade.tech	1
2	Bruno Lima	1997-07-20	21987654321	bruno.lima@universidade.tech	2
3	Carla Rodrigues	1996-09-15	31987654321	carla.rodrigues@universidade.tech	3
4	Daniela Ferreira	1995-11-30	41987654321	daniela.ferreira@universidade.tech	4
5	Eduarda Nakamura	1999-01-25	51987654321	eduarda.nakamura@universidade.tech	5
NULL	NULL	NULL	NULL	NULL	NULL

FONTE: A autora (2024).

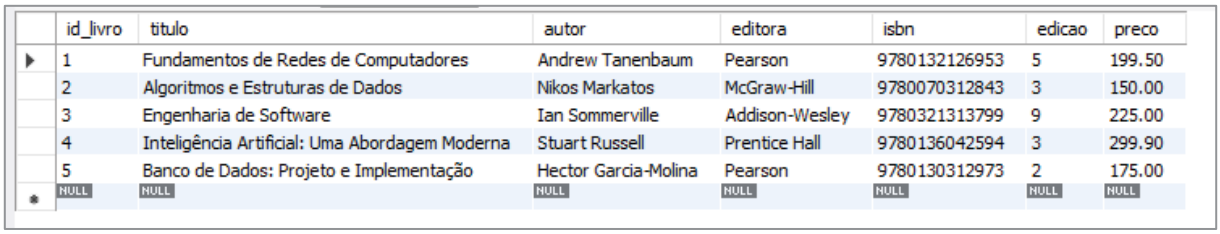
FIGURA 30 – DADOS DA TABELA PROFESSOR NO BANCO DE DADOS



id_professor	nome	especializacao	data_contratacao	email
1	Dr. João Silva	Redes de Computadores	2010-03-15	joao.silva@universidade.tech
2	Dr. Maria Oliveira	Engenharia de Software	2012-05-10	maria.oliveira@universidade.tech
3	Dr. Carlos Santos	Inteligência Artificial	2015-08-22	carlos.santos@universidade.tech
4	Dr. Ana Costa	Banco de Dados	2017-01-30	ana.costa@universidade.tech
5	Dr. Pedro Lima	Algoritmos	2019-11-18	pedro.lima@universidade.tech
NULL	NULL	NULL	NULL	NULL

FONTE: A autora (2024).

FIGURA 31 – DADOS DA TABELA LIVRO DIDÁTICO NO BANCO DE DADOS



id_livro	titulo	autor	editora	isbn	edicao	preco
1	Fundamentos de Redes de Computadores	Andrew Tanenbaum	Pearson	9780132126953	5	199.50
2	Algoritmos e Estruturas de Dados	Nikos Markatos	McGraw-Hill	9780070312843	3	150.00
3	Engenharia de Software	Ian Sommerville	Addison-Wesley	9780321313799	9	225.00
4	Inteligência Artificial: Uma Abordagem Moderna	Stuart Russell	Prentice Hall	9780136042594	3	299.90
5	Banco de Dados: Projeto e Implementação	Hector Garcia-Molina	Pearson	9780130312973	2	175.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

FONTE: A autora (2024).

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

O projeto consiste na refatoração de um código-fonte em Java relacionado a algoritmos de ordenação, originalmente disponibilizado no site *Geeks for Geeks* e sugerido pelo professor da disciplina. O objetivo principal é aplicar os princípios do *Clean Code* para aprimorar a legibilidade, clareza, organização e manutenibilidade do código.

Durante o desenvolvimento, diversas melhorias foram implementadas para promover maior consistência, eficiência e compreensão do código. Entre elas, destacaram-se a padronização e renomeação de variáveis, a extração de métodos para favorecer a reutilização lógica, a simplificação do método principal e ajustes no estilo de codificação. A organização geral do código também foi otimizada, assegurando uma distribuição mais clara das responsabilidades e maior uniformidade estrutural.

A adoção das boas práticas de *Clean Code* é fundamental no contexto do desenvolvimento ágil, pois contribui para a criação de soluções mais simples, compreensíveis e adaptáveis.

Para Martin (2009, p.6),

> Escrever um código limpo exige o uso disciplinado de uma miríade de pequenas técnicas aplicadas por meio de uma sensibilidade meticulosamente adquirida sobre “limpeza”. A “sensibilidade ao código” é o segredo. Alguns de nós já nascemos com ela. Outros precisam se esforçar para adquiri-la. Ela não só nos permite perceber se o código é bom ou ruim, como também nos mostra a estratégia e disciplina de como transformar um código ruim em um limpo.

Além disso, a refatoração contínua é essencial para a manutenção da qualidade do software. Conforme ressalta Martin (2009, p. 166), “agora que temos testes, podemos manter nosso código limpo, refatorando gradualmente para aumentar a coesão, diminuir acoplamento e eliminar duplicações.” Essas práticas facilitam a evolução do sistema, reduzem a complexidade e promovem maior eficiência na manutenção e expansão do software.

7.1 ARTEFATOS DO PROJETO

Apresentam-se os principais artefatos desenvolvidos no projeto, ressaltando as melhorias implementadas e a organização do código, com foco na obtenção de maior qualidade e clareza.

7.1.1 Padronização e renomeação de variáveis

As variáveis foram renomeadas e padronizadas para o idioma inglês, visando melhorar a consistência, legibilidade e o alinhamento às boas práticas de desenvolvimento. Essas mudanças facilitam a compreensão e manutenção do código. Exemplos das alterações realizadas incluem:
arr → array, **n → arrayLength**, **temp → temporary**, **trocado → hasSwap**

7.1.2 Extração de método para centralizar a lógica de troca de elementos

A lógica responsável pela troca de elementos foi extraída para um novo método denominado **swapElementPosition()**, promovendo maior clareza, reutilização e facilitando a manutenção do código.

FIGURA 32 – CÓDIGO ORIGINAL

```
static void bubbleSort(int array[], int arrayLength)
{
    int i, j, temporary;
    boolean hasSwap;
    for (i = 0; i < arrayLength - 1; i++) {
        hasSwap = false;
        for (j = 0; j < arrayLength - i - 1; j++) {
            if (array[j] > array[j + 1]) {

                // Swap array[j] and array[j+1]
                temporary = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temporary;
                hasSwap = true;
            }
        }

        // If no two elements were
        // trocado by inner loop, then break
        if (hasSwap == false)
            break;
    }
}
```

