

UNIVERSIDADE FEDERAL DO PARANÁ

LUCAS NORBERTO BUENO DOS SANTOS

MEMORIAL DE PROJETOS: INTEGRAÇÃO INCREMENTAL ORIENTADA A
EVIDÊNCIAS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

CURITIBA

2025

LUCAS NORBERTO BUENO DOS SANTOS

MEMORIAL DE PROJETOS: INTEGRAÇÃO INCREMENTAL ORIENTADA A
EVIDÊNCIAS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

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.

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

CURITIBA

2025

TERMO DE APROVAÇÃO

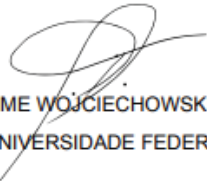
Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Desenvolvimento Ágil de Software da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **LUCAS NORBERTO BUENO DOS SANTOS**, intitulada: **MEMORIAL DE PROJETOS: INTEGRAÇÃO INCREMENTAL ORIENTADA A EVIDÊNCIAS NO DESENVOLVIMENTO ÁGIL DE SOFTWARE**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 12 de Setembro de 2025.



RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora



JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial apresenta, em formato de parecer, a trajetória acadêmico-profissional construída ao longo do curso e os resultados dos projetos desenvolvidos. O objetivo é evidenciar como os artefatos produzidos - trechos de código, capturas de execução, *scripts*, modelos e referências - materializam práticas de desenvolvimento ágil: entregas incrementais, feedback rápido, rastreabilidade e reprodutibilidade. A integração dos conteúdos é demonstrada na progressão de fundamentos de programação e design de código para modelagem e comunicação técnica, seguida de planejamento e priorização orientados a valor, construção de interfaces e consumo de dados, desenvolvimento mobile que avança de UI e navegação para integração a serviços REST com corrotinas, infraestrutura com execução de serviços containerizados e testes com automação de verificações objetivas. Em conjunto, os projetos mostram aprendizado progressivo, foco no valor entregue e capacidade de adaptação, compondo um portfólio coerente com os princípios do desenvolvimento ágil.

Palavras-chave: Desenvolvimento Ágil; Integração de Disciplinas; Portfólio de Projetos; Experiência Profissional.

ABSTRACT

This memorial, presented as an evaluative report, outlines the academic–professional trajectory built throughout the program and the results of the projects developed. Its aim is to show how the produced artifacts - code snippets, execution screenshots, scripts, models, and references - materialize agile *software* practices: incremental delivery, rapid feedback, traceability, and reproducibility. Integration of the contents is demonstrated in the progression from programming fundamentals and code design to modeling and technical communication; then to value-oriented planning and prioritization; interface construction and data consumption; mobile development that advances from UI and navigation to integrating REST services with coroutines; infrastructure through the execution of containerized services; and testing with automated, objective checks. Taken together, the projects show progressive learning, a focus on delivered value, and adaptability, forming a portfolio consistent with the principles of agile development.

Keywords: Agile Development; Discipline Integration; Project Portfolio; Professional Experience.

SUMÁRIO

1 PARECER TÉCNICO.....	7
2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE.....	10
2.1 ARTEFATOS DO PROJETO.....	11
3 DISCIPLINA: MAG1 E MAG2 – MODELAGEM ÁGIL DE SOFTWARE 1 E 2.....	27
3.1 ARTEFATOS DO PROJETO - MAG1.....	28
3.2 ARTEFATOS DO PROJETO - MAG2.....	31
4 DISCIPLINA: GAP1 E GAP2 – GERENCIAMENTO ÁGIL DE PROJETOS DE SOFTWARE 1 E 2.....	35
4.1 ARTEFATOS DO PROJETO – GAP1.....	36
4.2 ARTEFATOS DO PROJETO – GAP2.....	36
5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO.....	38
5.1 ARTEFATOS DO PROJETO.....	39
6 DISCIPLINA: BD – BANCO DE DADOS.....	43
6.1 ARTEFATOS DO PROJETO.....	44
7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO.....	56
7.1 ARTEFATOS DO PROJETO.....	57
8 DISCIPLINA: WEB1 E WEB2 – DESENVOLVIMENTO WEB 1 E 2.....	59
8.1 ARTEFATOS DO PROJETO - WEB1.....	60
8.2 ARTEFATOS DO PROJETO - WEB2.....	62
9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE.....	66
9.1 ARTEFATOS DO PROJETO.....	67
10 DISCIPLINA: MOB1 E MOB2 – DESENVOLVIMENTO MOBILE 1 E 2.....	71
10.1 ARTEFATOS DO PROJETO - MOB1.....	72
10.2 ARTEFATOS DO PROJETO - MOB2.....	73
11 DISCIPLINA: INFRA - INFRAESTRUTURA PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SOFTWARE (DEVOPS).....	75
11.1 ARTEFATOS DO PROJETO.....	76
12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS.....	78
12.1 ARTEFATOS DO PROJETO.....	78
13 CONCLUSÃO.....	81
REFERÊNCIAS.....	82

1 PARECER TÉCNICO

Este memorial reúne e organiza, sob a ótica da integração de práticas e artefatos, os projetos realizados ao longo das disciplinas do curso, explicitando de que maneira cada entrega contribuiu para o desenvolvimento ágil de *software* e onde cada projeto e suas evidências se encontram no documento. O objetivo deste parecer é integrar os resultados em uma narrativa coerente, evidenciando incrementos, *feedback* rápido, rastreabilidade e reprodutibilidade - pilares que, combinados, sustentam a criação de *software* com foco em valor e melhoria contínua (Beck, 2000; Schwaber; Sutherland, 2020).

A base foi lançada nas disciplinas Métodos Ágeis Para Desenvolvimento de Software (MADS), Modelagem Ágil de Software 1 e 2 (MAG1 e MAG2), com a consolidação de princípios e práticas de métodos ágeis (priorização, critérios de aceitação e ciclos curtos) e o uso da modelagem como instrumento de comunicação e redução de ambiguidade. Esse alicerce metodológico e de modelagem deu coesão às entregas posteriores, viabilizando que as práticas ágeis não ficassem apenas no plano conceitual, mas fossem efetivamente operacionalizadas em artefatos e código legível, testável e evolutivo (Martin, 2008).

Nas disciplinas MAG, a ênfase recaiu na modelagem como instrumento de comunicação e redução de ambiguidade, alinhando expectativas entre solução e domínio. Os diagramas e descrições funcionam como artefatos de descoberta e de verificação: antecipam riscos, expõem suposições e criam um canal semântico com o restante do time em um contexto profissional. No contexto ágil, essa documentação enxuta é menos um registro estático e mais um artefato de conversação que evolui junto com o produto (Beck, 2000).

As disciplinas MADS, Gerenciamento Ágil de Projetos de Software 1 e 2 (GAP1 e GAP2) estruturaram o como entregar: priorização, critérios de aceitação, limites de *Work In Progress* (WIP) e cadência de inspeção/adaptação. Seus capítulos reúnem planos, critérios e artefatos de acompanhamento, ancorando as demais entregas em uma lógica de incrementos pequenos e verificáveis (Schwaber; Sutherland, 2020). Neste contexto, “feito” significa “atende ao critério e há evidência” - um padrão que se repete em Desenvolvimento WEB1 e WEB2, Mobile, Infraestrutura Para Desenvolvimento e Implantação de Software (INFRA) e Testes Automatizados (TEST).

Introdução À Programação (INTRO) e Aspectos Ágeis de Programação (AAP), por sua vez, reforçaram a operacionalização técnica dessa base, lógica, estruturação de código e tratamento de erros deram lastro à implementação de critérios de aceitação em código e à construção de artefatos legíveis e evolutivos. Esses fundamentos reaparecem nas trilhas subsequentes (WEB, Mobile e TEST), sustentando a clareza de fluxo, a organização das rotinas e a verificação objetiva dos resultados (Martin, 2008).

Em Banco de Dados (BD), a passagem do modelo conceitual/lógico para scripts e consultas evidencia a conexão entre domínio e persistência. Os artefatos demonstram como decisões de modelagem informam a estrutura dos dados consumidos pelas aplicações. Essa camada serve de referência para as disciplinas de desenvolvimento, mantendo coerência entre entidades e funcionalidades.

Nas trilhas de WEB1, WEB2 e *User Experience* (UX) No Desenvolvimento Ágil de Software, o foco foi interface, interação e consumo de dados. Suas seções mostram como escolhas de *layout*, microinterações e mensagens de *feedback* sustentam a compreensão e a fluidez de tarefas. O tratamento de estados (sucesso/erro) e a clareza de rótulos e navegação dialogam com heurísticas consolidadas de usabilidade (Nielsen; Molich, 1990), além de reforçar a noção de qualidade incorporada desde a interface.

O eixo Mobile progride de forma incremental e verificável. Em Desenvolvimento Mobile 1 (MOB1), o aplicativo FinApp consolidou a *User Interface* (UI), navegação entre *activities* e listas em memória, com escopo deliberadamente reduzido para treinar fluxo e legibilidade. Em Desenvolvimento Mobile 2 (MOB2), o incremento foi a integração a um serviço REST usando Retrofit/Gson e corrotinas, mantendo a UI responsiva e aplicando filtro local quando necessário. Essa evolução, detalhada no Cap. 10, caracteriza o princípio ágil de crescer a solução por fatias verticais, do *layout* ao dado remoto, sempre com evidência de funcionamento ao final de cada ciclo (Beck, 2000).

Em INFRA, a execução de um serviço containerizado validou a capacidade de executar ambiente local reproduzível com publicação de portas, acesso ao serviço e registro de evidências. O uso de contêiner reforça reprodutibilidade e *onboarding* rápido, fundamentais para integração contínua e para a automação de fluxos futuros (Fowler, 2006; Humble; Farley, 2010).

Em TEST, a automação que cria uma nota no *aNotepad* materializa um critério de aceitação objetivo (título e corpo exatos) e registra evidência reproduzível (artefato e captura de tela). Mesmo simples, o exercício traduz o espírito ágil de verificações automatizadas que encurtam o *feedback loop* e diminuem ambiguidades no que significa “pronto”.

Como conjunto, o documento mostra uma trajetória de integração incremental orientada a evidências. Cada disciplina aporta um artefato que não é isolado, mas encaixa no restante - modelos que dialogam com código, *scripts* que refletem decisões de domínio, automações que verificam critérios de aceitação, ambientes que reproduzem condições de execução. Essa coerência prática (artefato → execução → evidência) demonstra rastreabilidade das decisões, redução de risco por incrementos, e capacidade de adaptação informada por dados - algo central tanto em *Extreme Programming* (XP) quanto nas recomendações clássicas de fluxo contínuo e *deployment* frequente (Beck, 2000; Humble; Farley, 2010).

Este parecer conclui que os projetos, do ponto de vista técnico e processual, caracterizam o desenvolvimento ágil de *software* na prática, sendo, pequenos lotes, resultados verificáveis, ambientes reproduzíveis, e decisões suportadas por artefatos que contam a história do produto.

2 DISCIPLINA: MADS – MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

O componente MADS centrou-se em fundamentos e práticas de métodos ágeis, com ênfase em priorização orientada a valor, critérios de aceitação, limites de WIP, cadências de inspeção e adaptação e métricas de fluxo. As atividades foram conduzidas por meio de estudos dirigidos, exercícios conceituais e análise de casos, formando o alicerce para a organização do trabalho e para a leitura crítica dos artefatos apresentados nos capítulos seguintes.

Este trabalho foi orquestrado usando *Scrum* como estrutura predominante, com práticas como *backlog* priorizado, *Sprint Planning*, *Daily*, *Reviews* e *Retrospectives*. O *Kanban* foi usado para limitar o Work in Progress (WIP). A camada técnica foi baseada em práticas de XP e princípios do *Lean Software Development*. Todas as teorias e *frameworks* explorados na disciplina.

O conhecimento adquirido em MADS serviu de base conceitual para a integração dos conteúdos das demais disciplinas, conforme descrito a seguir: em MAG1 e MAG2, os conceitos de histórias de usuário e *backlog*, estudados em MADS, orientaram a transformação dos modelos UML em requisitos priorizados; em UX, princípios de inspeção e adaptação embasaram a validação dos wireframes antes de sua implementação em WEB1 e WEB2; em BD, as boas práticas de controle de versão contínuo, discutidas em MADS, respaldaram a inclusão de *scripts* SQL no repositório *DevOps* configurado em INFRA; em TEST, métricas de cobertura, obtidas posteriormente com *Playwright*, foram interpretadas à luz dos indicadores de qualidade apresentados na disciplina; e, em GAP1 e GAP2, o conceito de velocidade de equipe, explorado teoricamente em MADS, fundamentou o plano de *releases* elaborado nas primeiras iterações dos projetos.

O estudo comparativo das metodologias ágeis permitiu identificar três lições principais: flexibilidade versus previsibilidade, pois a análise conceitual evidenciou que *frameworks* ágeis oferecem maior capacidade de adaptação a mudanças do que modelos sequenciais, desde que haja disciplina na inspeção e transparência de informações para não perder o controle; integração de práticas, uma vez que *Scrum*, XP, *Kanban* e *Lean* compartilham princípios (*feedback* rápido, foco em valor, melhoria contínua) e podem ser combinados de forma complementar, sendo decisivo

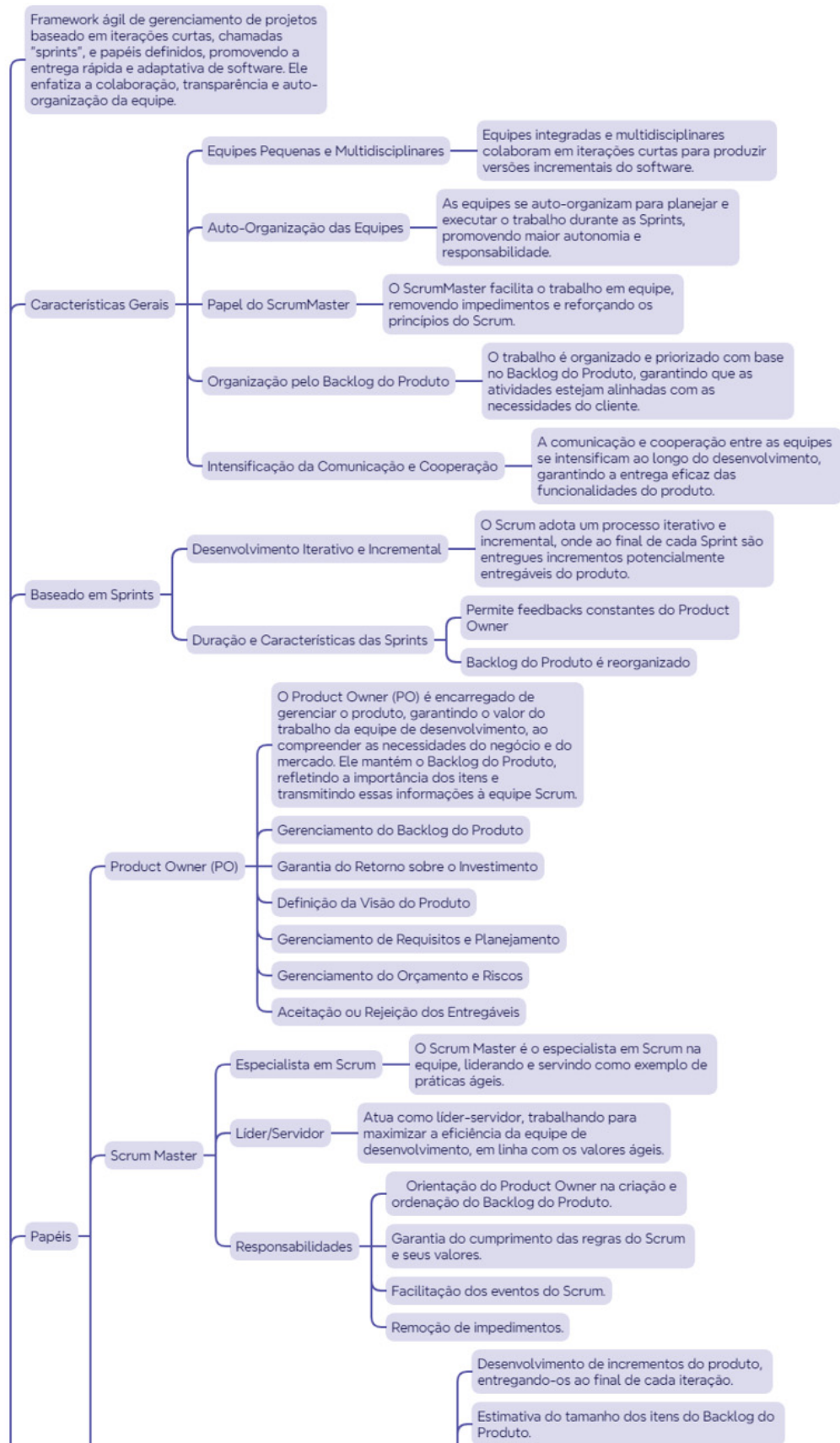
compreender suas interseções para selecionar, futuramente, as práticas mais adequadas a cada contexto de projeto; e importância da cultura de equipe, pois as discussões em sala reforçaram que a adoção bem-sucedida de métodos ágeis depende mais de postura colaborativa, comunicação e comprometimento do que da escolha de artefatos específicos.

Essas reflexões fundamentam a aplicação prática que será realizada nos componentes posteriores do curso, proporcionando aos discentes critérios objetivos para selecionar, adaptar e justificar o uso de abordagens ágeis em cenários acadêmicos ou profissionais.