FIGURA 33 – CÓDIGO REFATORADO

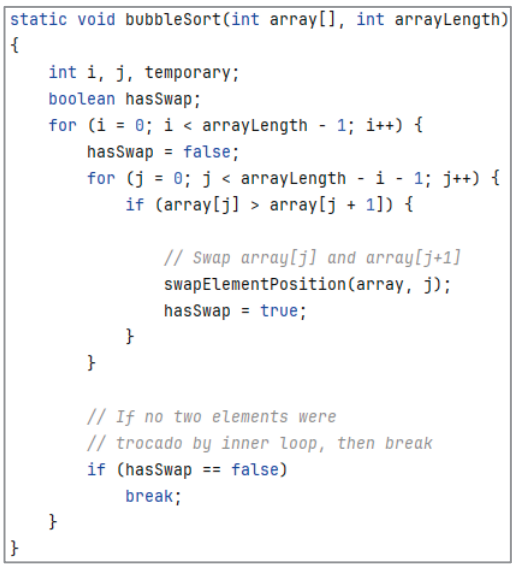
```
static void bubbleSort(int array[], int arrayLength)  no usages
{
    int i, j, temporary;
    boolean hasSwap;
    for (i = 0; i < arrayLength - 1; i++) {
        hasSwap = false;
        for (j = 0; j < arrayLength - i - 1; j++) {
            if (array[j] > array[j + 1]) {

                // Swap array[j] and array[j+1]
                swapElementPosition(array, j);
                hasSwap = true;
            }
        }

        // If no two elements were
        // trocado by inner loop, then break
        if (hasSwap == false)
            break;
    }
}

private static void swapElementPosition(int[] array, int j) {
    int temporary;
    temporary = array[j];
    array[j] = array[j + 1];
    array[j + 1] = temporary;
}
```

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

7.1.3 Refatoração e simplificação do método bubbleSort

O método **bubbleSort** foi aprimorado com diversas melhorias que tornaram o código mais limpo e legível. A variável **temporary** foi realocada para o método **swapElementPosition**, enquanto a variável **hasSwap** passou a ser inicializada diretamente, utilizando a negação **!hasSwap** para simplificar a lógica condicional. Além disso, foi criada a variável auxiliar **lastElementIndex** para evitar repetições e tornar o laço **for** mais claro. As variáveis **i** e **j** foram declaradas diretamente no laço, reduzindo seu escopo e alinhando-se às boas práticas de programação. Por fim, comentários redundantes foram removidos, resultando em um código mais limpo e fácil de entender.

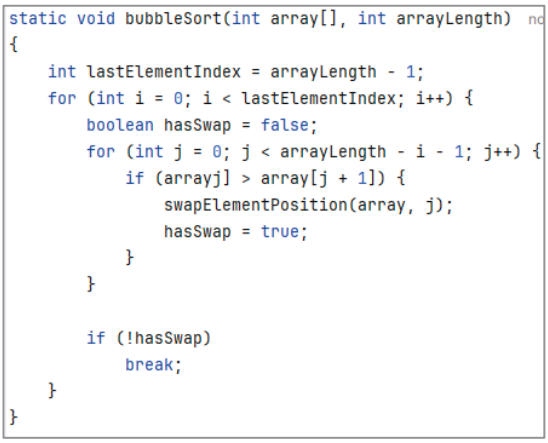
FIGURA 34 – ANTES: VERSÃO ORIGINAL DO MÉTODO BUBBLESORT



```
static void bubbleSort(int array[], int arrayLength)
{
    int i, j, temporary;
    boolean hasSwap;
    for (i = 0; i < arrayLength - 1; i++) {
        hasSwap = false;
        for (j = 0; j < arrayLength - i - 1; j++) {
            if (array[j] > array[j + 1]) {

                // Swap array[j] and array[j+1]
                swapElementPosition(array, j);
                hasSwap = true;
            }
        }

        // If no two elements were
        // trocado by inner loop, then break
        if (hasSwap == false)
            break;
    }
}
```

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

FIGURA 35 – DEPOIS: MÉTODO BUBBLESORT SIMPLIFICADO



```
static void bubbleSort(int array[], int arrayLength)
{
    int lastElementIndex = arrayLength - 1;
    for (int i = 0; i < lastElementIndex; i++) {
        boolean hasSwap = false;
        for (int j = 0; j < arrayLength - i - 1; j++) {
            if (arrayj] > array[j + 1]) {
                swapElementPosition(array, j);
                hasSwap = true;
            }
        }

        if (!hasSwap)
            break;
    }
}
```

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

7.1.4 Centralização e renomeação do método de impressão

A linha **System.out.println("Array ordenado: ");** foi movida do método **main** para dentro do método **printArray**, centralizando a responsabilidade pela exibição do resultado e aprimorando a organização do código. Além disso, o método **printArray** foi renomeado para **showSortedArray**, tornando mais explícito que o conteúdo exibido é o array já ordenado, o que contribui para a legibilidade e clareza do código.

FIGURA 36 – MÉTODO MAIN ORIGINAL

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

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

FIGURA 37 – MÉTODO MAIN MODIFICADO

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

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

FIGURA 38 – MÉTODO PRINTARRAY ORIGINAL

```
static void printArray(int array[], int arrayLength)
{
    int i;
    for (i = 0; i < arrayLength; i++)
        System.out.print(array[i] + " ");
    System.out.println();
}
```

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

FIGURA 39 – MÉTODO SHOWSORTEDARRAY

```
static void showSortedArray(int array[], int arrayLength)
{
    System.out.println("Array ordenado: ");
    for (int i = 0; i < arrayLength; i++)
        System.out.print(array[i] + " ");
    System.out.println();
}
```

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

7.1.5 Padronização e organização do código

Foram implementadas melhorias para garantir uniformidade, clareza e organização do código. A abertura das chaves dos métodos passou a ocorrer na mesma linha da declaração, conforme as convenções recomendadas. A declaração dos arrays foi padronizada seguindo as sugestões da IDE **IntelliJ**. Além disso, o método **main** foi reposicionado para o topo da classe, com o objetivo de melhorar a organização e facilitar a leitura da estrutura geral do programa.

FIGURA 40 – CÓDIGO ANTES DA PADRONIZAÇÃO E REORGANIZAÇÃO

```
class BubbleSort {  no usages
    static void bubbleSort(int array[], int arrayLength)  no usage
    {
        int lastElementIndex = arrayLength - 1;
        for (int i = 0; i < lastElementIndex; i++) {
            boolean hasSwap = false;
            for (int j = 0; j < arrayLength - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    swapElementPosition(array, j);
                    hasSwap = true;
                }
            }

            if (!hasSwap)
                break;
        }
    }

    private static void swapElementPosition(int array[], int j)
    {
        int temporary;
        temporary = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temporary;
    }

    static void showSortedArray(int array[], int arrayLength)  no
    {
        System.out.println("Array ordenado: ");
        for (int i = 0; i < arrayLength; i++)
            System.out.print(array[i] + " ");
        System.out.println();
    }

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

FIGURA 41 – CÓDIGO APÓS PADRONIZAÇÃO DE ESTILO E ESTRUTURA

```
class BubbleSort {  no usages
    public static void main(String[] args) {
        int[] array = { 64, 34, 25, 12, 22, 11, 90 };
        int arrayLength = array.length;
        bubbleSort(array, arrayLength);
        showSortedArray(array, arrayLength);
    }

    static void bubbleSort(int[] array, int arrayLength) {  no usage
        int lastElementIndex = arrayLength - 1;
        for (int i = 0; i < lastElementIndex; i++) {
            boolean hasSwap = false;
            for (int j = 0; j < arrayLength - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    swapElementPosition(array, j);
                    hasSwap = true;
                }
            }

            if (!hasSwap)
                break;
        }
    }

    private static void swapElementPosition(int[] array, int j) {
        int temporary;
        temporary = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temporary;
    }

    static void showSortedArray(int[] array, int arrayLength) {  n
        System.out.println("Array ordenado: ");
        for (int i = 0; i < arrayLength; i++)
            System.out.print(array[i] + " ");
        System.out.println();
    }
}
```

FONTE: A autora (2024), adaptado de Tiwari (2024) via Geeks for Geeks.

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

O projeto de **Desenvolvimento Web 1** consistiu na criação de uma aplicação web com dois módulos independentes de CRUD, voltados para a gestão de Alunos e Cursos, oferecendo funcionalidades completas de listagem, inserção, edição e remoção de registros. Os dados foram armazenados localmente no navegador, utilizando o Local Storage, o que eliminou a necessidade de back-end ou banco de dados externo. A interface da aplicação foi construída com Bootstrap e CSS, utilizando componentes visuais padronizados, ícones e layouts fornecidos pelo framework, o que contribuiu significativamente para a usabilidade e acessibilidade da aplicação, tornando a navegação mais intuitiva.

No projeto da disciplina de **Desenvolvimento Web 2**, a aplicação foi expandida com a inclusão de um terceiro módulo CRUD voltado para a gestão de Matrículas, complementando os módulos de Alunos e Cursos já existentes. Essa versão integrou-se com um back-end desenvolvido em Spring Boot com banco de dados PostgreSQL, proporcionando maior segurança, escalabilidade e persistência dos dados. Cada matrícula relacionava um aluno a um curso em uma data específica, armazenando também a nota final do aluno. As telas de inserção e edição incluíram componentes de seleção para aluno e curso, garantindo integridade dos dados e facilidade de uso. O sistema manteve as operações completas de CRUD, incluindo listagem, inserção, alteração e remoção, além de um menu de navegação entre os módulos.

A adoção de práticas ágeis no desenvolvimento de software é fundamental, pois possibilita entregas rápidas, ajustes constantes e foco nas funcionalidades mais relevantes. Conforme destaca Sommerville (2011, p. 53),

> Métodos ágeis são métodos de desenvolvimento incremental que se concentram em desenvolvimento rápido, releases frequentes do software, redução de overheads dos processos e produção de códigos de alta qualidade.

Dessa forma, a aplicação é construída em etapas, com melhorias progressivas e validações contínuas.

8.1 ARTEFATOS DO PROJETO

A seguir, são apresentados os principais artefatos desenvolvidos ao longo dos projetos. As aplicações abrangem desde funcionalidades básicas com armazenamento local até a integração entre front-end e back-end, com o uso de banco de dados relacional.

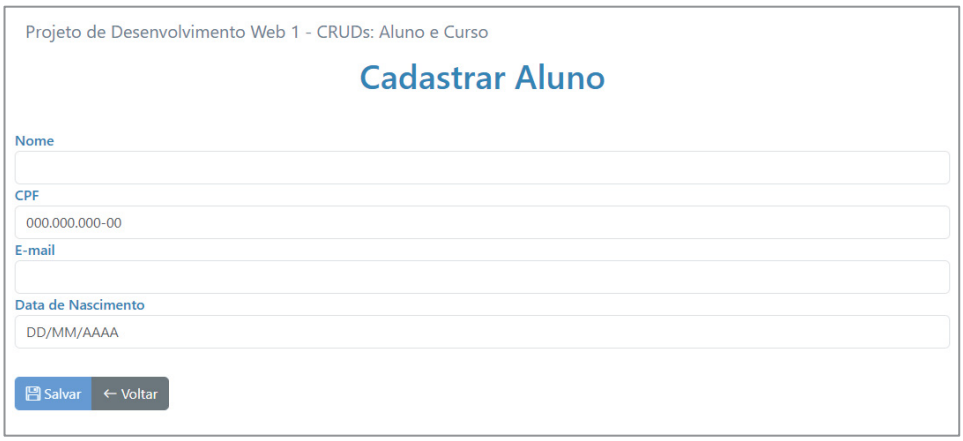
8.1.1 Alunos – Listagem e Cadastro com Local Storage (Projeto 1)

A aplicação foi desenvolvida utilizando Angular, TypeScript, CSS e Bootstrap, contemplando componentes para listagem, cadastro, edição e remoção de alunos. Os dados foram armazenados no navegador por meio de Local Storage. A interface foi construída de forma responsiva, com validações e atualizações automáticas da lista de alunos.

FIGURA 42 – PROJETO 1: LISTA DE ALUNOS

FONTE: A autora (2024).

FIGURA 43 – PROJETO 1: FORMULÁRIO DE CADASTRAR ALUNO



FONTE: A autora (2024).

8.1.2 Cadastro de Matrícula – Spring Boot e PostgreSQL (Projeto 2)

Na segunda etapa, o sistema foi implementado com Spring Boot e banco de dados PostgreSQL integrado por meio de APIs REST. No formulário de cadastro de matrícula, os campos **Aluno** e **Curso** foram configurados para carregar dados diretamente do banco de dados, garantindo registros válidos, consistentes e atualizados por meio do back-end desenvolvido com Spring Boot.

FIGURA 44 – PROJETO 2: FORMULÁRIO DE CADASTRAR MATRÍCULA

FONTE: A autora (2024).

8.1.3 Implementação do menu de navegação

Foi implementado um menu prático e intuitivo, que permite o acesso rápido às principais funcionalidades do sistema, melhorando a navegação e a usabilidade.

FIGURA 45 – PROJETO 2: MENU DE NAVEGAÇÃO

FONTE: A autora (2024).

8.1.4 Estrutura de pastas do Projeto

Para garantir a organização, clareza e facilitar a manutenção, o projeto foi estruturado com uma divisão clara entre front-end e back-end. Cada parte seguiu princípios específicos para melhorar a escalabilidade e o desenvolvimento colaborativo.

8.1.4.1 Front-end (Angular)

No front-end, o projeto foi organizado de forma modular, seguindo o conceito de separação por funcionalidades. Essa abordagem facilita o desenvolvimento, a manutenção e a escalabilidade da aplicação.

8.1.4.2 Back-end (Spring Boot)

No back-end, o projeto foi organizado em camadas que separam claramente as responsabilidades da aplicação, seguindo as boas práticas de arquitetura de software.

FIGURA 46 – FRONT-END (ANGULAR): ORGANIZADO POR MÓDULOS

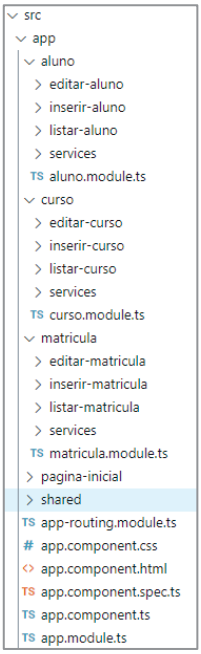


FIGURA 47 – BACK-END (SPRING BOOT): ARQUITETURA EM CAMADAS

FONTE: A autora (2024).

8.1.5 Inicialização e consumo de dados no Angular

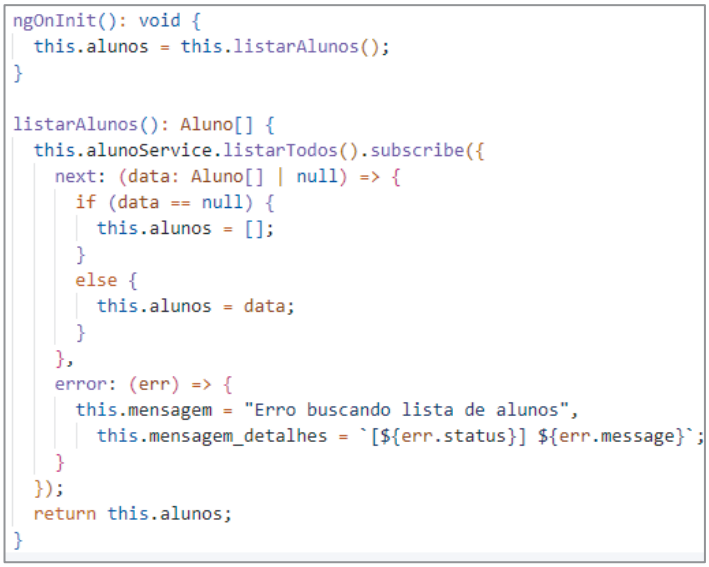
No Projeto 2, a aplicação Angular tem consumido APIs para obter e exibir dados. Os trechos a seguir mostram o método **listarTodos** do **AlunoService**, responsável pela chamada HTTP que busca a lista de alunos, e o carregamento desses dados no componente via método **ngOnInit**, garantindo a atualização da interface assim que o componente é inicializado.

FIGURA 48 – MÉTODO LISTARTODOS NO ALUNOSERVICE