2.1 ARTEFATOS DO PROJETO

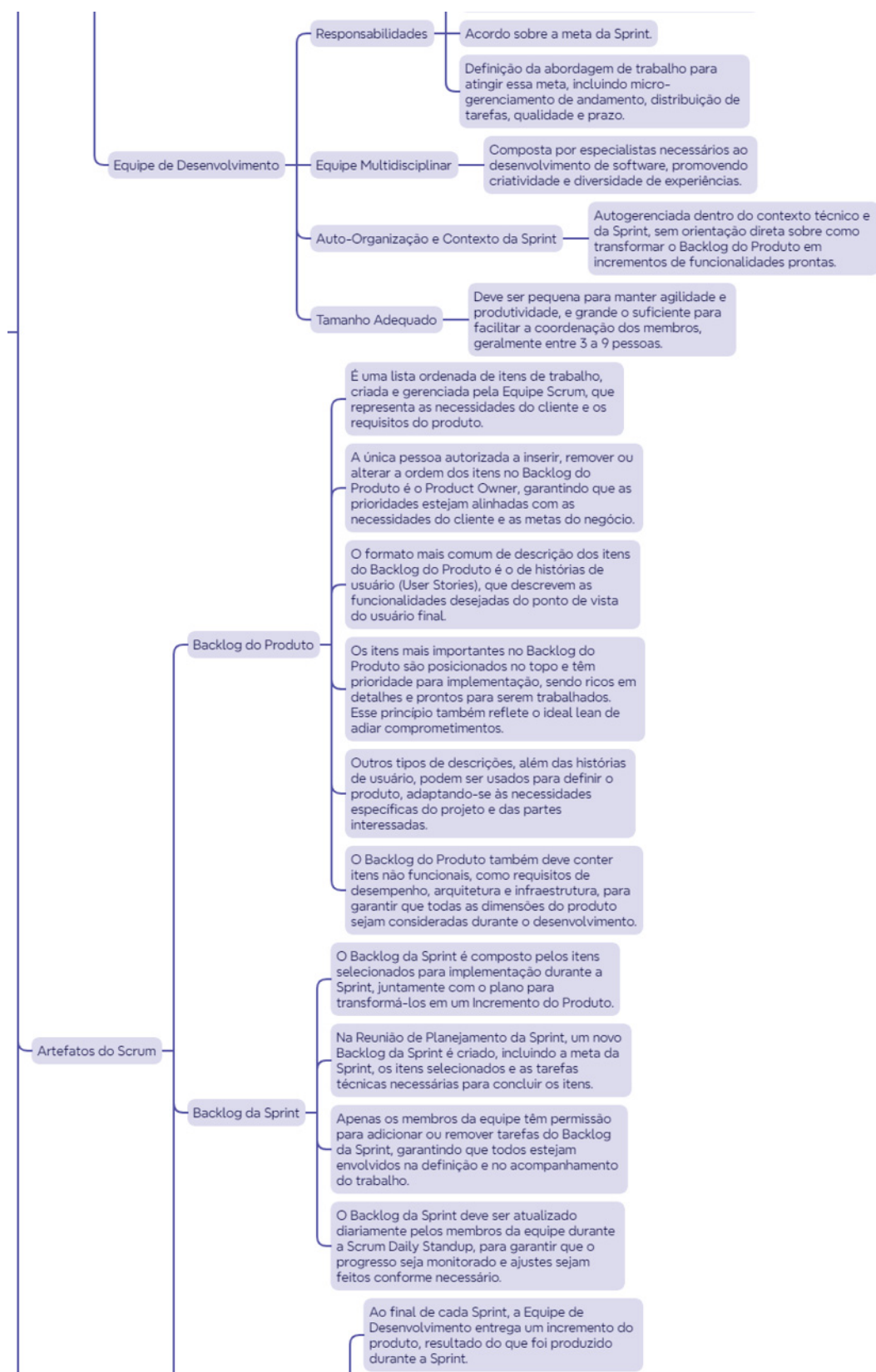
Um Mapa Mental foi criado com a ajuda do *software* Xmind, para consolidar em nove ramos principais, o corpo de conhecimentos explorado na disciplina.

FIGURA 01 - MAPA MENTAL SCRUM 1



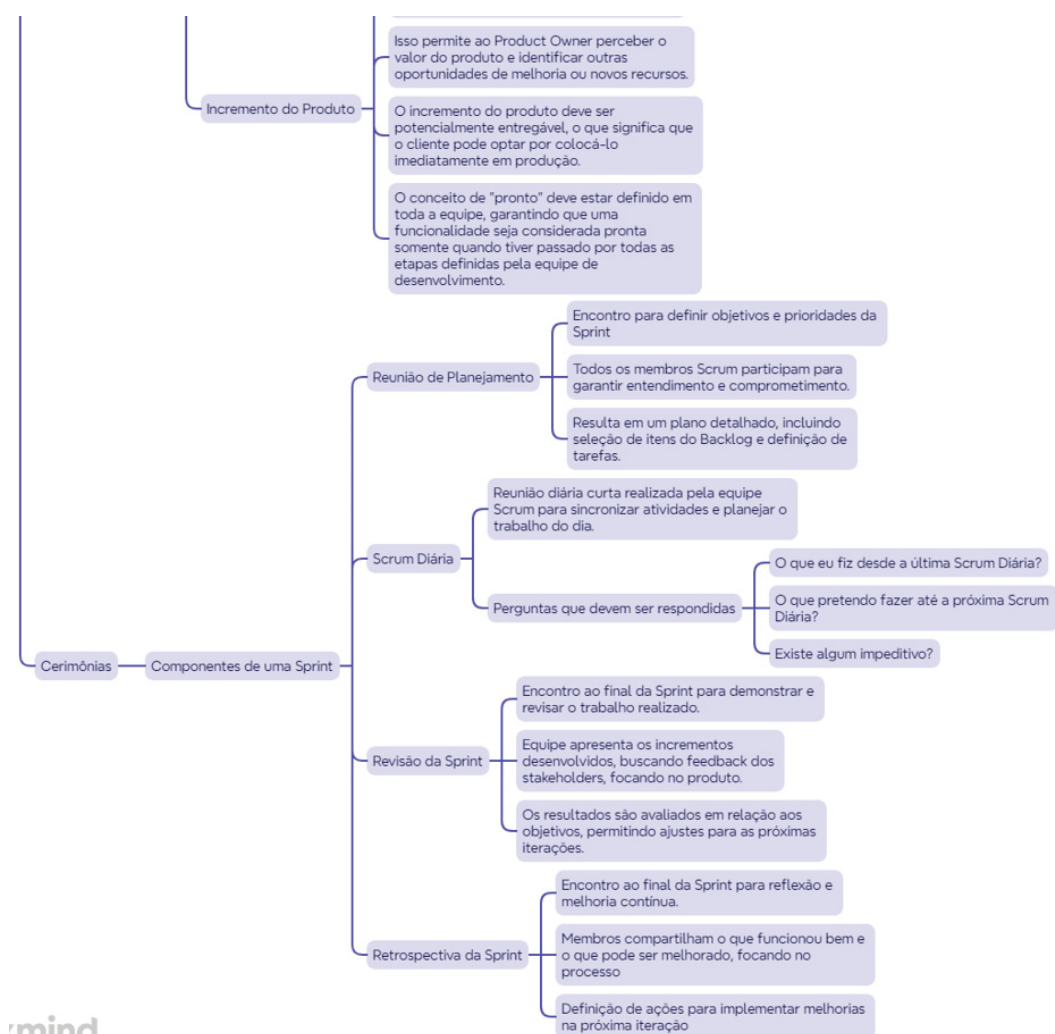
FONTE: O autor (2025).

FIGURA 02 - MAPA MENTAL SCRUM 2



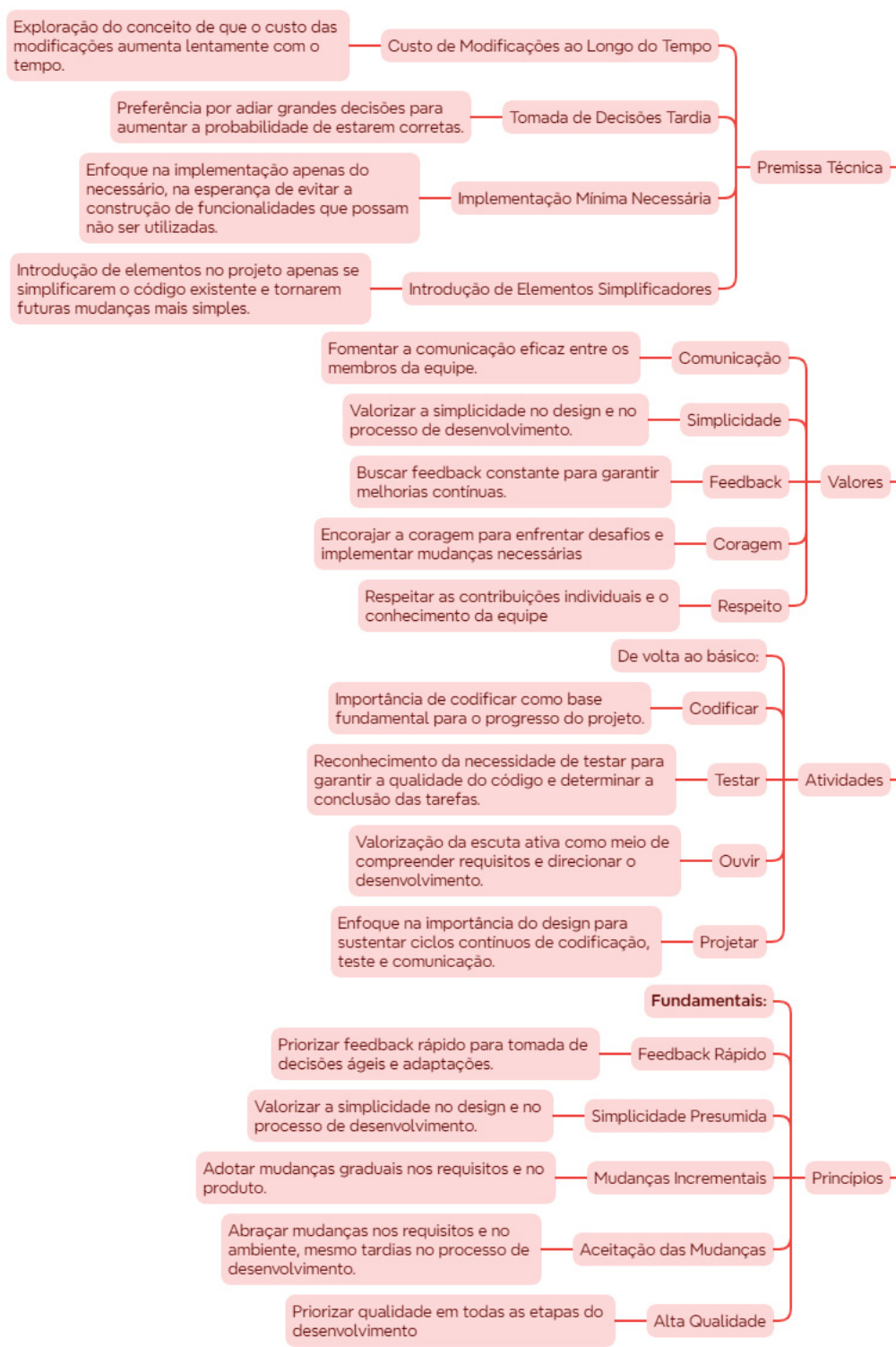
FONTE: O autor (2025).

FIGURA 03 - MAPA MENTAL SCRUM 3



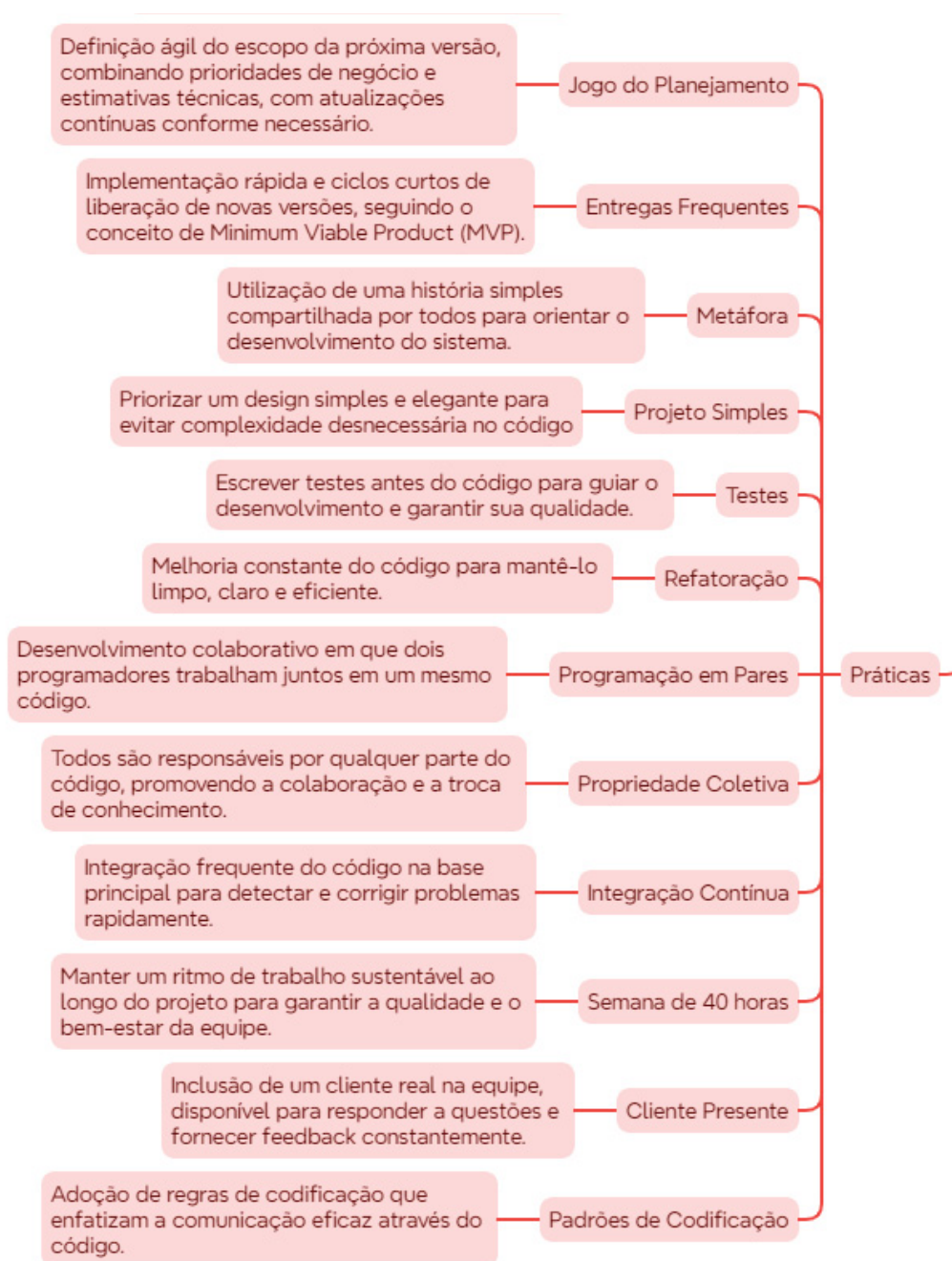
FONTE: O autor (2025).

FIGURA 04 - MAPA MENTAL EXTREME PROGRAMMING 1



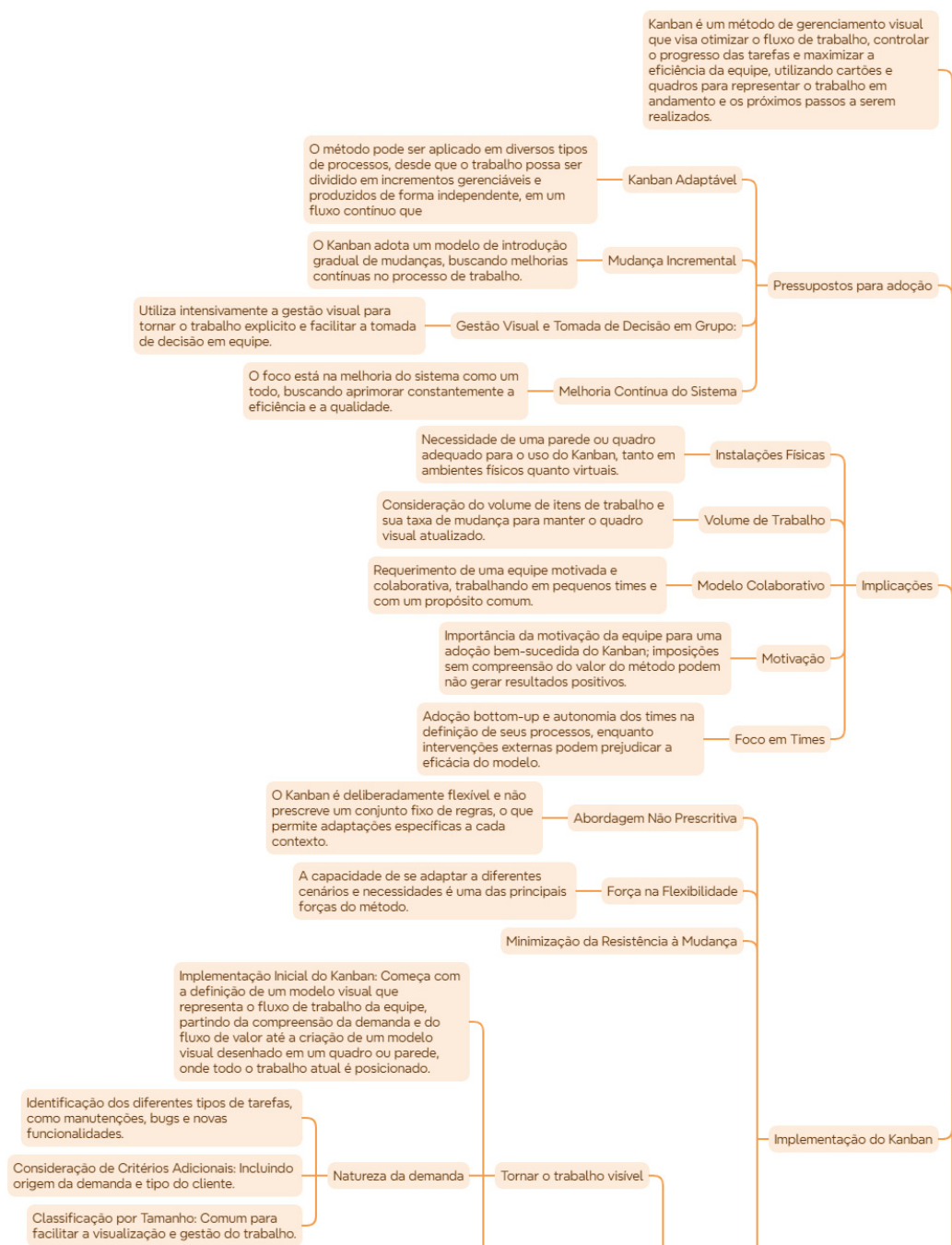
FONTE: O autor (2025).