```
listarTodos(): Observable<Aluno[] | null> {
  return this.httpClient.get<Aluno[]>(this.BASE_URL, this.httpOptions).pipe(
    map((resp: HttpResponse<Aluno[]>) => {
      if (resp.status != 200) {
        return [];
      }
      else {
        let lista: Aluno[] = [];
        if (resp.body && resp.body.length > 0) {
          let alu: Aluno = new Aluno();
          resp.body.forEach(a => {
            alu = new Aluno(a.id, a.nome, a.cpf, a.email, a.dataNascimento);
            lista.push(alu);
          })
        }
        return lista;
      }
    }),
    catchError((e, c) => {
      if (e.status == 404) {
        return of([]);
      }
      else {
        return throwError(() => e);
      }
    })
  );
}
```

FONTE: A autora (2024).

FIGURA 49 – MÉTODO NGONINIT DO COMPONENTE, QUE CHAMA LISTARALUNOS



```
ngOnInit(): void {
  this.alunos = this.listarAlunos();
}

listarAlunos(): Aluno[] {
  this.alunoService.listarTodos().subscribe({
    next: (data: Aluno[] | null) => {
      if (data == null) {
        this.alunos = [];
      }
      else {
        this.alunos = data;
      }
    },
    error: (err) => {
      this.mensagem = "Erro buscando lista de alunos",
        this.mensagem_detalhes = `[${err.status}] ${err.message}`;
    }
  });
  return this.alunos;
}
```

FONTE: A autora (2024).

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

O projeto consistiu no desenvolvimento de um site, na apresentação da proposta do produto e na justificativa para sua criação. Incluiu ainda a elaboração de protótipos de interface, decisões de design (como definição de cores, layout, ícones e tipografia) e a coleta de feedback a partir da apresentação das telas a um usuário potencial. Atendendo a esses requisitos, foi desenvolvida uma plataforma para cadastro e organização de jogos, com foco na experiência do usuário. O site foi projetado para facilitar a navegação e a interação, oferecendo uma interface intuitiva e acessível, alinhada às reais necessidades reais do público-alvo.

Os protótipos contemplaram as principais funcionalidades da aplicação e foram construídos com base em princípios de usabilidade e design centrado no usuário. As escolhas visuais foram justificadas por critérios como clareza, legibilidade e harmonia na navegação. Destaca-se que os protótipos foram desenvolvidos diretamente com Angular, TypeScript, Bootstrap e CSS, utilizando Local Storage para armazenamento — sem o uso de ferramentas específicas de prototipação visual — o que permitiu uma prototipação funcional e já integrada ao front-end.

Por fim, os protótipos foram apresentados a um usuário, com o objetivo de avaliar a experiência de uso. O feedback coletado possibilitou identificar pontos fortes e oportunidades de melhoria, reforçando a importância de uma abordagem ágil, iterativa e centrada no usuário.

A aplicação de UX no desenvolvimento de software está fortemente relacionada à valorização do papel ativo do cliente durante o processo. Como destacado por Sommerville (2011, p. 40), “os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.” Esse envolvimento está alinhado aos princípios das metodologias ágeis, que enfatizam a colaboração contínua com o usuário, entregas incrementais e adaptações constantes. Ao incorporar feedbacks frequentes e testes de usabilidade ao longo do desenvolvimento, o processo se torna mais eficiente, reduz o retrabalho e assegura que o produto evolua continuamente, atendendo às necessidades reais dos usuários.

9.1 ARTEFATOS DO PROJETO

A seguir, são apresentados os principais artefatos do projeto, que evidenciam as decisões de *design* adotadas e os recursos implementados para garantir uma experiência do usuário clara, eficiente e intuitiva.

9.1.1 Objetivo e funcionalidades da Aplicação

O *GameCollector* é um site destinado ao cadastro e organização de jogos, permitindo que o usuário registre informações detalhadas, como nome, categoria, console, imagem, código de barras, status, tempo de jogo e descrição. Além disso, o sistema possibilita o cadastro de consoles (como *PlayStation 1*, *PlayStation 2*, *Nintendo*, entre outros), que podem ser vinculados aos jogos cadastrados, incluindo também jogos desejados. O usuário pode movimentar facilmente um jogo da lista de desejados para sua coleção ao adquiri-lo, facilitando o gerenciamento e a organização de sua biblioteca de jogos.

9.1.2 Motivação para o desenvolvimento

Durante a pandemia e o período de isolamento social, houve um aumento significativo na busca por formas de entretenimento em casa, o que resultou em um crescimento expressivo no tempo dedicado aos videogames. Conforme destacado pelo portal Valor Investe (2021, não p.), “mercado de games cresce 140% no Brasil” durante esse período. Para atender a essa demanda crescente, foi desenvolvido o *GameCollector*, uma plataforma que permite aos colecionadores organizar tanto os jogos que já possuem quanto aqueles que desejam adquirir no futuro, facilitando o controle e a gestão de suas coleções de forma prática e acessível.

9.1.3 Protótipos de telas

A seguir, são apresentados protótipos desenvolvidos, incluindo a tela de **edição de jogo desejado**, estruturada em formato de formulário, e a tela de listagem **“Minha Coleção”**, que exibe os jogos já cadastrados pelo usuário.

FIGURA 50 – EDITAR JOGO DESEJADO: FORMULÁRIO (VERSÃO DESKTOP)

FONTE: A autora (2024).

FIGURA 51 – MINHA COLEÇÃO: LISTAGEM DE JOGOS (VERSÃO DESKTOP)

FONTE: A autora (2024).

9.1.3.1 Versão Mobile – responsividade

Para garantir uma boa experiência em diferentes dispositivos, os protótipos também foram adaptados para telas menores, como tablets e smartphones. A interface responsiva permite que os elementos se reorganizem de forma fluida, mantendo a legibilidade, a funcionalidade e a consistência visual do sistema.