FIGURA 05 - MAPA MENTAL EXTREME PROGRAMMING 2



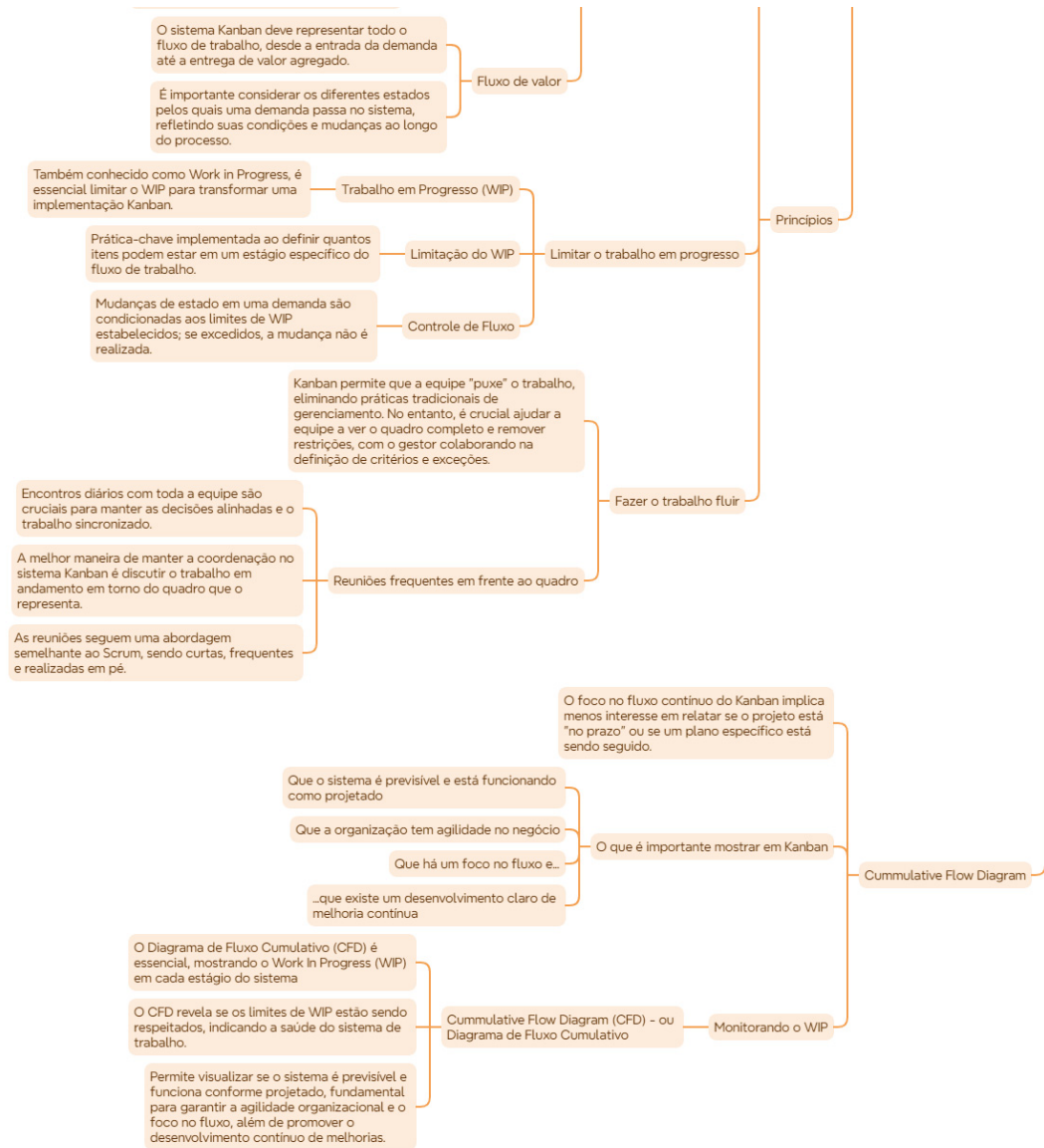
FONTE: O autor (2025).

FIGURA 06 - MAPA MENTAL KANBAN 1



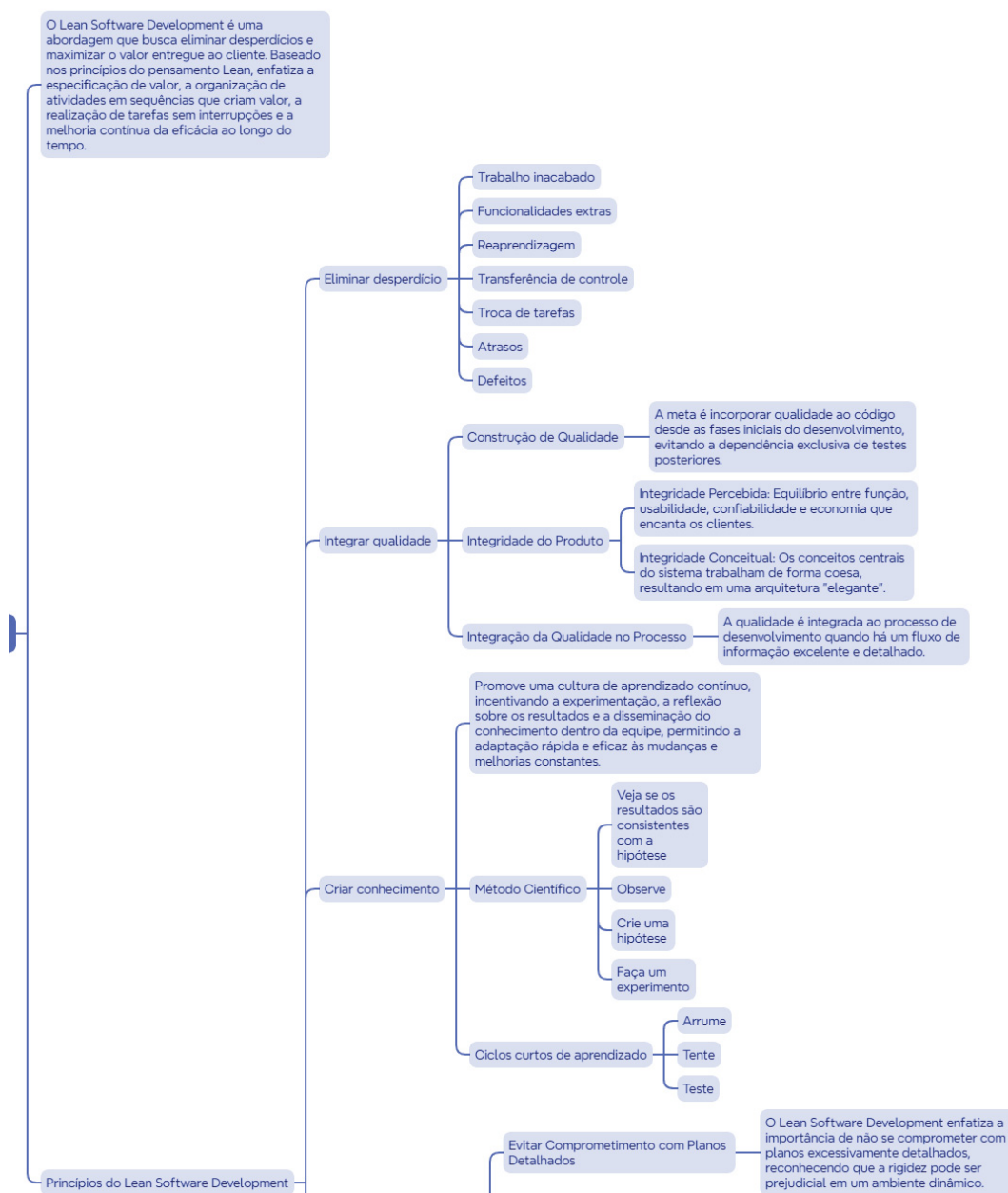
FONTE: O autor (2025).

FIGURA 07 - MAPA MENTAL KANBAN 2



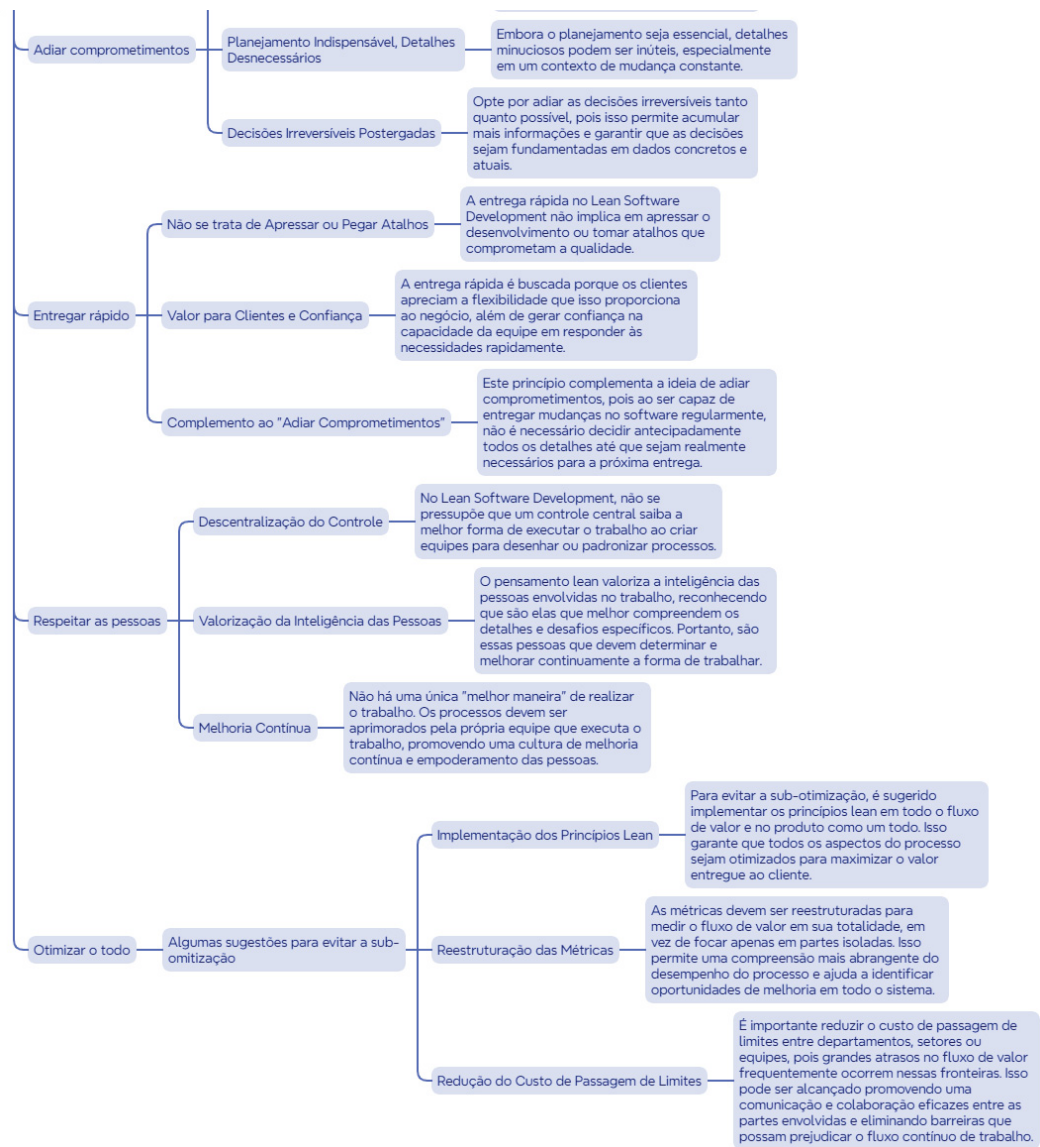
FONTE: O autor (2025).

FIGURA 08 - MAPA MENTAL LEAN SOFTWARE 1



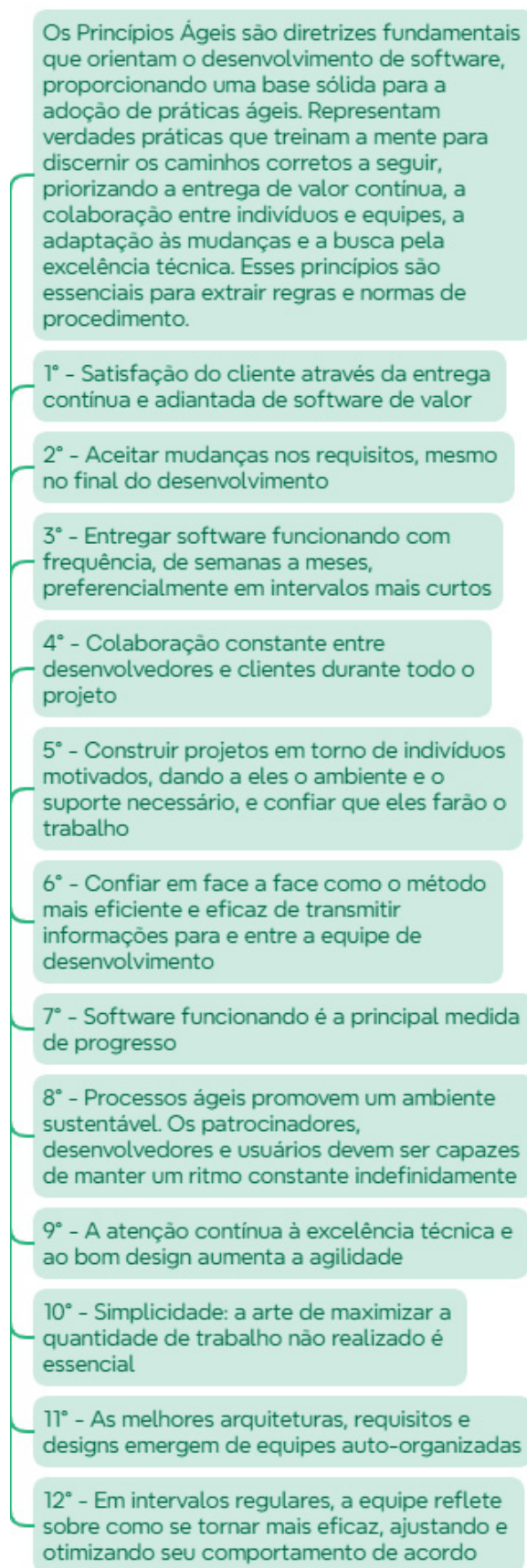
FONTE: O autor (2025).

FIGURA 09 - MAPA MENTAL LEAN SOFTWARE 2



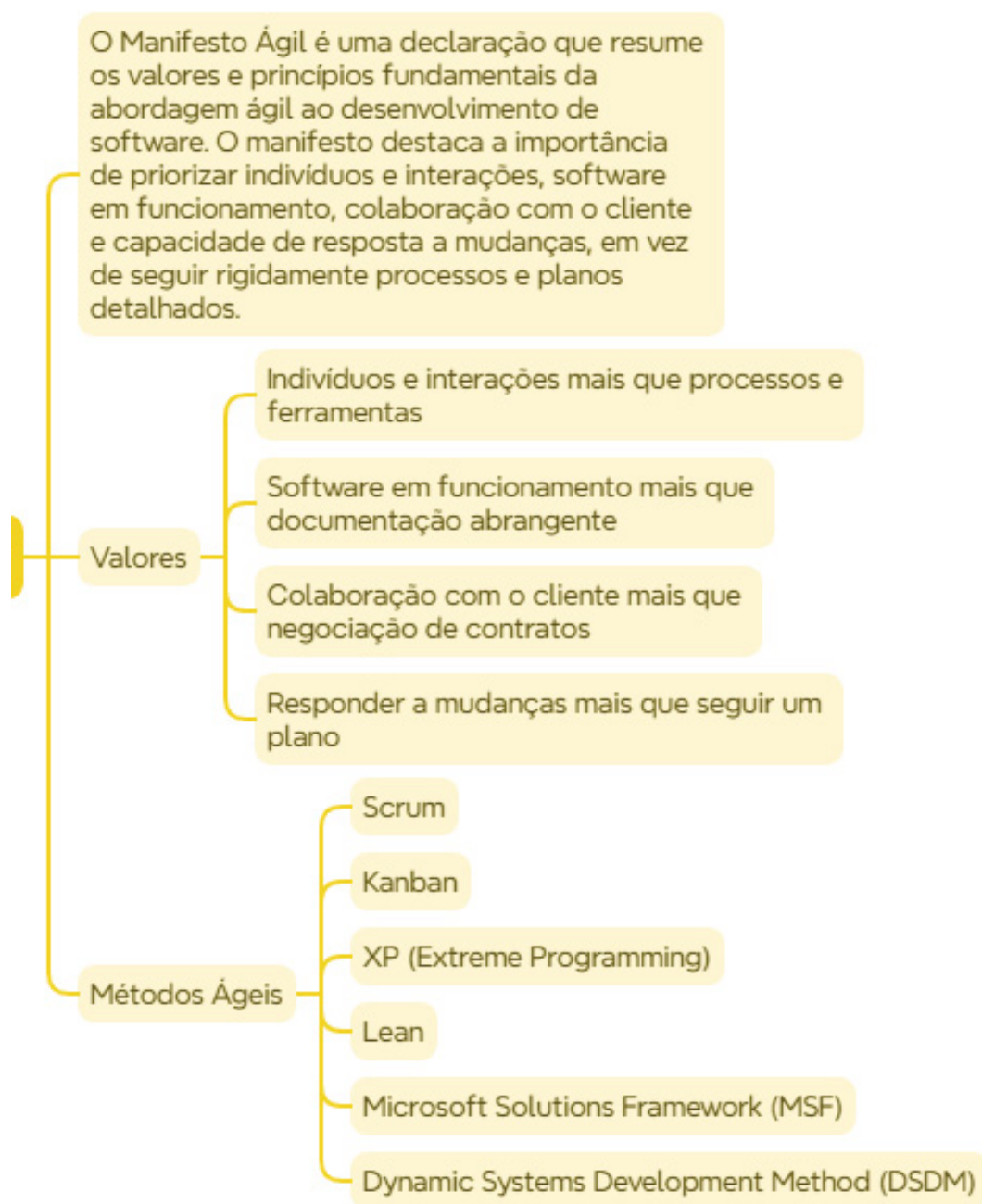
FONTE: O autor (2025).