A seguir, são apresentadas as versões responsivas das telas, demonstrando como o layout se ajusta automaticamente às resoluções reduzidas, sem comprometer a usabilidade nem a organização das informações. Essa adaptação é essencial para oferecer uma navegação eficiente ao usuário, independentemente do dispositivo utilizado.

FIGURA 52 – EDITAR JOGO DESEJADO: FORMULÁRIO (VERSÃO RESPONSIVA – TABLET)

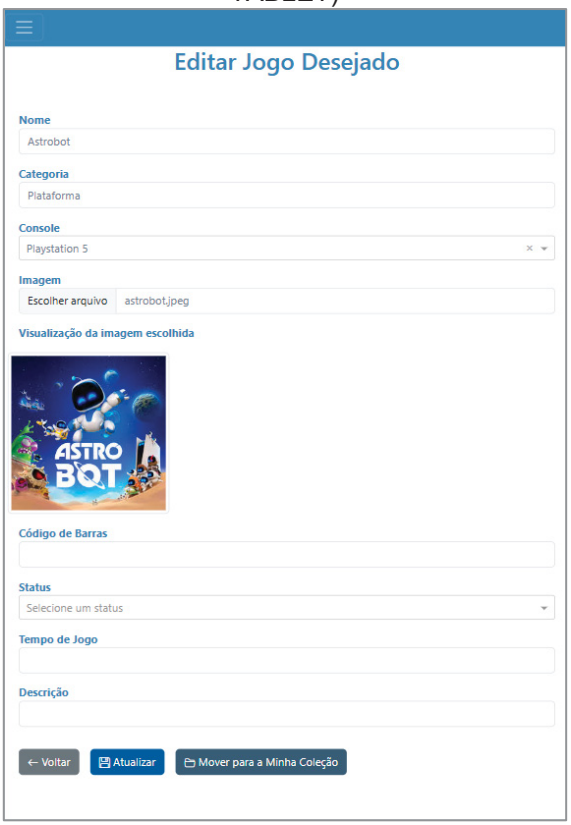


FIGURA 53 – MINHA COLEÇÃO: LISTAGEM DE JOGOS (VERSÃO RESPONSIVA – TABLET)

FONTE: A autora (2024).

9.1.4 Justificativas das escolhas de design

9.1.4.1 Cores

A cor azul foi escolhida por transmitir a sensação de tranquilidade, favorecer a legibilidade e reforçar a credibilidade, além de contribuir para uma interface harmoniosa e agradável. Para complementar, os formulários utilizaram tons de cinza, criando contraste com o azul e reforçando um *design clean*.

9.1.4.2 Layouts

Os *layouts* foram estruturados com espaçamento adequado entre os rótulos e campos, garantindo uma visualização clara e organizada. O *design* também foi pensado para ser responsivo, adaptando-se bem tanto a computadores quanto a dispositivos móveis.

9.1.4.3 Fontes

A tipografia escolhida combinou as fontes Roboto com Helvetica Neue, oferecendo excelente legibilidade e consistência visual. Os títulos foram destacados com tamanho maior e negrito, e os rótulos seguiram o mesmo padrão, facilitando a leitura e navegação.

9.1.4.4 Ícones

Foram utilizados ícones reconhecíveis (como lixeira, lápis, “+”, disquete e seta, além de ícones no menu, favorecendo uma navegação intuitiva e coesa.

9.1.4.5 Feedback ao usuário - Validação e animações

Para facilitar o preenchimento dos formulários, foram aplicadas validações com avisos sobre campos obrigatórios e limite mínimo de caracteres. Também foram adicionadas animações para indicar ações como criação, atualização, exclusão e movimentação de jogos. Um spinner foi incluído nos botões de salvar, atualizar, informando que a ação está em andamento.

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

O projeto teve como objetivo a criação e configuração de um ambiente de desenvolvimento integrado utilizando containers Docker, com base em uma imagem disponibilizada pelo professor. Essa imagem serviu como ponto de partida para a implantação de serviços essenciais ao ciclo de vida de software.

Durante o processo, foi criado um container nomeado com a matrícula da aluna e, manualmente, foram publicadas as portas 22 (SSH), 80 (HTTP), 443 (HTTPS), e 9091 (Jenkins), permitindo o acesso externo aos serviços configurados no ambiente. Essa etapa garantiu a comunicação entre o container e outras interfaces, como navegadores e ferramentas locais.

A configuração do ambiente proporcionou uma base padronizável e replicável, facilitando a automação de tarefas como versionamento de código, execução de builds e integração contínua.

Todo o processo foi registrado por meio de capturas de tela que evidenciaram o funcionamento do container, o acesso ao GitLab via navegador e a realização das operações de commit e push no repositório.

Essa infraestrutura é essencial no desenvolvimento ágil de software, pois possibilita entregas mais rápidas, seguras e alinhadas às boas práticas de DevOps. Para Humble, Molesky e O'Reilly (2015, p. 238-239),

> Quaisquer atividades manuais que são repetidas deveriam ser consideradas potenciais desperdícios e, dessa forma, candidatas para simplificação e automação. Isso inclui: build — criar pacotes da fonte em um único passo; provisionamento — criar ambientes de teste automatizados; deploy — implementar pacotes automaticamente; e teste — executar suítes de teste automatizadas, todos armazenados no controle de versão para garantir repetibilidade, auditabilidade e segurança no processo.

Dessa maneira, a configuração do ambiente garante que as operações essenciais ocorram de forma contínua e sem interferência manual, aumentando a estabilidade do sistema.

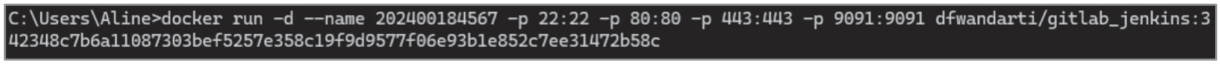
10.1 ARTEFATOS DO PROJETO

A seguir, apresentam-se os principais artefatos que comprovam a configuração e a utilização do ambiente de infraestrutura baseado em Docker, GitLab e Jenkins. Cada etapa do processo foi registrada por meio de capturas de tela, evidenciando o funcionamento dos serviços, a integração com o controle de versão e a execução do *container*.

10.1.1 Publicação das portas no container

O container foi nomeado com o número de matrícula da aluna. Durante sua criação, foram publicadas as portas 22 (SSH), 80 (HTTP), 443 (HTTPS) e 9091 (Jenkins), possibilitando o acesso externo aos serviços configurados.

FIGURA 54 – CONFIGURAÇÃO DAS PORTAS 22, 80, 443 E 9091



FONTE: A autora (2025).

10.1.2 Container em execução

O container foi iniciado com sucesso, utilizando a imagem Docker **dfwandarti/gitlab_jenkins:3**. As figuras demonstram que o container está em execução e que as portas do GitLab e Jenkins foram corretamente mapeadas, garantindo o acesso às aplicações.

FIGURA 55 – SAÍDA DO COMANDO DOCKER PS (PARTE 1)

FONTE: A autora (2025).

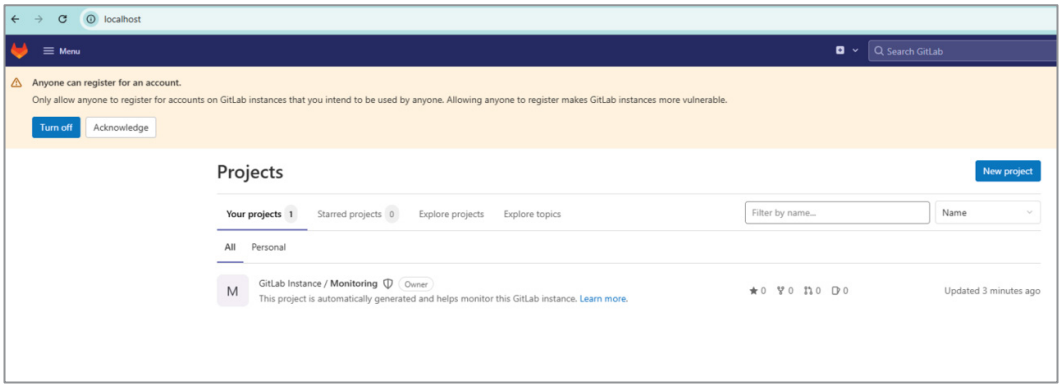
FIGURA 56 – SAÍDA DO COMANDO DOCKER PS (PARTE 2)

FONTE: A autora (2025).

10.1.3 Acesso ao GitLab pelo navegador

Após a inicialização do container, o aesso à interface do GitLab foi realizado com sucesso por meio do navegador, confirmando o correto funcionamento do ambiente configurado.

FIGURA 57 – TELA INICIAL DO GITLAB ACESSADA VIA NAVEGADOR APÓS LOGIN BEM-SUCEDIDO



FONTE: A autora (2025).

10.1.4 Commit e Push no repositório

Foi criado um arquivo README.md para documentar o projeto. Esse arquivo foi adicionado à área de preparação com o comando **git add**, e as alterações foram registradas no histórico local usando o **git commit -m** “Adicionando arquivo README.md”. Por fim, as modificações foram enviadas para o repositório remoto no GitLab com o comando **git push origin main**, atualizando com sucesso a branch principal.

FIGURA 58 – COMANDOS DE COMMIT E PUSH REALIZADOS PARA ATUALIZAR O REPOSITÓRIO REMOTO