FIGURA 10 - MAPA MENTAL PRINCÍPIOS ÁGEIS



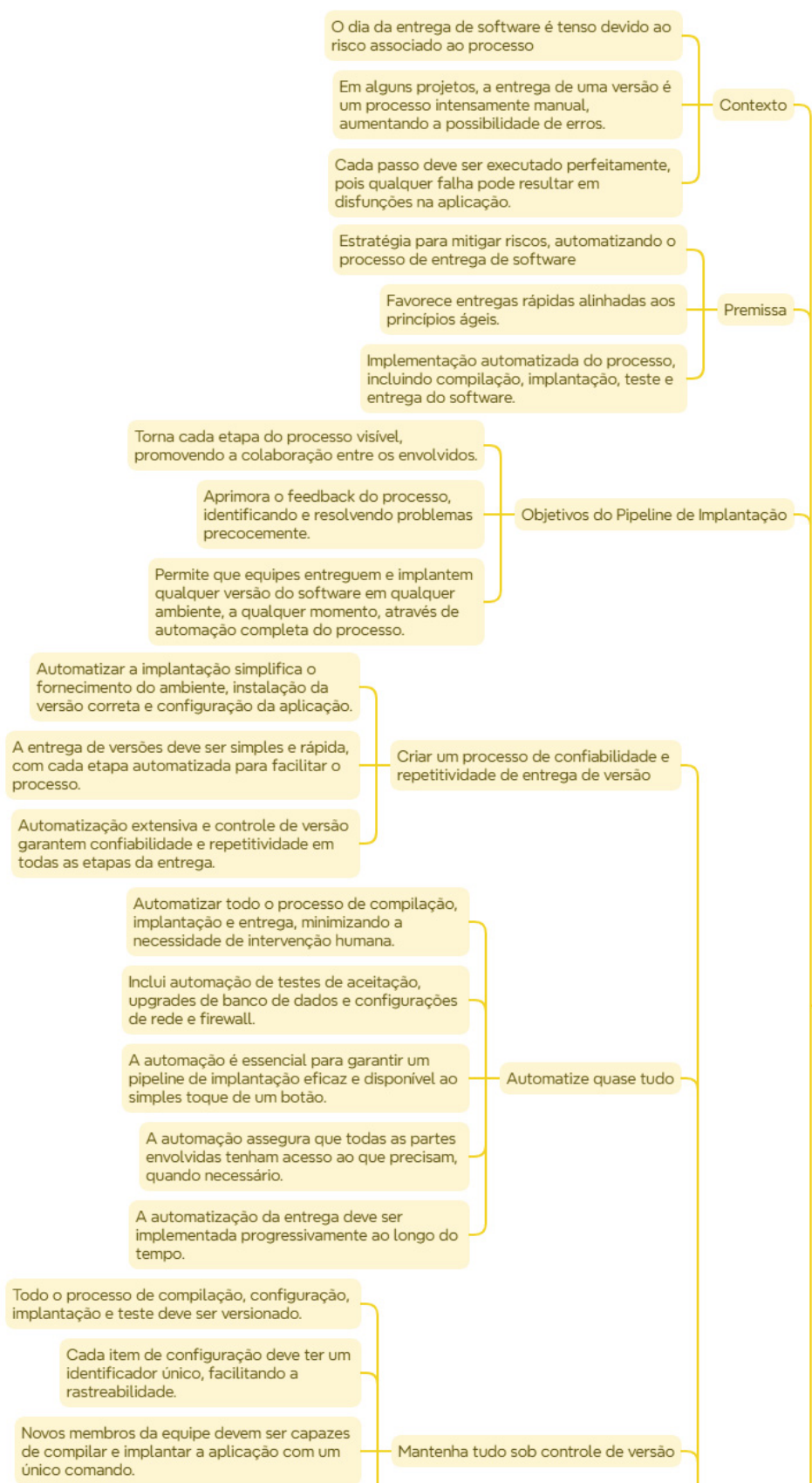
FONTE: O autor (2025).

FIGURA 11 - MAPA MENTAL MANIFESTO ÁGIL



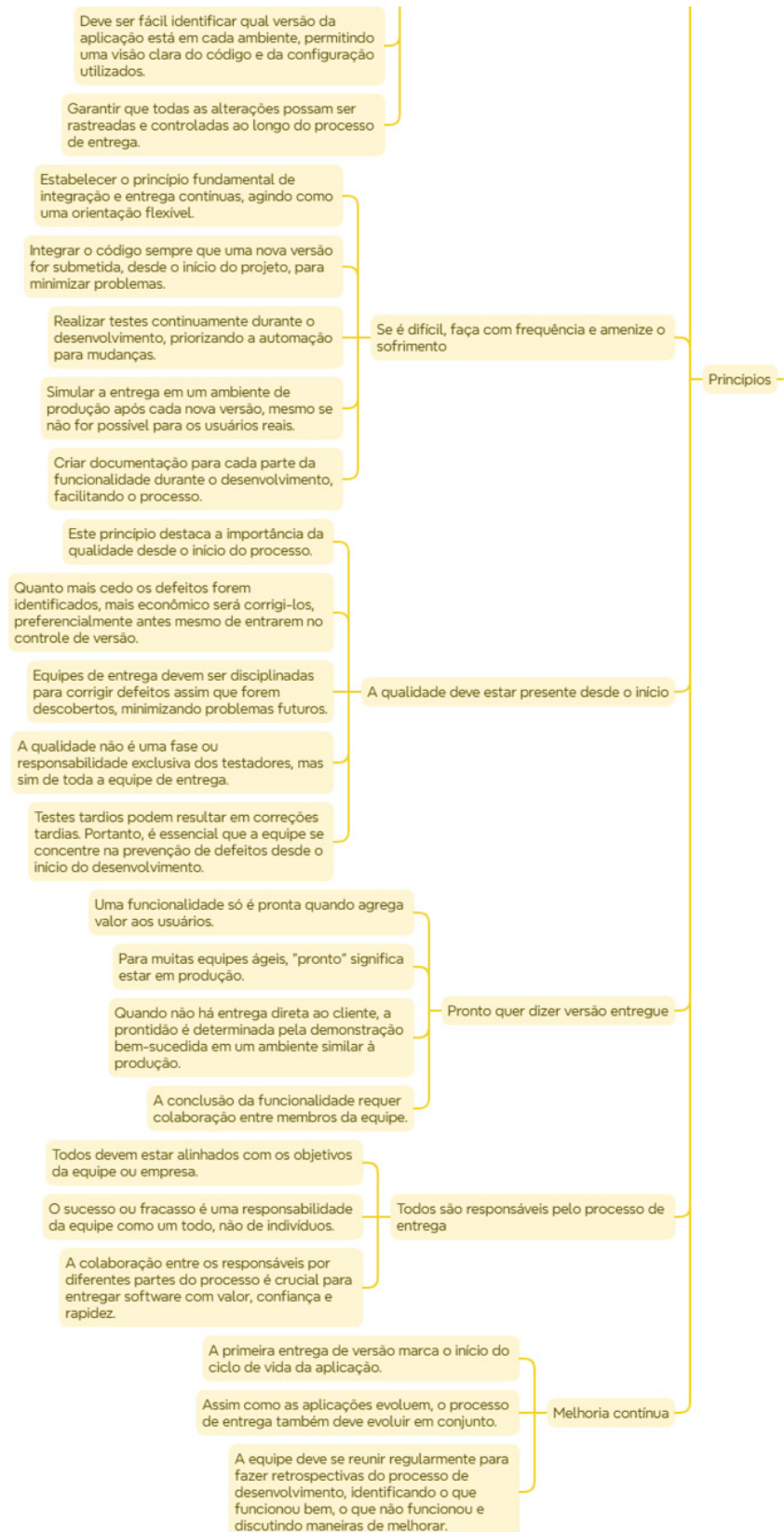
FONTE: O autor (2025).

FIGURA 12 - MAPA MENTAL ENTREGA CONTÍNUA 1



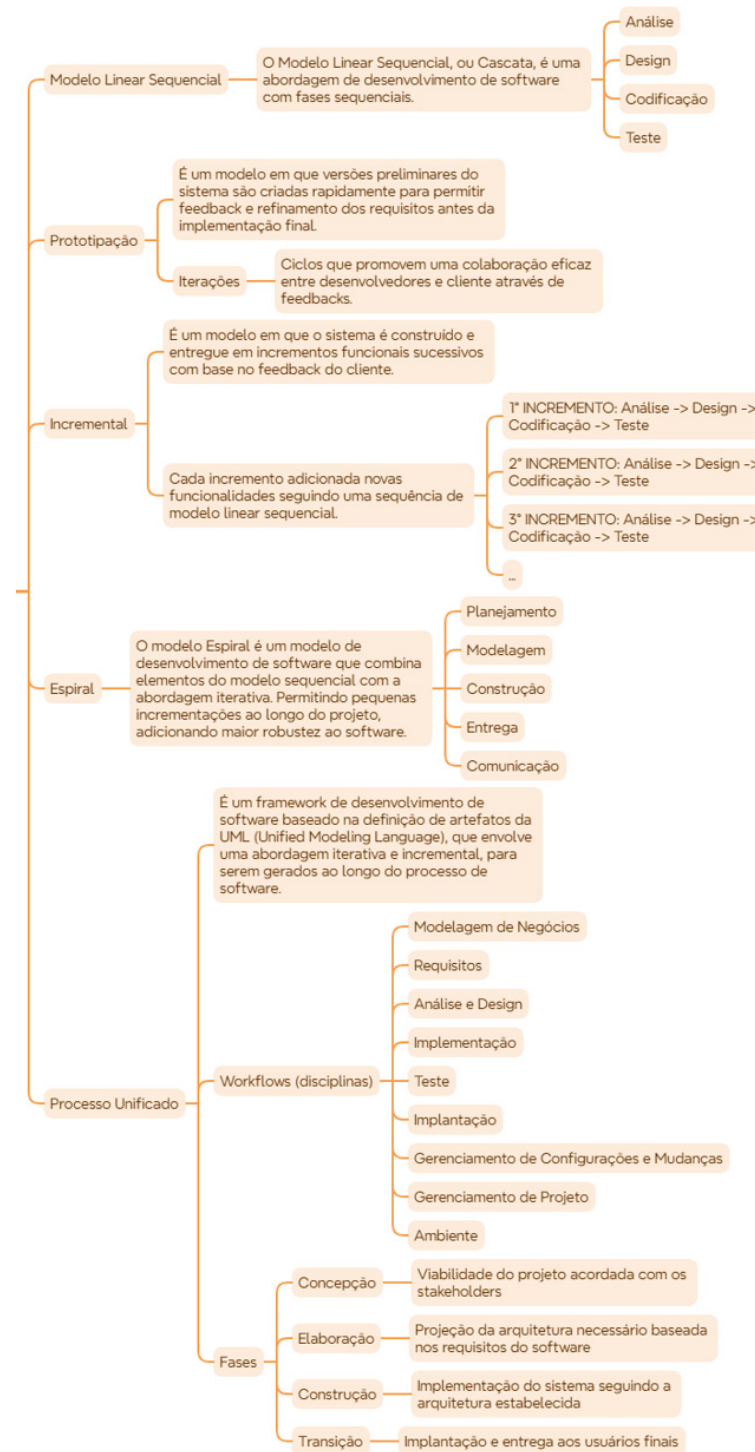
FONTE: O autor (2025).

FIGURA 13 - MAPA MENTAL ENTREGA CONTÍNUA 2



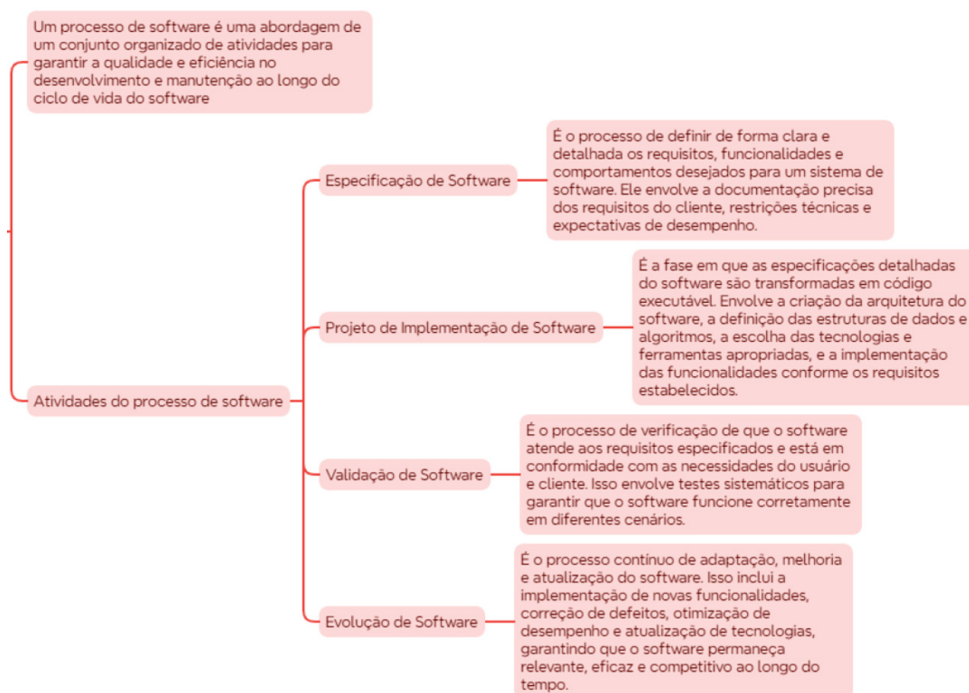
FONTE: O autor (2025).

FIGURA 14 - MAPA MENTAL MODELOS TRADICIONAIS DE PROCESSO 1



FONTE: O autor (2025).

FIGURA 15 - MAPA MENTAL PROCESSO DE SOFTWARE



FONTE: O autor (2025).

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

O objetivo das disciplinas MAG foi construir, de ponta a ponta, o modelo conceitual de um sistema de gestão de condomínio. O trabalho iniciou-se com diagramas de casos de uso – níveis 1 e 2: no nível 1 foram mapeadas interações globais do ator *Síndico*; no nível 2, as funcionalidades foram refinadas de “Pesquisar Morador” e “Pesquisar Apartamento”, para e “Manter Moradores” e “Manter Apartamentos”, por exemplo. Esses diagramas forneceram a visão de escopo e serviram como contrato de fronteira para as disciplinas posteriores.

Em seguida, a equipe criou histórias de usuário no formato *Como <ator>, quero <objetivo>, para <benefício>*. O exemplo-guia (HU001 - *Pesquisar Apartamento*) ilustra o padrão: *SENDO um síndico, QUERO pesquisar informações de apartamentos, PARA facilitar a gestão e manutenção dos registros*.

Cada história recebeu uma lista de critérios de aceitação (oito no total para o exemplo HU001, cobrindo exibição, inclusão, edição, exclusão e pesquisa) e regras de negócio, detalhadas em linguagem Gherkin para dar suporte a BD.

A modelagem estrutural foi capturada em um Diagrama de Classes com atributos e relacionamentos correspondentes a entidades-chave (Apartamento, Morador, Veículo, Bloco). Esse diagrama, acompanhado de Diagramas de Sequência (ex.: DS1 - *Pesquisar Apartamento* e DS4 - *Manter Moradores*), documentou o fluxo de mensagens entre fronteira de interface, camada de serviço e persistência, servindo como guia direto para a implementação em *Spring Boot* e *PostgreSQL*.

A abordagem incremental adotada nas duas disciplinas refletiu princípios ágeis de feedback rápido e documentação leve. Artefatos eram revisados ao final de cada iteração e ajustados conforme retorno dos membros da equipe (*stakeholders*).

A integração com demais componentes do curso é clara: os diagramas UML alimentaram scripts de criação de tabelas utilizados em BD; as histórias de usuário entraram no *backlog* priorizado em GAP1; os fluxos de sequência nortearam a divisão de componentes front-end em WEB1, WEB2, MOB1 e MOB2; e os critérios de aceitação transformaram-se em casos de teste unitários em INFRA.

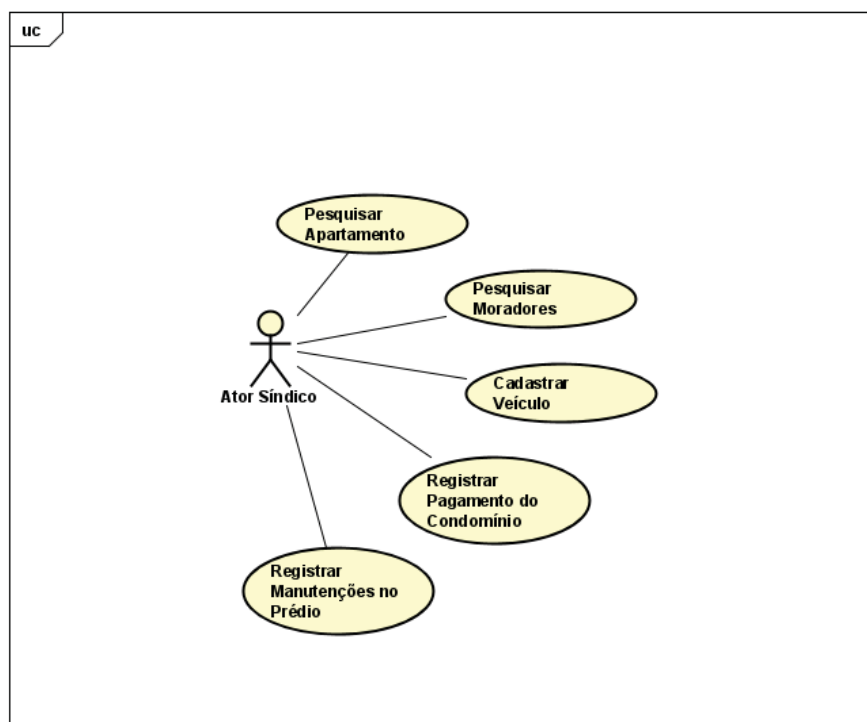
Do ponto de vista pedagógico, MAG1 e MAG2 consolidaram a importância da modelagem como ponte entre requisitos e código. Sem um *design* bem definido, a fase de coding tende a se tornar caótica e suscetível a retrabalho. Já com modelos

claros, o desenvolvimento flui de maneira mais previsível e aderente aos princípios de qualidade do Manifesto Ágil.

Em síntese, o conjunto de artefatos produzidos - casos de uso hierárquicos, histórias de usuário com critérios verificáveis, diagrama de classes coerente e diagramas de sequência alinhados - forneceu um *backbone* consistente para todo o ciclo de vida dos *softwares* trabalhados no curso, demonstrando como a modelagem ágil pode ser enxuta, porém suficientemente rica para sustentar entregas frequentes e de valor agregado ao negócio.

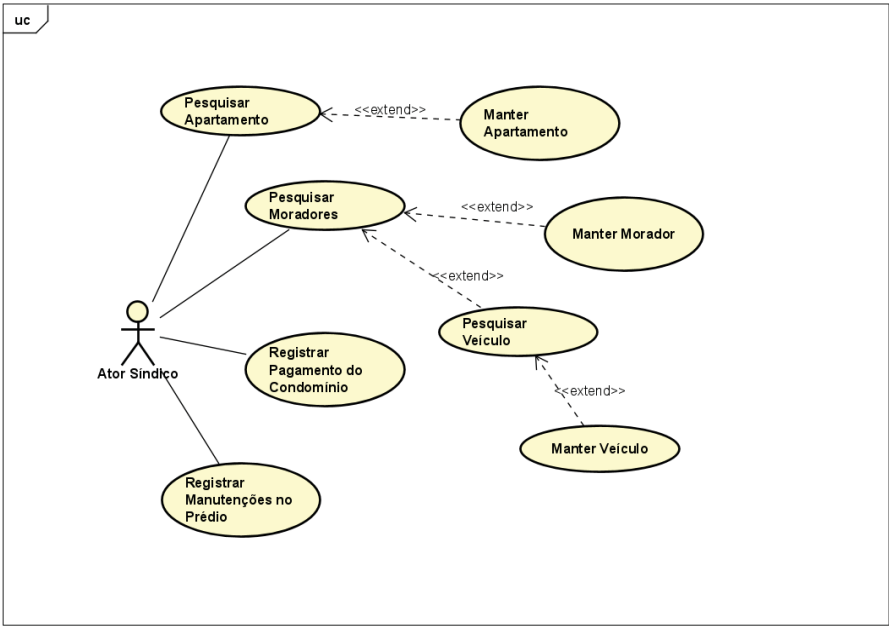
3.1 ARTEFATOS DO PROJETO - MAG1

FIGURA 16 – DIAGRAMA DE CASOS DE USO NÍVEL 2



FONTE: O autor (2025).

FIGURA 17 - DIAGRAMA DE CASOS DE USO NÍVEL 2 DETALHADO.



FONTE: O autor (2025).

Histórias de Usuário

HU001 - Pesquisar Apartamento

SENDO um síndico,
QUERO pesquisar informações de apartamentos,
PARA facilitar a gestão e manutenção dos registros dos apartamentos.

FIGURA 18 - DESENHO DE TELA PESQUISAR APARTAMENTO

Voltar + Novo Pesquisar Morador		
Nome Do Morador	Nº Apto	Ações
Nome Morador 38	019	Editar Excluir
Nome Morador 14	046	Editar Excluir
Nome Morador 5	020	Editar Excluir
Nome Morador 40	011	Editar Excluir

FONTE: O autor (2025).

LISTA DE CRITÉRIOS DE ACEITAÇÃO

1. Deve preencher a tabela com todos os apartamentos cadastrados
2. Deve permitir incluir um novo apartamento
3. Deve permitir alterar os dados de um apartamento cadastrado
4. Deve permitir excluir um apartamento cadastrado
5. Deve permitir consultar os dados de um apartamento cadastrado
6. Deve pesquisar os apartamentos na tabela bloco da tela
7. Deve pesquisar os apartamentos na tabela da tela
8. Deve voltar à tela anterior

CRITÉRIOS DE ACEITAÇÃO - DETALHAMENTO

- 1) Deve preencher a tabela com todos os apartamentos cadastrados.

Dado que	Nenhum apartamento está cadastrado no sistema
Quando	O síndico acessa a tela de pesquisa de apartamentos
Então	A tabela da tela deve ser preenchida com todos os dados do banco de dados

- 2) Deve permitir incluir um novo apartamento.

Dado que	O síndico está na tela de gerenciamento de apartamento
Quando	Seleciona a opção para incluir um novo apartamento
Então	O sistema chama a HU002 – Manter Apartamento passando o parâmetro 'Novo'

- 3) Deve permitir alterar os dados de um apartamento cadastrado.

Dado que	O síndico está na tela de gerenciamento de apartamentos
Quando	Seleciona um apartamento da lista e clica no botão 'Editar'
Então	O sistema chama o HU002 - Manter Apartamento passando o parâmetro 'Editar' e os dados do candidato da linha

- 4) Deve permitir excluir um apartamento cadastrado.

Dado que	A lista com os apartamentos está na tela
Quando	O botão 'Excluir' é pressionado em uma determinada linha

Então	O sistema deve exibir uma confirmação solicitando a confirmação da exclusão do apartamento selecionado
--------------	--

5) Deve permitir consultar os dados de um apartamento cadastrado.

Dado que	A lista de apartamentos está na tela
Quando	O síndico clica em um apartamento da lista para consulta
Então	O sistema chama a HU002 – Manter Apartamento passando o parâmetro “Consultar” e deve exibir todos os dados do apartamento selecionado

6) Deve pesquisar os apartamentos na tabela bloco da tela.

Dado que	A lista de blocos de apartamentos está na tela
Quando	O síndico digita o bloco de um apartamento na barra de pesquisa
Então	O sistema filtra e apresenta na tela somente os blocos de apartamentos que obedeçam ao critério

7) Deve pesquisar os apartamentos na tabela da tela

Dado que	A lista de apartamentos está na tela
Quando	O síndico digita o número de um apartamento na barra de pesquisa
Então	O sistema filtra e apresenta na tela somente os apartamentos que obedeçam ao critério

8) Deve voltar à tela anterior

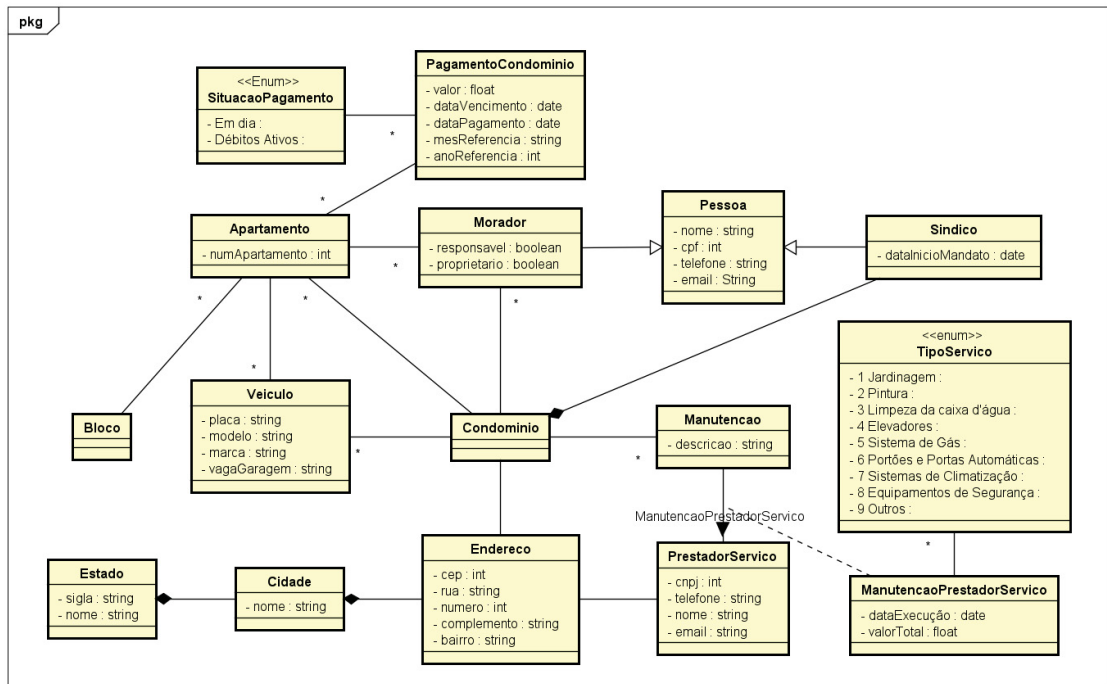
Dado que	Os apartamentos estão na tabela
Quando	O botão 'Voltar' é pressionado
Então	O sistema volta para a tela anterior

REGRAS DE NEGÓCIO

Não há

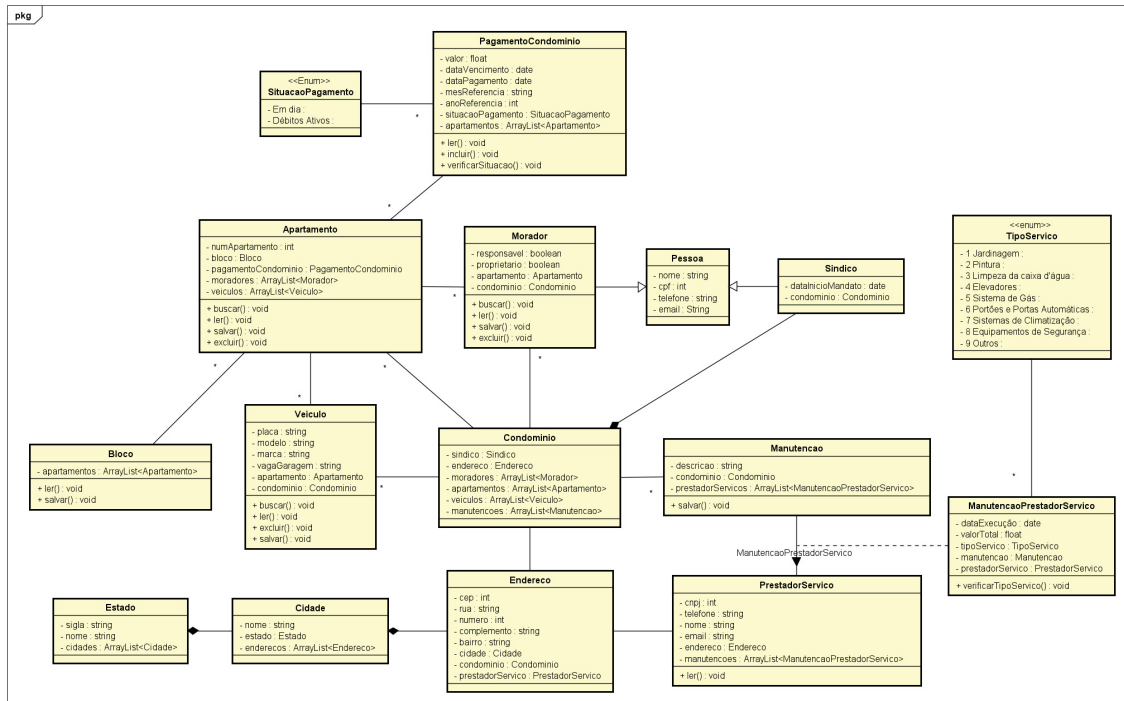
3.2 ARTEFATOS DO PROJETO - MAG2

FIGURA 19 - DIAGRAMA DE CLASSES COM ATRIBUTOS ASSOCIADOS



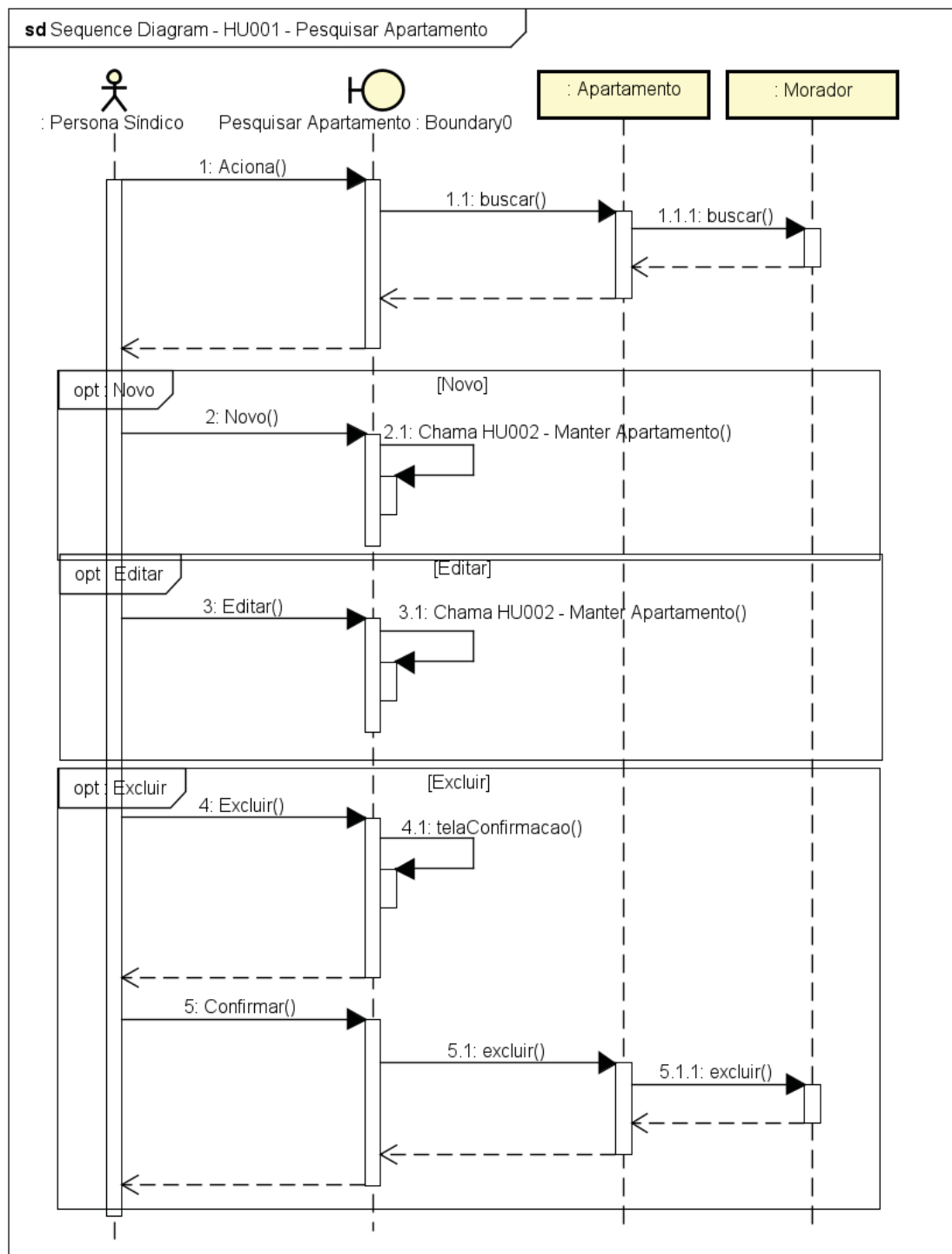
FONTE: O autor (2025).

FIGURA 20 - DIAGRAMA DE SEQUÊNCIA



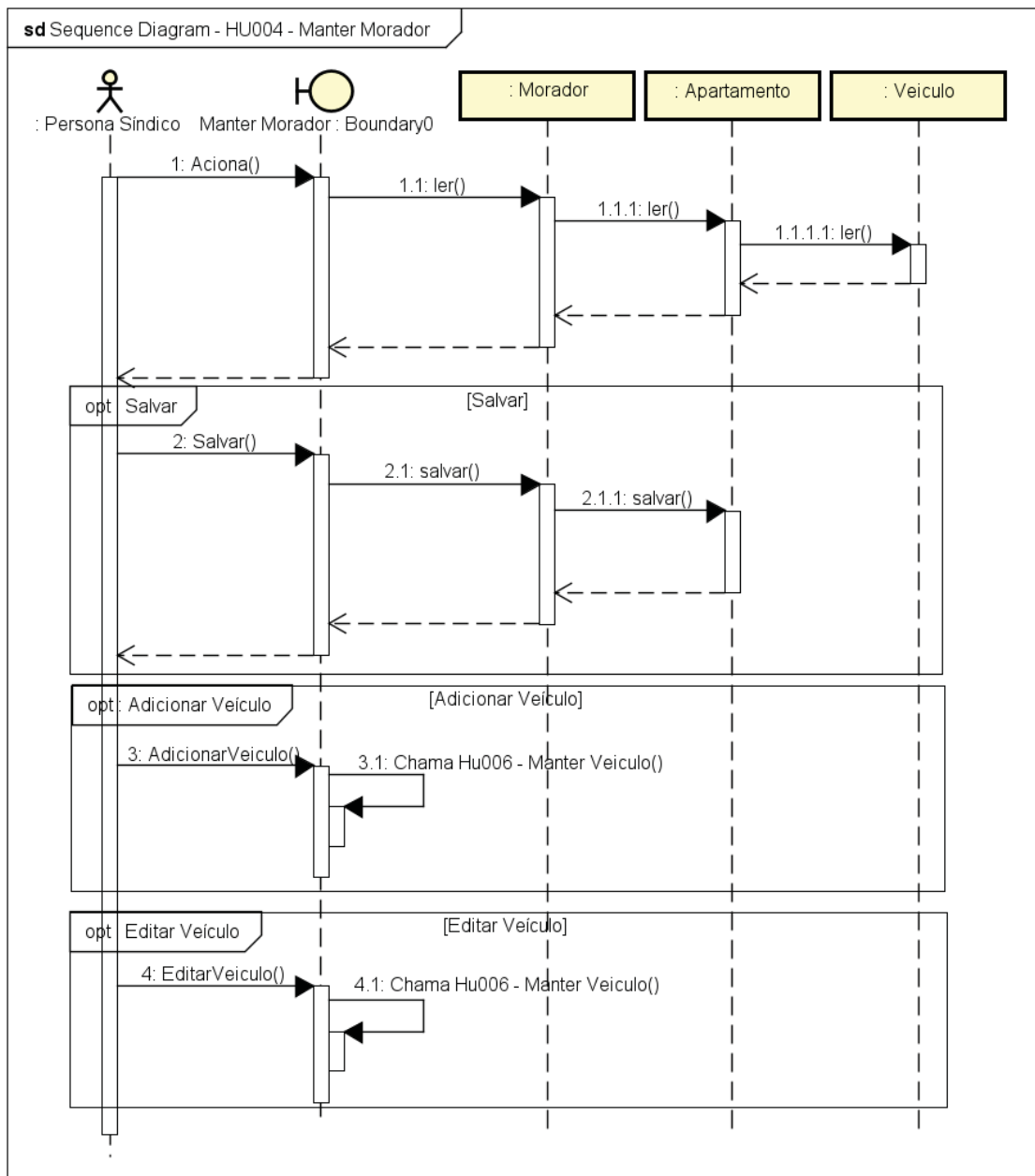
FONTE: O autor (2025).