```
C:\Users\Aline\Monitoring>echo "# Meu Trabalho de Infra - UFPR" > README.md

C:\Users\Aline\Monitoring>git add README.md

C:\Users\Aline\Monitoring>git add .

C:\Users\Aline\Monitoring>git commit -m "Adicionando arquivo README.md"
[main (root-commit) 6eeaee8] Adicionando arquivo README.md
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

C:\Users\Aline\Monitoring>git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 271 bytes | 271.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To http://localhost/gitlab-instance-0eb99cfd/Monitoring.git
 * [new branch]      main -> main
```

FONTE: A autora (2025).

10.1.5 Histórico de commits (git log)

O comando **git log** foi utilizado para visualizar detalhadamente o histórico de commits realizados no repositório, o que é fundamental para rastrear as modificações feitas ao longo do tempo, facilitar a compreensão da evolução do código e identificar contribuições específicas.

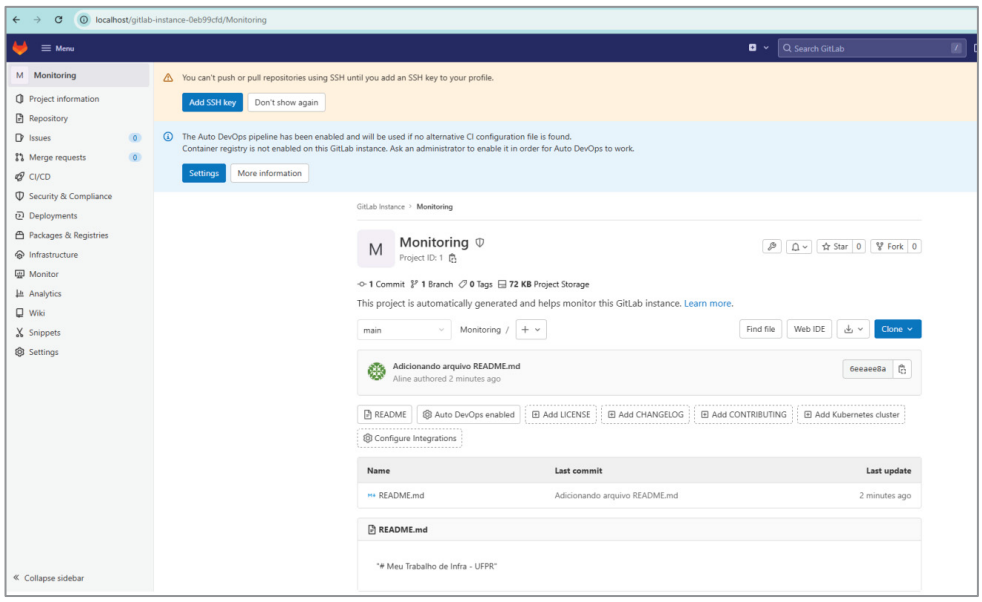
FIGURA 59 – SAÍDA DO COMANDO GIT LOG EXIBINDO OS COMMITS REALIZADOS LOCALMENTE

FONTE: A autora (2025).

10.1.6 Commit visível no GitLab

No repositório remoto do GitLab, foi possível visualizar o commit realizado, confirmando a integração bem-sucedida entre o ambiente local e o servidor. A interface do GitLab exibe o commit '**Adicionando arquivo README.md**', evidenciando a atualização do arquivo no diretório principal do projeto.

FIGURA 60 – VISUALIZAÇÃO DO COMMIT CONFIRMADO NO REPOSITÓRIO REMOTO DO GITLAB



FONTE: A autora (2025).

11 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

O projeto consistiu no desenvolvimento de um script automatizado para interagir com o site https://pt.anotepad.com, simulando a criação de uma nota online. A automação preencheu o título da nota com o título "**Entrega trabalho TEST DAS 2024**" e inseriu no corpo o nome completo e a matrícula da aluna responsável. Essa automação simulou o comportamento de um usuário ao acessar a página, garantindo o preenchimento correto e ágil dos campos, eliminando a necessidade de entrada manual.

Além da criação da nota, o código incluiu mecanismos de tratamento de erro para detectar possíveis falhas no carregamento da página, aumentando a robustez da automação. O fluxo contemplou a abertura do navegador, navegação ao site, preenchimento automático dos campos e uma pausa para a visualização do conteúdo antes do término da execução.

Esse projeto exemplificou como testes automatizados podem ser aplicados em tarefas simples, garantindo precisão e repetibilidade. A documentação do processo incluiu capturas de tela que comprovaram cada etapa de execução, desde o acesso ao site até a inserção dos dados e o resultado final.

No desenvolvimento ágil, a automação de testes é fundamental para validar funcionalidades rapidamente, detectar falhas precocemente garantir entregas contínuas com maior qualidade e segurança. Ela promove a eficiência e a qualidade do software.

Para Sommerville (2011, p.48),

> Automação de testes é essencial para o desenvolvimento *test-first*. Os testes são escritos como componentes executáveis antes que a tarefa seja implementada. Esses componentes de teste devem ser autônomos, devem simular a submissão de entrada a ser testada e devem verificar se o resultado atende à especificação de saída. Um *framework* de testes automatizados é um sistema que torna mais fácil escrever os testes executáveis e submeter um conjunto de testes para execução.

Assim, a automação assegura maior precisão, repetibilidade e confiança no desenvolvimento, viabilizando entregas rápidas e seguras em ambientes ágeis.

11.1 ARTEFATOS DO PROJETO

A seguir, são apresentados os artefatos desenvolvidos no projeto de automação, detalhando as tecnologias utilizadas e as etapas realizadas pelo script durante a simulação de uso do site Anotepad.

11.1.1 Tecnologias utilizadas: Java e Playwright

O projeto foi implementado com a linguagem Java, utilizando a ferramenta Playwright para automação de navegadores web. O Playwright possibilita a simulação de interações do usuário, como navegação, preenchimento de formulários e captura de dados, garantindo testes precisos e repetíveis.

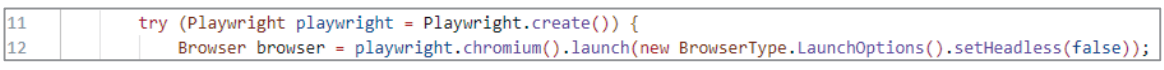
11.1.2 Visão geral da automação

A automação simulou a interação de um usuário real com o site **Anotepad**. O script acessou a página utilizando o navegador Chromium em modo visível, permitindo acompanhar todas as ações. Em seguida, preencheu automaticamente os campos de título e corpo da nota, exibindo o resultado diretamente na interface do navegador.

11.1.3 Inicialização do Playwright e navegador

O script iniciou a biblioteca Playwright e configurou uma instância do navegador Chromium. Essa inicialização prepara o ambiente para a execução das etapas subsequentes da automação, garantindo que o navegador esteja pronto para interagir com o site alvo.

FIGURA 61 – INICIALIZAÇÃO DO PLAYWRIGHT E ABERTURA DO NAVEGADOR CHROMIUM EM MODO VISÍVEL



FONTE: A autora (2025).

11.1.4 Navegação até o site

Após a inicialização, o script abriu uma nova aba e realizou a navegação até o site do Anotepad.

FIGURA 62 – ABERTURA DE NOVA ABA E NAVEGAÇÃO AO SITE COM O TEMPO LIMITE DE 60 SEGUNDOS