FIGURA 21 - DIAGRAMA DE SEQUÊNCIA DA HISTÓRIA DE USUÁRIO 01



FONTE: O autor (2025).

FIGURA 22 - DIAGRAMA DE SEQUÊNCIA DA HISTÓRIA DE USUÁRIO 04



FONTE: O autor (2025).

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

Em GAP1, o projeto envolveu a elaboração de um plano de *release* para um jogo, demonstrando a aplicação prática de conceitos de gerenciamento ágil. O processo iniciou-se com o cálculo da velocidade da equipe, um indicador crucial para a previsibilidade em ambientes ágeis. Este cálculo considerou tanto as horas disponíveis por *sprint* quanto a estimativa de *story points*, permitindo uma projeção realista da capacidade de entrega. Com base nessa velocidade, as histórias do backlog foram cuidadosamente priorizadas, alinhando-as ao valor de negócio e distribuindo as funcionalidades em *sprints* sucessivas.

O plano de *release* foi visualizado em um quadro detalhado, que não apenas continha as datas previstas de entrega e os requisitos específicos para cada *sprint*, mas também incorporava indicadores visuais como o *burn-down chart*. Este *chart* é fundamental para monitorar o progresso da *sprint*, permitindo que a equipe e os *stakeholders* identifiquem desvios e tomem ações corretivas em tempo real. A transparência proporcionada por esses artefatos visuais é uma pedra angular do gerenciamento ágil, facilitando a comunicação e o alinhamento contínuo.

Na disciplina GAP2, foi aprofundado os princípios de *Kanban* por meio de um jogo de simulação. Esta experiência prática permitiu a administração de um quadro de tarefas com limites de *Work in Progress* (WIP), um conceito central do *Kanban* que visa otimizar o fluxo de trabalho e evitar sobrecarga. A simulação também focou na medição de métricas como o *lead time*, que representa o tempo total desde o início até a conclusão de uma tarefa. Ao final do jogo, os resultados foram analisados criticamente por meio de um Diagrama de Fluxo Cumulativo (CFD). O CFD é uma ferramenta poderosa que visualiza o fluxo de trabalho ao longo do tempo, mostrando a evolução das fases (como *Ready*, *Analysis*, *Development*, *Test* e *Deployed*) e permitindo a identificação de gargalos. A análise desses gargalos levou a discussões sobre estratégias de melhoria contínua, como ajustes nos limites de WIP ou otimização de etapas específicas do processo. Essas atividades em GAP1 e GAP2 evidenciaram a importância da transparência, da adaptação contínua e da otimização do fluxo de trabalho na gestão de projetos ágeis, preparando os

alunos para aplicar esses conceitos em cenários reais de desenvolvimento de software.

4.1 ARTEFATOS DO PROJETO – GAP1

FIGURA 23 - PLANO DE RELEASE PARA UM JOGO

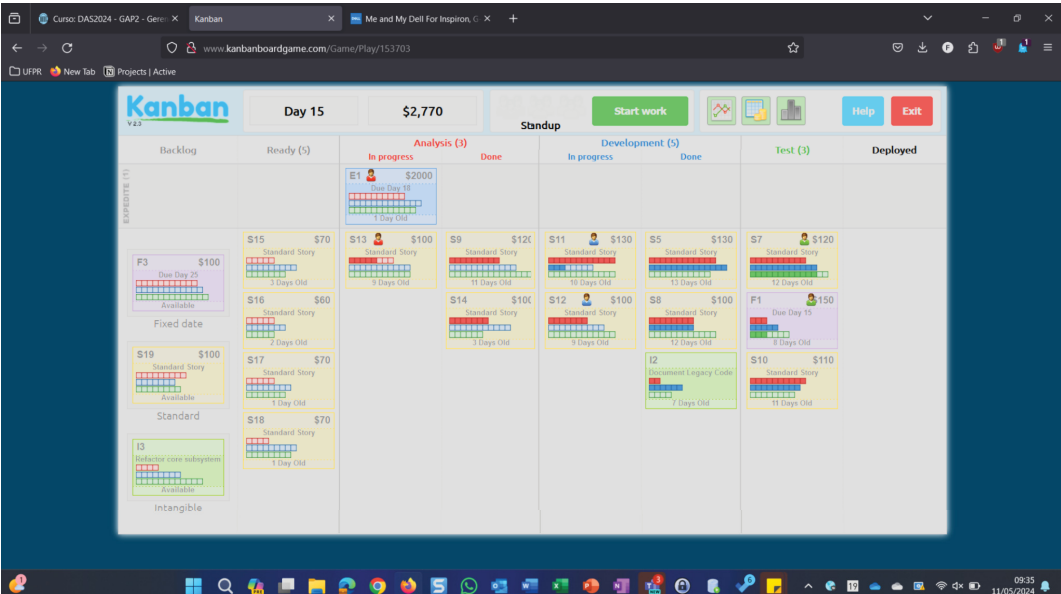
Gerenciamento Ágil de Projetos I
Profa. Dra. Rafaela Mantovani Fontana
Template para o Plano de Release

Cálculo da Velocidade:			
Horas disponíveis por dia: 4		Tamanho da Sprint: 2 Semanas	
Horas disponíveis por Sprint: 40		Velocidade: 5	
Plano de Release:			
Iteração/Sprint 1	Iteração/Sprint 2	Iteração/Sprint 3	Iteração/Sprint N
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
<HU001 – Desenvolver High Concept Document> SENDO um jogador que adora jogos casuais e desafiadores, QUERO empilhar paçocas para criar a maior torre possível, PARA marcar o máximo de pontos e competir no leaderboard. ESTIMATIVA (2)	<HU004 – Arte da Paçoca > SENDO um jogador que aprecia gráficos atraentes e temáticos, QUERO que as paçocas tenham uma arte detalhada e visualmente agradável, PARA manter o jogo visualmente interessante. ESTIMATIVA (1)	<HU007 - Interface do Usuário> SENDO um jogador, QUERO uma tela de menu inicial intuitiva, PARA navegar facilmente pelo jogo. ESTIMATIVA (2)	<HU10 – Sistema de Feedback> SENDO um desenvolvedor, QUERO implementar um sistema de feedback dentro do jogo, PARA coletar opiniões dos jogadores e fazer melhorias contínuas após o lançamento. ESTIMATIVA (2)
<HU002 – Protótipo de Físicas Inicial> SENDO um desenvolvedor, QUERO criar um protótipo jogável básico, PARA testar as mecânicas de empilhamento e física do jogo. ESTIMATIVA (2)	<HU005 – Cenário Piquenique> SENDO um jogador, QUERO diferentes cenários de fundo, PARA manter o jogo visualmente interessante. ESTIMATIVA (2)	<HU008 – Interface de Leaderboard> SENDO um jogador competitivo, QUERO uma tela de leaderboard, PARA ver minha posição em relação a outros jogadores. ESTIMATIVA (2)	<HU011 – Otimização Para Diferentes Dispositivos Móveis> SENDO um jogador, QUERO que o jogo seja otimizado para diferentes dispositivos móveis, PARA garantir uma jogabilidade suave e sem travamentos. ESTIMATIVA (1)
<HU003 – Testar Desafio e Diversão da Mecânica de Física> SENDO um jogador, QUERO que as paçocas caiam se empilhadas incorretamente, PARA que o jogo seja desafiador e com físicas divertidas. ESTIMATIVA (1)	<HU006 – Cenário Torre de Pisa> SENDO um jogador, QUERO diferentes cenários de fundo, PARA manter o jogo visualmente interessante. ESTIMATIVA (2)	<HU009 – Sistema de Recompensas> SENDO um jogador, QUERO desbloquear prêmios ao atingir certos marcos, PARA ser recompensado pelo meu desempenho e dedicação. ESTIMATIVA (1)	<HU12 - Polimento> SENDO um desenvolvedor, QUERO polir o jogo e corrigir bugs, PARA garantir uma experiência suave e profissional. ESTIMATIVA (2)

FONTE: O autor (2025).

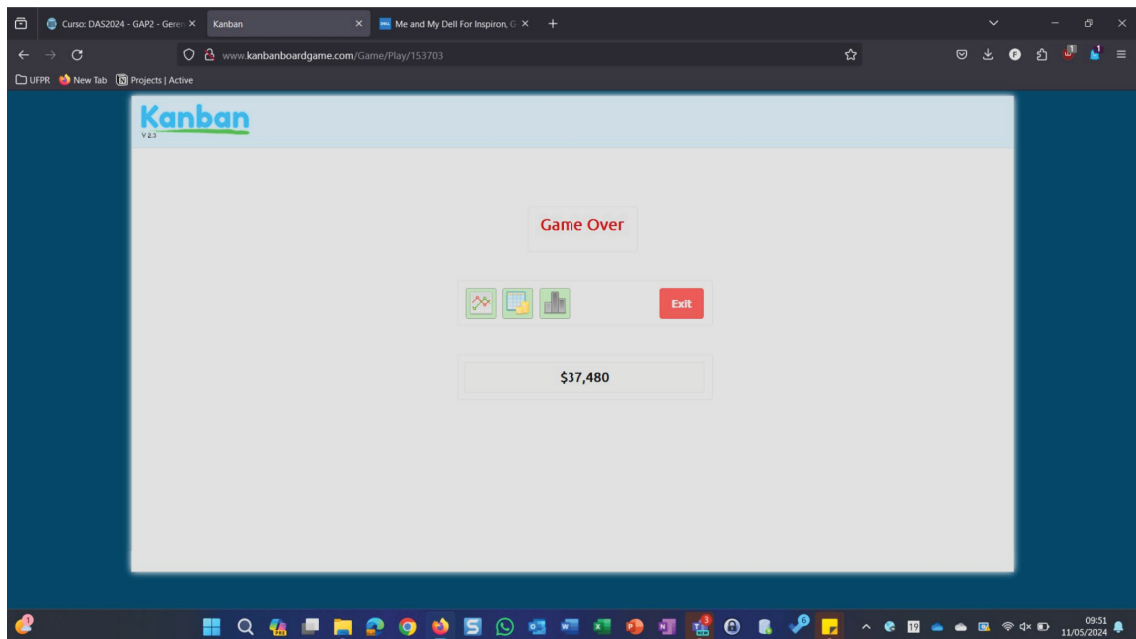
4.2 ARTEFATOS DO PROJETO – GAP2

FIGURA 24 - SIMULAÇÃO KANBAN



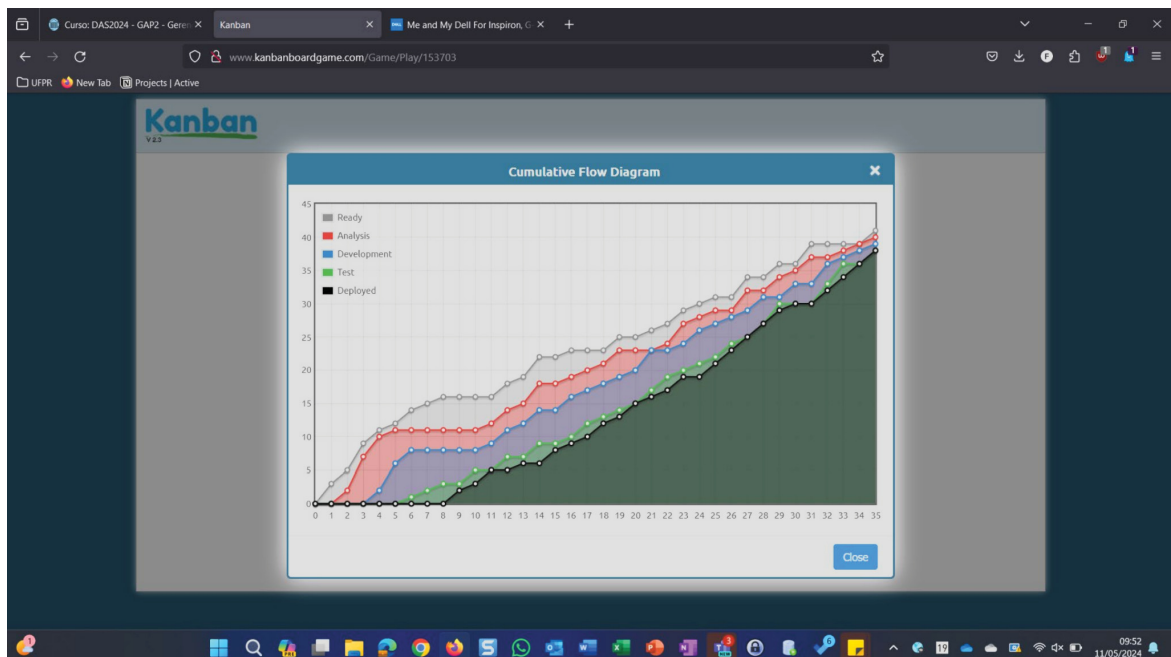
FONTE: O autor (2025).

FIGURA 25 - SIMULAÇÃO KANBAN RESULTADO



FONTE: O autor (2025).

FIGURA 26 - DIAGRAMA DE FLUXO CUMULATIVO (CFD)



FONTE: O autor (2025).

5 DISCIPLINA: INTRO – INTRODUÇÃO À PROGRAMAÇÃO

Ao longo das quarenta e cinco aulas do módulo, foram consolidados os fundamentos da lógica de programação - variáveis, operadores, estruturas de decisão e repetição - e introduzidos conceitos essenciais de Programação Orientada a Objetos, como classes, herança e polimorfismo. O conteúdo avançou para tratamento de exceções, coleções da *Java Collections Framework*, testes de unidade com JUnit, princípios de *Test-Driven Development*, acesso a dados com JDBC e adoção do padrão DAO. Complementarmente, abordou-se controle de versão no Git, configuração de ambientes no *NetBeans* e VS Code e automação de builds, oferecendo um percurso teórico-prático que serviu de alicerce para o trabalho final e para as disciplinas subsequentes.

O desafio proposto pela disciplina consistiu em implementar o *backend* de um sistema bancário didático, incluindo cadastros de clientes, contas correntes e contas de investimento. Foram disponibilizados um diagrama de classes, um conjunto de testes JUnit e um *script* DDL para MySQL. A entrega foi considerada concluída quando todos os 42 testes passaram - meta atingida integralmente.

O desenvolvimento evoluiu em três frentes pedagógicas: primeiro, sintaxe Java e *TDD*, em que a base do sistema foi construída por meio de testes de unidade que guiavam cada funcionalidade, reforçando o ciclo *Red-Green-Refactor*; segundo, Programação Orientada a Objetos, com herança, polimorfismo e exceções estruturando a hierarquia Conta → ContaCorrente/ContaInvestimento, garantindo encapsulamento das regras de negócio; e terceiro, persistência com JDBC e DAO, na qual a camada de dados utilizou *DAOs* especializados instanciados via *DaoFactory*, enquanto a *ConnectionFactory* centralizou os parâmetros de conexão, promovendo a separação de responsabilidades.

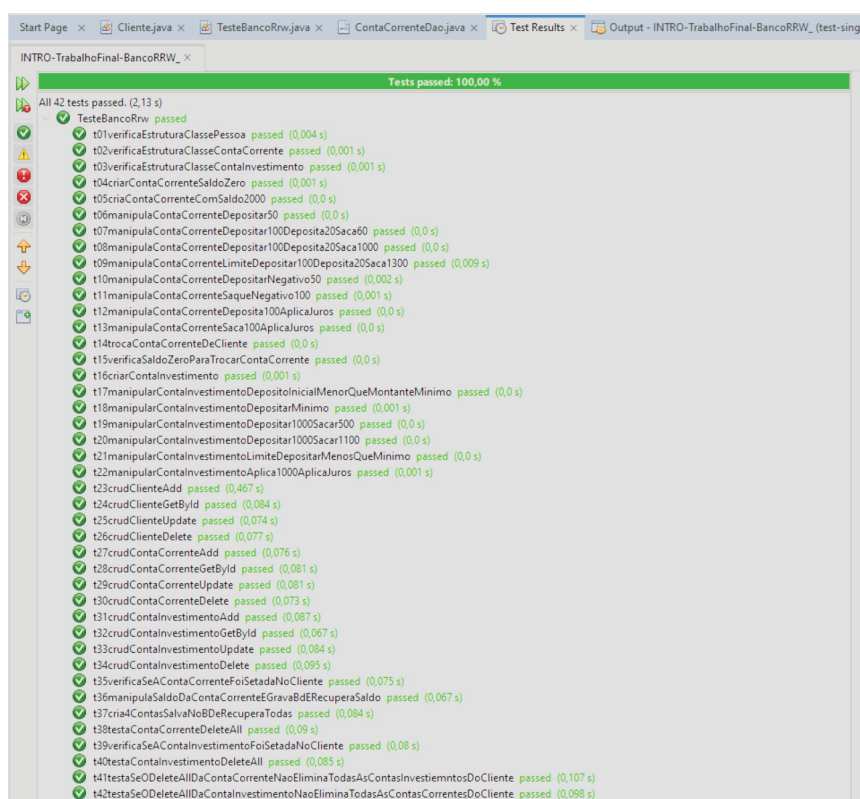
Essa abordagem incremental consolidou o controle de versão no Git, documentação de código e revisão em pares e serviu de base para reutilizações posteriores, como o projeto CRUD-pessoa de WEB1, WEB2 e o módulo de folha de pagamento nas disciplinas BD e AAP. Os modelos criados nesta etapa foram refinados em MAG 1-2, gerando diagramas de classes detalhados que, por sua vez, originaram *scripts* Liquibase aplicados em BD. A prática de TDD aqui desenvolvida inspirou a estratégia de testes automatizados em TEST, enquanto o padrão DAO foi reaproveitado na camada de repositório em AAP. Por fim, o fluxo de *commits* em Git

e o uso do NetBeans estabeleceram a fundação para os pipelines CI/CD montados na disciplina INFRA.