```
13          Page page = browser.newPage();
14          page.navigate("https://pt.anotepad.com", new Page.NavigateOptions().setTimeout(60000));
```

FONTE: A autora (2025).

11.1.5 Preenchimento da nota

Em seguida, o script preencheu automaticamente o campo de título da nota e inseriu, no corpo do texto, informações específicas, como o nome e a matrícula da aluna. Essa automatização garante precisão e agilidade no registro de dados, eliminando a necessidade de entrada manual e reduzindo possibilidade de erros.

FIGURA 63 – PREENCHIMENTO AUTOMÁTICO DO TÍTULO DA NOTA E DO CORPO COM O NOME E MATRÍCULA DA ALUNA

```
15          page.fill("#edit_title", "Entrega trabalho TEST DAS 2024");
16          page.fill("#edit_textarea", "Aline Rodrigues de Lima / 202400184567");
```

FONTE: A autora (2025).

11.1.6 Pausa para a visualização

O script realizou uma breve pausa após o preenchimento da nota, permitindo que o usuário visualizassee o conteúdo antes da finalização do processo.

FIGURA 64 – ESPERA DE 6 SEGUNDOS PARA A VISUALIZAÇÃO DA NOTA PREENCHIDA

```
17          page.waitForTimeout(6000);
```

FONTE: A autora (2025).

11.1.7 Tratamento de erro

Para garantir a robustez da automação, o script incluiu um tratamento de erro que detecta quando a página demora a carregar além do esperado. Nessa situação, o erro é capturado e uma mensagem informativa é exibida no console, permitindo que o usuário identifique o problema e tome as medidas adequadas.

FIGURA 65 – TRATAMENTO DE ERRO POR ATRASO NO CARREGAMENTO COM MENSAGEM NO CONSOLE

```
18          } catch (TimeoutError e) {
19              System.out.println("Falha ao carregar a pagina. Verifique sua conexao com a internet");
20          }
```

FONTE: A autora (2025).

11.1.8 Código completo

A seguir, apresenta-se o código completo da automação para criação de uma nota no site Anotepad. O script abrangeu todas as etapas do processo, desde a inicialização do Playwright, navegação até o site, preenchimento dos campos necessários, até o tratamento de possíveis erros durante a execução, garantindo robustez e confiabilidade à automação.

FIGURA 66 – CÓDIGO COMPLETO DA AUTOMAÇÃO PARA CRIAÇÃO DE NOTA NO ANOTEPAD, DETALHANDO O FLUXO COMPLETO DA EXECUÇÃO

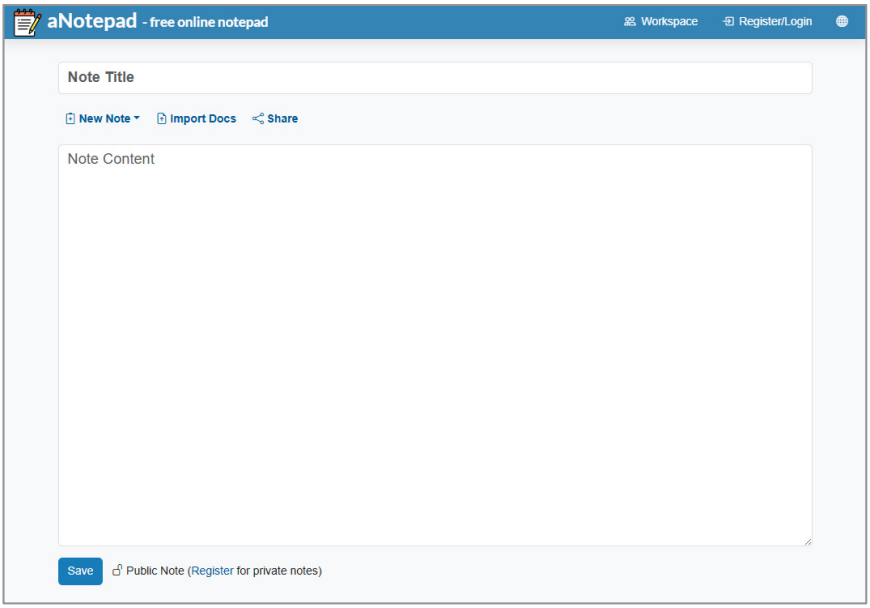
```
1   package org.example;
2
3   import com.microsoft.playwright.Browser;
4   import com.microsoft.playwright.BrowserType;
5   import com.microsoft.playwright.Page;
6   import com.microsoft.playwright.Playwright;
7   import com.microsoft.playwright.*;
8
9   public class Main {
10      public static void main(String[] args) {
11          try (Playwright playwright = Playwright.create()) {
12              Browser browser = playwright.chromium().launch(new BrowserType.LaunchOptions().setHeadless(false));
13              Page page = browser.newPage();
14              page.navigate("https://pt.anotepad.com", new Page.NavigateOptions().setTimeout(60000));
15              page.fill("#edit_title", "Entrega trabalho TEST DAS 2024");
16              page.fill("#edit_textarea", "Aline Rodrigues de Lima / 202400184567");
17              page.waitForTimeout(6000);
18          } catch (TimeoutError e) {
19              System.out.println("Falha ao carregar a pagina. Verifique sua conexao com a internet");
20          }
21      }
22  }
```

FONTE: A autora (2025).

11.1.9 Capturas de tela da automação

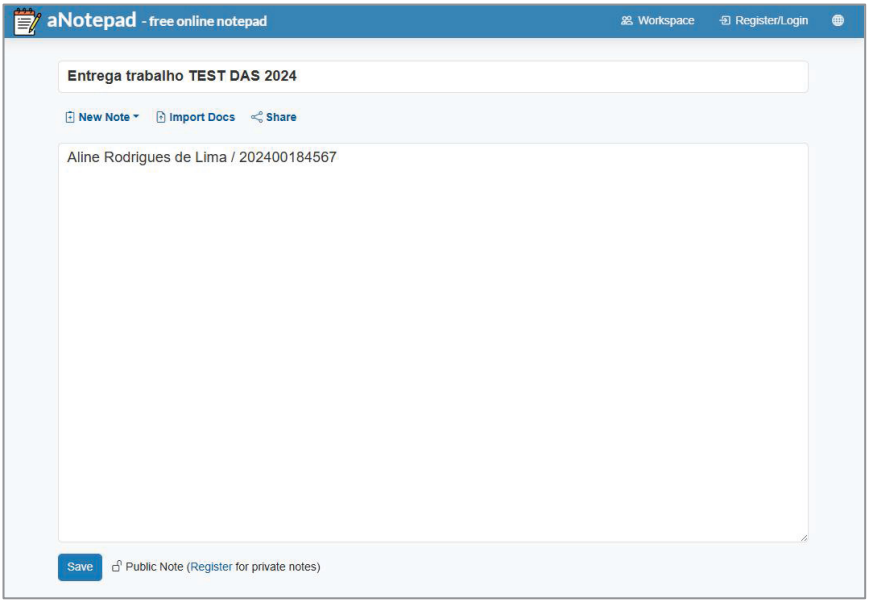
A seguir, são apresentadas imagens que ilustram o funcionamento da automação desenvolvida. As capturas mostram desde o acesso inicial ao site até o preenchimento automático dos campos e o resultado final após a execução completa do script.

FIGURA 67 – PÁGINA INICIAL DO ANOTEPAD ANTES DA EXECUÇÃO DA AUTOMAÇÃO



FONTE: A autora (2025).

FIGURA 68 – CAMPOS DA NOTA PREENCHIDOS AUTOMATICAMENTE



FONTE: A autora (2025).

12 CONCLUSÃO

Este memorial apresentou a integração e aplicação dos princípios ágeis em todo o ciclo de desenvolvimento de software, abrangendo desde a modelagem até a construção e implantação de sistemas. Por meio dos diversos projetos desenvolvidos, foi possível vivenciar na prática e entender melhor conceitos fundamentais como planejamento iterativo, desenvolvimento orientado a testes unitários, refatoração baseada nos princípios de *Clean Code*, modelagem ágil para alinhamento claro dos requisitos, design de interfaces focado na experiência do usuário e a aplicação de testes automatizados para garantir qualidade e produtividade, entre outros.

Contudo, alguns desafios se destacaram durante essa trajetória, especialmente ao implementar um back-end orientado a testes com casos pré-definidos, o que exigiu precisão e aderência rigorosa aos requisitos. Também foi desafiador gerenciar o fluxo de trabalho ágil com Kanban, garantindo que as tarefas avançassem para cumprir prazos de forma contínua e eficiente. Além disso, a disciplina UX trouxe uma importante lição: muitas vezes, como desenvolvedores, não percebemos completamente as necessidades e expectativas reais dos usuários, o que reforça a importância de um design centrado no usuário e da validação constante para garantir uma boa experiência.

Esses aprendizados evidenciaram que o desenvolvimento ágil vai muito além da técnica, exigindo organização, comunicação eficaz e foco no usuário. A colaboração constante, o feedback rápido e a capacidade de adaptação contínua são essenciais para garantir a entrega de software funcional, de qualidade e sustentável. Os projetos realizados demonstraram como a aplicação desses princípios promove um processo eficiente, alinhado às necessidades reais do cliente e dos usuários finais.

13 REFERÊNCIAS

AMBLER, Scott W. **Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process**. 1. ed. New York: Wiley, 2002.

BECK, Kent et al. Manifesto Ágil: princípios por trás do manifesto para desenvolvimento ágil de software. Agile Alliance, 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/principles.html. Acesso em: 18 jul. 2025.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. 4. ed. Porto Alegre: Sagra Luzzatto, 2004.

HIRAMA, Kechi. **Engenharia de software: qualidade e produtividade com tecnologia**. São Paulo: Pearson Prentice Hall, 2011.

HUMBLE, Jez; MOLESKY, Joanne; O'REILLY, Barry. **Lean Enterprise: como empresas de alta performance inovam em escala**. São Paulo: Casa do Código, 2015.

KANBANBOARDGAME.COM. Software Development Kanban Game. Disponível em: https://kanbanboardgame.com. Acesso em: 02 jul. 2025.

KANBAN UNIVERSITY. O guia oficial do método Kanban. Versão 1 – fev. 2021. Kanban University, 2021. Disponível em: https://kanban.university/wp-content/uploads/2021/04/The-Official-Kanban-Guide_Portuguese_A4.pdf. Acesso em: 27 jul. 2025.

MARTIN, Robert C. **Código limpo: Habilidades Práticas do Agile Software**. São Paulo: Alta Books, 2009.

MARTIN, Robert C. **Desenvolvimento Ágil Limpo: de volta às origens**. Rio de Janeiro: Alta Books, 2020.

SCRUM GUIDES. Scrum Guide (Guia do Scrum). Versão 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-European.pdf. Acesso em: 26 jul. 2025.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

VALORINVESTE. Com pandemia, mercado de games cresce 140% no Brasil, aponta estudo. Valor Investe, São Paulo, 23 jan. 2021. Disponível em: https://valorinveste.globo.com/objetivo/gastar-bem/noticia/2021/01/23/com-pandemia-mercado-de-games-cresce-140percent-no-brasil-aponta-estudo.ghtml. Acesso em: 20 jun. 2025.