A disciplina consolidou o ciclo “*Red-Green-Refactor*”: partindo de testes quebrados, o código foi incrementado até todos passarem, reforçando *design for testability*. A integração com MySQL demonstrou desafios reais de ambiente - configuração de *Path*, driver JDBC e script DDL - resolvidos por meio da *ConnectionFactory*. Como resultado, ganhei confiança para aplicar TDD + DAO em projetos maiores (ex.: CRUD-pessoa) e percebi que dominar fundamentos de sintaxe e POO acelera a curva em frameworks avançados como Spring.

5.1 ARTEFATOS DO PROJETO

FIGURA 27 - SAÍDA JUNIT EXIBINDO 42 CASOS DE TESTES APROVADOS



FONTE: O autor (2025).

QUADRO 01 – TRECHO DE CÓDIGO: CONNECTIONFACTORY (JAVA)

```
package bancorrw.dao;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
```

```

import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class ConnectionFactory {
    private static Properties properties;

    static {
        try {
            readProperties();
        } catch (IOException e) {
            throw new ExceptionInInitializerError(e);
        }
    }

    private ConnectionFactory() {
    }

    public static Connection getConnection() throws SQLException {
        String url = properties.getProperty("db.url");
        String user = properties.getProperty("db.user");
        String password = properties.getProperty("db.password");
        return DriverManager.getConnection(url, user, password);
    }

    private static void readProperties() throws IOException {
        properties = new Properties();
        String filePath =
"C:\\Users\\a371132\\OneDrive\\Documentos\\UFPR\\INTRO\\trabalhoFinal\\trabalhoFinal\\INTRO-TrabalhoFinal-BancoRRW_\\INTRO-TrabalhoFinal-BancoRRW_\\DataBase.properties";
        System.out.println("Loading properties file from: " + filePath);
        try (FileInputStream fis = new FileInputStream(filePath)) {
            properties.load(fis);
        }
    }
}

```

FONTE: O autor (2025).

QUADRO 02 – TRECHO DE CÓDIGO: TESTEBANCORRW (JAVA)

```

@Test

public void t04criarContaCorrenteSaldoZero() throws Exception{

```

```

        Cliente cliente = new
Cliente(-1,"Marcelo","0886",LocalDate.of(1995,2,3),"111");

        ContaCorrente c = new ContaCorrente(1000,0.05,-1, cliente, 0);

        double saldo = c.getSaldo();

        assertEquals("Era esperado um saldo zerado para conta criada.",0.0,
saldo,0.0);

    }

    @Test

    public void t05criaContaCorrenteComSaldo2000() throws Exception {

        Cliente cliente = new Cliente(-1, "Rafael", "333",
LocalDate.of(2000, Month.MARCH, 1),"111");

        ContaCorrente conta = new ContaCorrente(1000,0.05,-1, cliente,
2000);

        assertEquals(conta,cliente.getContaCorrente());

        assertEquals(cliente.getContaCorrente().getSaldo(),2000.0,0.0);

    }

    @Test

    public void t06manipulaContaCorrenteDepositar50() throws Exception {

        Cliente cliente = new
Cliente(-1,"Marcelo","0886",LocalDate.of(1995,2,3),"111");

        ContaCorrente c = new ContaCorrente(1000,0.05,-1, cliente, 0);

        c.deposita(50);

        double saldo = c.getSaldo();

        assertEquals(50.0, saldo,0.0);

    }

    @Test

    public void t07manipulaContaCorrenteDepositar100Deposita20Saca60()
throws Exception {

```

```

        Cliente cliente = new
Cliente(-1,"Marcelo","0886",LocalDate.of(1995,2,3),"111");

        ContaCorrente c = new ContaCorrente(1000,0.05,-1, cliente, 0);

        c.deposita(100);

        assertEquals(100.0, c.getSaldo(),0.0);

        c.deposita(20);

        assertEquals(120.0, c.getSaldo(),0.0);

        c.saca(80);

        assertEquals(40.0, c.getSaldo(),0.0);

    }

    @Test

    public void t08manipulaContaCorrenteDepositar100Deposita20Saca1000()
throws Exception {

        Cliente cliente = new
Cliente(-1,"Marcelo","0886",LocalDate.of(1995,2,3),"111");

        ContaCorrente c = new ContaCorrente(1000,0.05,-1, cliente, 0);

        c.deposita(100);

        assertEquals(100.0, c.getSaldo(),0.0);

        c.deposita(20);

        assertEquals(120.0, c.getSaldo(),0.0);

        c.saca(1000);

        assertEquals(-880.0, c.getSaldo(),0.0);

    }

```

FONTE: O autor (2025).

6 DISCIPLINA: BD – BANCO DE DADOS

A disciplina de Banco de Dados estruturou a passagem do modelo conceitual ao modelo físico, com atenção à integridade, consistência e desempenho no armazenamento de dados. O trabalho avaliativo foi dividido em duas frentes complementares. Na primeira, concentrei-me na modelagem, levantamento de entidades e atributos, identificação de relacionamentos e cardinalidades, definição de chaves e regras de negócio e elaboração do diagrama entidade-relacionamento estendido (EER). O modelo foi revisto à luz das formas normais, com especial cuidado para evitar redundâncias e anomalias de inserção/atualização, e acompanhado de um dicionário de dados com nomenclatura e descrições padronizadas, favorecendo a comunicação entre análise e desenvolvimento.

Na segunda frente, realizei a especificação e implementação em SQL, transformando o modelo em um esquema relacional completo. O conjunto DDL abrange chaves primárias e estrangeiras, restrições *UNIQUE* e *CHECK*, tipos de dados adequados e políticas de nulidade. Para dados de exemplo e cenários de uso, desenvolvi scripts DML (*INSERT/UPDATE/DELETE*) e consultas *SELECT* com junções, filtros e agregações que representam relatórios típicos do domínio modelado. Em consultas mais custosas, avaliei a criação de índices quando havia benefício comprovado em leitura, anotando o impacto esperado sobre escrita. A validação foi feita no SGBD: testei integridade referencial, consistência dos resultados e comportamento transacional (*COMMIT/ROLLBACK*). Os *scripts* foram organizados por ordem de aplicação, de modo que pudessem ser re-executados com segurança em ambientes limpos.

Esse trabalho de Banco de Dados integrou-se diretamente às demais disciplinas do curso. As definições conceituais dialogaram com MAG1 e MAG2, que forneceu insumos e nomenclatura coerente para o desenho lógico. Do ponto de vista de implementação, as tabelas e restrições deram base à camada de persistência usada em INTRO e AAP (acesso via JDBC/DAO), tornando claras as regras de negócio no nível do esquema. Em WEB1 e WEB2, as validações de formulário e mensagens de erro foram alinhadas às restrições do banco, reduzindo divergências entre interface e dados. Para TEST, as consultas e os dados de exemplo funcionaram como oráculos, permitindo comprovar regras, montar cenários e

automatizar verificações. Finalmente, na perspectiva de processo, os *scripts* passaram pelo controle de versão e passaram a compor os artefatos que podem ser executados em diferentes ambientes, facilitando a automação discutida em INFRA.

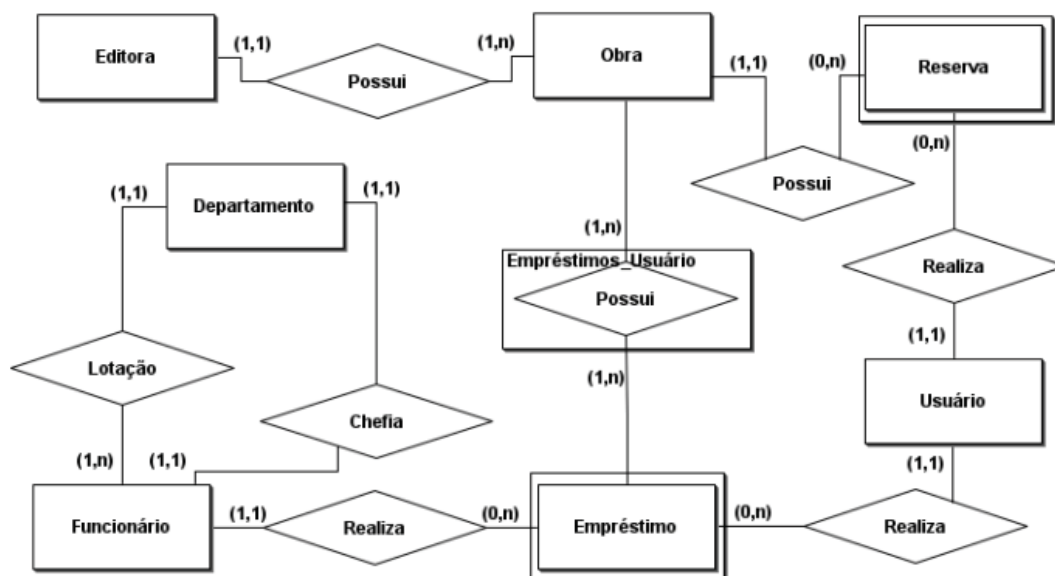
Como reflexões do projeto, três aprendizados se destacam. Primeiro, modelar antes de codificar reduz retrabalho, ajustes feitos no EER tendem a ser muito menos custosos do que correções realizadas após a criação de tabelas e dados. Segundo, proteger a integridade na fonte - por meio de chaves e restrições bem definidas - simplifica o código de aplicação, pois impede estados inválidos já no nível do SGBD. Terceiro, medir antes de otimizar é essencial, já que índices devem responder a consultas reais e ser acompanhados de comentários/justificativas, evitando sobrecarga desnecessária em operações de escrita. Entre os desafios, ressalto a modelagem de relacionamentos N×N (com tabelas associativas e chaves compostas), as decisões entre 1×1 e atributos opcionais (avaliando impacto de cardinalidade e custo de junções), e a escolha de políticas de exclusão (*RESTRICT* ou *CASCADE*) conforme a regra de negócio. No geral, o resultado foi um esquema coerente, implementado com *scripts* claros e re-executáveis e acompanhado de consultas representativas, que serviram de base para integrações com as demais disciplinas e deram maior previsibilidade às entregas do curso.

6.1 ARTEFATOS DO PROJETO

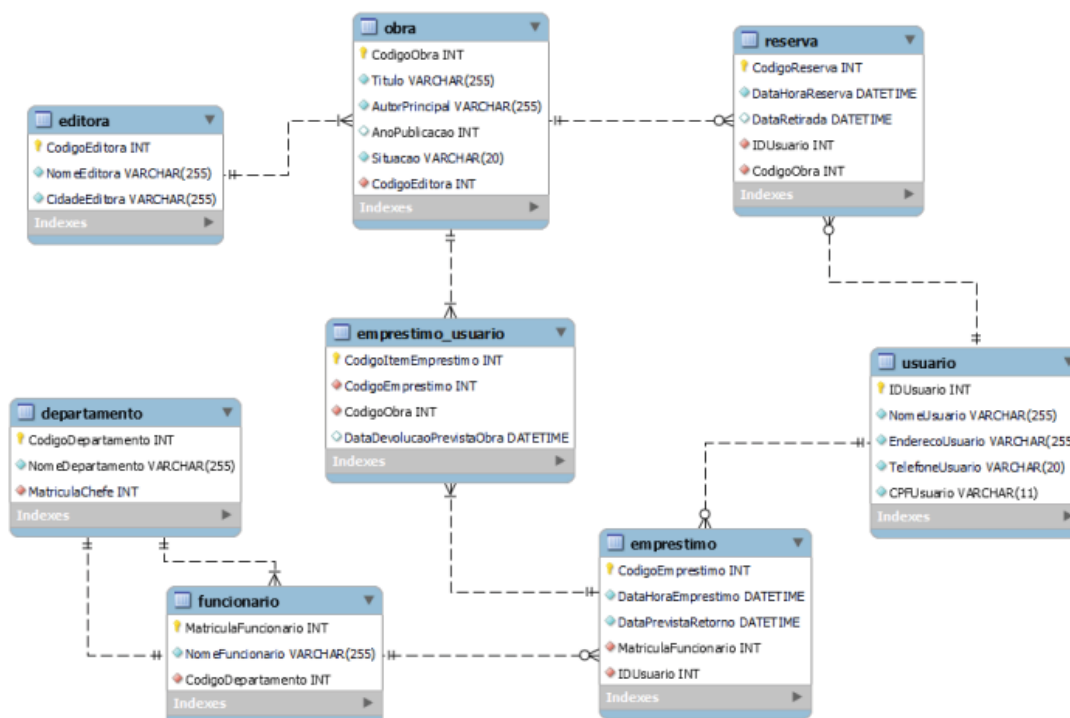
FIGURA 28 - MODELO ENTIDADE-RELACIONAMENTO CONCEITUAL E DIAGRAMA

Questão 1 – modelagem de dados

1) Modelo Entidade-relacionamento CONCEITUAL



2) Modelo Lógico - Diagrama de Entidade Relacionamento (tabelas)

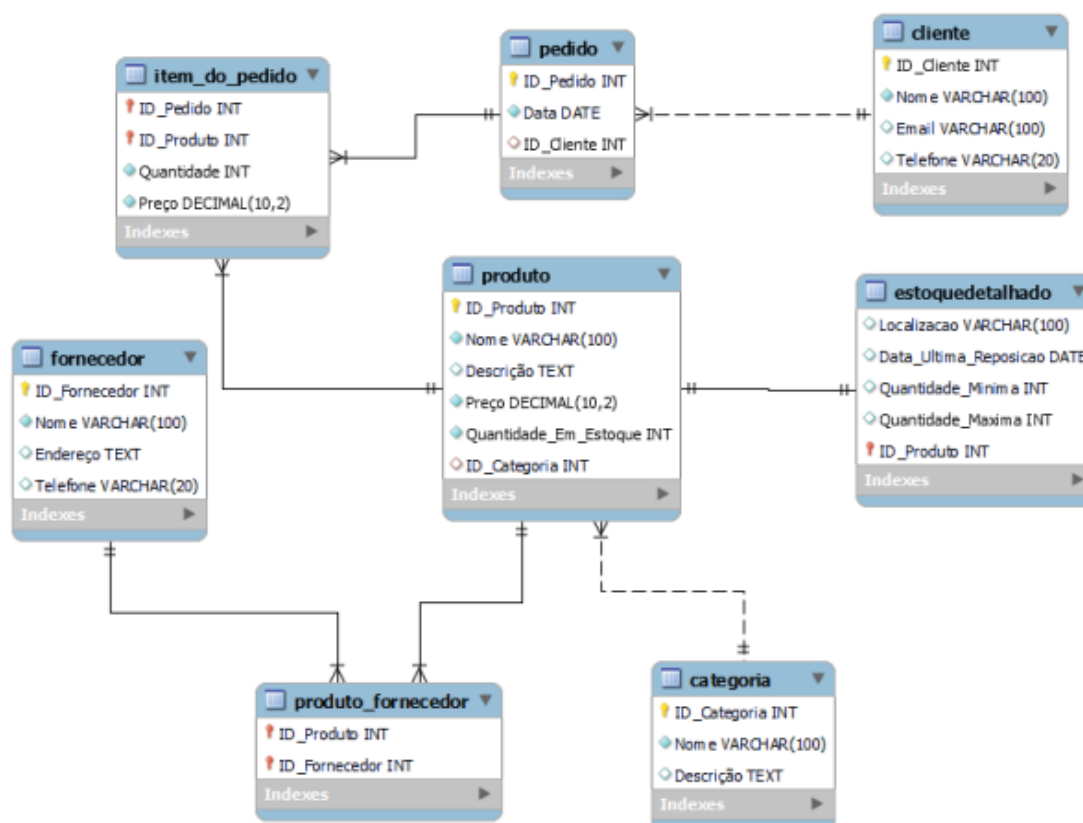


FONTE: O autor (2025).

FIGURA 29 - MODELO LÓGICO (EER) E DESCRIÇÃO DO MODELO

Questão 2 – especificação e implementação em SQL

Modelo Lógico (EER)



Descrição do Modelo EER para um Sistema de Controle de Estoque

O sistema de controle de estoque é composto pelas seguintes entidades principais:

1. **Produto**
2. **Categoria**
3. **Fornecedor**
4. **Cliente**
5. **Pedido**
6. **Item do Pedido**
7. **EstoqueDetalhado**

Entidades e Relacionamentos

1. **Produto**
 - ID_Produto (PK)
 - Nome
 - Descrição
 - Preço
 - Quantidade_Em_Estoque
 - ID_Categoria (FK)
2. **Categoria**
 - ID_Categoria (PK)
 - Nome
 - Descrição
3. **Fornecedor**
 - ID_Fornecedor (PK)
 - Nome
 - Endereço
 - Telefone
4. **Cliente**
 - ID_Cliente (PK)
 - Nome
 - Email
 - Telefone
5. **Pedido**
 - ID_Pedido (PK)
 - Data
 - ID_Cliente (FK)
6. **Item do Pedido**
 - ID_Pedido (PK, FK)
 - ID_Produto (PK, FK)
 - Quantidade
 - Preço
7. **EstoqueDetalhado**
 - ID_Produto (PK, FK)
 - Localizacao
 - Data_Ultima_Reposicao
 - Quantidade_Minima
 - Quantidade_Maxima

Relacionamentos

1. **Categoria e Produto (1:N)**
 - Uma categoria pode ter vários produtos, mas cada produto pertence a uma única categoria.
 - Exemplo: Categoria (Eletrônicos) -> Produto (Celular, Laptop)
2. **Fornecedor e Produto (N:M)**
 - Um produto pode ser fornecido por vários fornecedores, e um fornecedor pode fornecer vários produtos.

- Exemplo: Fornecedor (ABC Eletrônicos) -> Produto (Celular, Laptop);
Fornecedor (XYZ Eletrônicos) -> Produto (Laptop, Tablet)
- Tabela intermediária (Associativa): **Produto_Fornecedor**
- 3. **Cliente e Pedido (1:N)**
 - Um cliente pode fazer vários pedidos, mas cada pedido é feito por um único cliente.
 - Exemplo: Cliente (João) -> Pedido (#001, #002)
- 4. **Pedido e Item do Pedido (1:N)**
 - Um pedido pode ter vários itens de pedido, mas cada item de pedido pertence a um único pedido.
 - Exemplo: Pedido (#001) -> Item do Pedido (Produto A, Produto B)
- 5. **Produto e Item do Pedido (1:N)**
 - Um item do pedido pertence a um único produto, mas um produto pode estar em vários itens de pedido.
 - Exemplo: Produto (Celular) -> Item do Pedido (#001, #002)
- 6. **Produto e EstoqueDetalhado (1:1)**
 - Cada produto terá exatamente uma entrada em **EstoqueDetalhado**, e cada entrada em **EstoqueDetalhado** está associada a um único produto.

Descrições e Constraints de Negócio

- **Produto:** Deve sempre ter uma categoria associada e uma quantidade em estoque não negativa.
- **Categoria:** Serve para agrupar produtos por similaridade.
- **Fornecedor:** Pode fornecer vários produtos, e um produto pode ser fornecido por múltiplos fornecedores.
- **Cliente:** Realiza pedidos e cada pedido é vinculado a um cliente específico.
- **Pedido:** Contém informações sobre a data e o cliente que o realizou.
- **Item do Pedido:** Representa os produtos que compõem um pedido, com a quantidade e o preço unitário no momento da compra.
- **Produto_Fornecedor:** Relaciona produtos com seus fornecedores.
- **EstoqueDetalhado:** Cada produto deve ter uma única entrada com detalhes de estoque, incluindo localização e informações de reposição.

FONTE: O autor (2025).

FIGURA 30 - CONTEÚDO DAS TABELAS

a)

ID_Categoria	Nome	Descrição
1	Eletrônicos	Dispositivos eletrônicos e acessórios
2	Roupas	Vestuário e acessórios
3	Alimentos	Comidas e bebidas
4	Móveis	Móveis e decoração
5	Brinquedos	Brinquedos e jogos

ID_Cliente	Nome	Email	Telefone
1	João Silva	joao@example.com	9999-9999
2	Maria Oliveira	maria@example.com	8888-8888
3	Carlos Santos	carlos@example.com	7777-7777
4	Ana Pereira	ana@example.com	6666-6666
5	Paulo Lima	paulo@example.com	5555-5555

ID_Produto	Localizacao	Data_Ultima_Reposicao	Quantidade_Minima	Quantidade_Maxima
1	Depósito A	2024-06-01	10	100
2	Depósito B	2024-06-02	5	50
3	Depósito C	2024-06-03	20	200
4	Depósito D	2024-06-04	2	20
5	Depósito E	2024-06-05	15	150

ID_Fornecedor	Nome	Endereço	Telefone
1	ABC Eletrônicos	Rua A, 123	1111-1111
2	XYZ Eletrônicos	Rua B, 456	2222-2222
3	Fabrica de Roupas	Rua C, 789	3333-3333
4	Móveis S/A	Rua D, 101	4444-4444
5	Brinquedos Ltda	Rua E, 202	5555-5555

ID_Pedido	ID_Produto	Quantidade	Preço
1	1	2	1500.00
1	3	5	50.00
2	2	1	3000.00
3	4	1	1200.00
4	5	3	80.00

b)

ID_Produto	Nome	Descrição	Preço	Quantidade_Em_Estoque	ID_Categoria
1	Celular	Smartphone Android	1500.00	50	1
2	Laptop	Laptop com Windows 10	3000.00	30	1
3	Camiseta	Camiseta de algodão	50.00	200	2
4	Sofá	Sofá de 3 lugares	1200.00	10	4
5	Boneca	Boneca de pano	80.00	100	5

ID_Produto	ID_Fornecedor
1	1
2	1
2	2
3	3
4	4
5	5

ID_Pedido	Data	ID_Cliente
1	2024-06-01	1
2	2024-06-02	2
3	2024-06-03	3
4	2024-06-04	4
5	2024-06-05	5

FONTE: O autor (2025).

QUADRO 03 - CONTEÚDO DO SCRIPT

```
-- a)
-----
--
-- Script SQL
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE;
SET SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- b)
-----
--
-- Schema mydb (mantido apenas como comentário do forward engineering)
-- Schema sistema_estoque
```

```

DROP SCHEMA IF EXISTS `sistema_estoque`;
CREATE SCHEMA IF NOT EXISTS `sistema_estoque`
    DEFAULT CHARACTER SET utf8mb4
    COLLATE utf8mb4_0900_ai_ci;
USE `sistema_estoque`;

-- c)
-----

--
-- Table `sistema_estoque`.`cliente`
DROP TABLE IF EXISTS `sistema_estoque`.`cliente`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`cliente` (
  `ID_Cliente` INT NOT NULL,
  `Nome` VARCHAR(100) NOT NULL,
  `Email` VARCHAR(100) NULL DEFAULT NULL,
  `Telefone` VARCHAR(20) NULL DEFAULT NULL,
  PRIMARY KEY (`ID_Cliente`)
) ENGINE=InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_0900_ai_ci;

-- Table `sistema_estoque`.`produto`
DROP TABLE IF EXISTS `sistema_estoque`.`produto`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`produto` (
  `ID_Produto` INT NOT NULL,
  `Nome` VARCHAR(100) NOT NULL,
  `Descricao` TEXT NULL DEFAULT NULL,
  `Preco` DECIMAL(10,2) NOT NULL,
  `Quantidade_Em_Estoque` INT NOT NULL,
  `ID_Categoria` INT NULL DEFAULT NULL,
  PRIMARY KEY (`ID_Produto`)
) ENGINE=InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_0900_ai_ci;

-- d)
-----

--
-- Table `sistema_estoque`.`categoria`
DROP TABLE IF EXISTS `sistema_estoque`.`categoria`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`categoria` (
  `ID_Categoria` INT NOT NULL,
  `Nome` VARCHAR(100) NOT NULL,

```

```

`Descricao` TEXT NULL DEFAULT NULL,
PRIMARY KEY (`ID_Categoria`)
) ENGINE=InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- Relacionamento produto -> categoria
ALTER TABLE `sistema_estoque`.`produto`
ADD INDEX `ID_Categoria` (`ID_Categoria` ASC) VISIBLE,
ADD CONSTRAINT `produto_ibfk_1`
FOREIGN KEY (`ID_Categoria`)
REFERENCES `sistema_estoque`.`categoria` (`ID_Categoria`);

-- Table `sistema_estoque`.`estoquedetalhado`
DROP TABLE IF EXISTS `sistema_estoque`.`estoquedetalhado`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`estoquedetalhado` (
  `Localizacao` VARCHAR(100) NULL DEFAULT NULL,
  `Data_Ultima_Reposicao` DATE NULL DEFAULT NULL,
  `Quantidade_Minima` INT NULL DEFAULT NULL,
  `Quantidade_Maxima` INT NULL DEFAULT NULL,
  `ID_Produto` INT NOT NULL,
  PRIMARY KEY (`ID_Produto`),
  CONSTRAINT `fk_produto`
FOREIGN KEY (`ID_Produto`)
REFERENCES `sistema_estoque`.`produto` (`ID_Produto`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- e)
-----
--
-- Table `sistema_estoque`.`fornecedor`
DROP TABLE IF EXISTS `sistema_estoque`.`fornecedor`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`fornecedor` (
  `ID_Fornecedor` INT NOT NULL,
  `Nome` VARCHAR(100) NOT NULL,
  `Endereco` TEXT NULL DEFAULT NULL,
  `Telefone` VARCHAR(20) NULL DEFAULT NULL,
  PRIMARY KEY (`ID_Fornecedor`)
) ENGINE=InnoDB

```

```

DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- Table `sistema_estoque`.`pedido`
DROP TABLE IF EXISTS `sistema_estoque`.`pedido`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`pedido` (
  `ID_Pedido` INT NOT NULL,
  `Data` DATE NOT NULL,
  `ID_Cliente` INT NULL DEFAULT NULL,
  PRIMARY KEY (`ID_Pedido`),
  INDEX `ID_Cliente` (`ID_Cliente` ASC) VISIBLE,
  CONSTRAINT `pedido_ibfk_1`
    FOREIGN KEY (`ID_Cliente`)
    REFERENCES `sistema_estoque`.`cliente` (`ID_Cliente`)
) ENGINE=InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- f)
-----
--
-- Table `sistema_estoque`.`item_do_pedido`
DROP TABLE IF EXISTS `sistema_estoque`.`item_do_pedido`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`item_do_pedido` (
  `ID_Pedido` INT NOT NULL,
  `ID_Produto` INT NOT NULL,
  `Quantidade` INT NOT NULL,
  `Preco` DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (`ID_Pedido`, `ID_Produto`),
  INDEX `ID_Produto` (`ID_Produto` ASC) VISIBLE,
  CONSTRAINT `item_do_pedido_ibfk_1`
    FOREIGN KEY (`ID_Pedido`)
    REFERENCES `sistema_estoque`.`pedido` (`ID_Pedido`),
  CONSTRAINT `item_do_pedido_ibfk_2`
    FOREIGN KEY (`ID_Produto`)
    REFERENCES `sistema_estoque`.`produto` (`ID_Produto`)
) ENGINE=InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- g)
-----
--

```

```

-- Table `sistema_estoque`.`produto_fornecedor`
DROP TABLE IF EXISTS `sistema_estoque`.`produto_fornecedor`;
CREATE TABLE IF NOT EXISTS `sistema_estoque`.`produto_fornecedor` (
  `ID_Produto` INT NOT NULL,
  `ID_Fornecedor` INT NOT NULL,
  PRIMARY KEY (`ID_Produto`, `ID_Fornecedor`),
  INDEX `ID_Fornecedor` (`ID_Fornecedor` ASC) VISIBLE,
  CONSTRAINT `produto_fornecedor_ibfk_1`
    FOREIGN KEY (`ID_Produto`)
    REFERENCES `sistema_estoque`.`produto` (`ID_Produto`),
  CONSTRAINT `produto_fornecedor_ibfk_2`
    FOREIGN KEY (`ID_Fornecedor`)
    REFERENCES `sistema_estoque`.`fornecedor` (`ID_Fornecedor`)
) ENGINE=InnoDB
  DEFAULT CHARACTER SET = utf8mb4
  COLLATE = utf8mb4_0900_ai_ci;

-- h)
-----
--
-- Preparação para carga de dados
SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE TABLE `Categoria`;
TRUNCATE TABLE `Produto`;
TRUNCATE TABLE `Fornecedor`;
TRUNCATE TABLE `Cliente`;
TRUNCATE TABLE `Pedido`;
TRUNCATE TABLE `Item_do_Pedido`;
TRUNCATE TABLE `Produto_Fornecedor`;
TRUNCATE TABLE `EstoqueDetalhado`;

SET FOREIGN_KEY_CHECKS = 1;

-- i)
-----
--
-- Inserts de exemplo

-- Categoria
INSERT INTO `Categoria` VALUES
(1, 'Eletrônicos', 'Dispositivos eletrônicos e acessórios'),
(2, 'Roupas', 'Vestuário e acessórios'),

```

```

(3, 'Alimentos', 'Comidas e bebidas'),
(4, 'Móveis', 'Móveis e decoração'),
(5, 'Brinquedos', 'Brinquedos e jogos');

-- Produto
INSERT INTO `Produto` VALUES
(1, 'Celular', 'Smartphone Android', 1500.00, 50, 1),
(2, 'Laptop', 'Laptop com Windows 10', 3000.00, 30, 1),
(3, 'Camiseta', 'Camiseta de algodão', 50.00, 200, 2),
(4, 'Sofá', 'Sofá de 3 lugares', 1200.00, 10, 4),
(5, 'Boneca', 'Boneca de pano', 80.00, 100, 5);

-- Fornecedor
INSERT INTO `Fornecedor` VALUES
(1, 'ABC Eletrônicos', 'Rua A, 123', '1111-1111'),
(2, 'XYZ Eletrônicos', 'Rua B, 456', '2222-2222'),
(3, 'Fabrica de Roupas', 'Rua C, 789', '3333-3333'),
(4, 'Moveis S/A', 'Rua D, 101', '4444-4444'),
(5, 'Brinquedos Ltda', 'Rua E, 202', '5555-5555');

-- Cliente
INSERT INTO `Cliente` VALUES
(1, 'João Silva', 'joao@example.com', '9999-9999'),
(2, 'Maria Oliveira', 'maria@example.com', '8888-8888'),
(3, 'Carlos Santos', 'carlos@example.com', '7777-7777'),
(4, 'Ana Pereira', 'ana@example.com', '6666-6666'),
(5, 'Paulo Lima', 'paulo@example.com', '5555-5555');

-- j)
-----

--
-- Pedido
INSERT INTO `Pedido` VALUES
(1, '2024-06-01', 1),
(2, '2024-06-02', 2),
(3, '2024-06-03', 3),
(4, '2024-06-04', 4),
(5, '2024-06-05', 5);

-- Item_do_Pedido
INSERT INTO `Item_do_Pedido` VALUES
(1, 1, 2, 1500.00),
(1, 3, 5, 50.00),

```

```

(2, 2, 1, 3000.00),
(3, 4, 1, 1200.00),
(4, 5, 3, 80.00);

-- k)
-----

--
-- EstoqueDetalhado
INSERT INTO `EstoqueDetalhado` (`ID_Produto`, `Localizacao`,
`Data_Ultima_Reposicao`,
`Quantidade_Minima`, `Quantidade_Maxima`) VALUES
(1, 'Depósito A', '2024-06-01', 10, 100),
(2, 'Depósito B', '2024-06-02', 5, 50),
(3, 'Depósito C', '2024-06-03', 20, 200),
(4, 'Depósito D', '2024-06-04', 2, 20),
(5, 'Depósito E', '2024-06-05', 15, 150);

-- Relacionamentos já criados acima garantem integridade referencial.

-- Restaura modos originais
-----

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

FONTE: O autor (2025).

7 DISCIPLINA: AAP – ASPECTOS ÁGEIS DE PROGRAMAÇÃO

Na disciplina AAP, o projeto consistiu em aplicar princípios de *clean code* a um algoritmo existente de ordenação por bolha (*Bubble Sort*), conforme o enunciado: produzir uma versão mais legível e justificá-la com pelo menos três alterações. A base do trabalho foi o arquivo `BubbleSortV3.java`, no qual a lógica original foi mantida, mas refatorada para melhorar clareza, coesão e manutenção futura, sem alteração do comportamento funcional.

As principais mudanças realizadas foram: a extração do método `swap(int[] array, int index1, int index2)`, isolando a operação de troca e reduzindo duplicação dentro do laço interno; a otimização com curto-circuito via `boolean swapped`, de modo que, quando nenhuma troca ocorre em uma passada, o algoritmo encerra cedo — preservando a complexidade assintótica no pior caso, mas evitando iterações desnecessárias em listas quase ordenadas; a adoção de nomes mais descritivos em parâmetros e variáveis auxiliares (por exemplo, `index1`, `index2`, `swapped`), reduzindo a carga cognitiva na leitura; a separação de responsabilidades com o método `printArray(int[] array)` e o uso de *for-each* para impressão, tornando a apresentação do resultado independente da lógica de ordenação; e comentários sucintos e pontuais para documentar intenções (como a razão do curto-circuito), sem explicar o óbvio e mantendo o foco na intenção do código.

Essas intervenções reforçam princípios de desenvolvimento ágil ao favorecer legibilidade, refatoração contínua e baixa acoplabilidade. Um código claro acelera a compreensão, reduz erros e facilita incrementos pequenos com *feedback* rápido. Além disso, a extração de métodos e a nomeação consistente facilitam a testabilidade (mesmo que testes não fizessem parte do escopo deste trabalho), pois delimitam pontos de verificação e estimulam a criação de critérios de aceitação objetivos para futuras automatizações.

A integração com demais disciplinas ocorre de forma prática. Em MADS e GAP, o requisito de “três melhorias mínimas” - como por exemplo a extração de método `swap`, otimização com curto-circuito e nomes mais descritivos - funciona como critério de aceitação simples e verificável; nas disciplinas MAG, a preocupação com nomenclatura e coesão reflete a mesma busca por clareza que guia a

modelagem e a comunicação técnica. Em INTRO, a refatoração se ancora nos fundamentos de estruturas de repetição, condicionais e *arrays*. Junto a TEST, a organização por métodos e a redução de efeitos colaterais preparam o terreno para testes automatizados em ciclos futuros, quando aplicável.

Em síntese, AAP consolidou uma prática essencial de melhorar o código sem mudar sua função, tornando o artefato mais legível, mais simples de evoluir e mais fácil de validar - um passo pequeno, porém estruturante, para sustentar incrementos frequentes no contexto do desenvolvimento ágil.

7.1 ARTEFATOS DO PROJETO

QUADRO 04 - ALGORITMO DE ORDENAÇÃO POR BOLHA

```
public class BubbleSortV3 {

    // Algoritmo de ordenação por bolha otimizado
    public static void bubbleSort(int[] array) { // Mudança de nome das
variáveis para torná-las mais descritivas
        for (int i = 0; i < array.length - 1; i++) {
            boolean swapped = false;
            for (int j = 0; j < array.length - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    swap(array, j, j + 1); // Troca elementos se estiverem na
ordem errada
                    swapped = true;
                }
            }
            // Se nenhuma troca foi realizada, o array já está ordenado
            if (!swapped) {
                break;
            }
        }
    }

    // Troca dois elementos no array
    private static void swap(int[] array, int index1, int index2) { //
Mudança de nome das variáveis para torná-las mais descritivas
        // Método auxiliar para trocar elementos, melhorando a legibilidade
        int temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
    }
}
```

```
// Imprime o conteúdo do array
public static void printArray(int[] array) {
    // Impressão usando for each para maior concisão
    for (int element : array) {
        System.out.print(element + " ");
    }
    System.out.println();
}

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

FONTE: O autor (2025).

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

As disciplinas WEB1 e WEB2 evoluíram para uma mesma aplicação acadêmica de gestão de Aluno, Curso e Matrícula. Em WEB1, o foco foi consolidar fundamentos de Angular (*TypeScript*, componentes, módulos, rotas e serviços) e entregar um CRUD completo para Aluno e Curso com persistência em *LocalStorage*. Em WEB2, a aplicação passou a *full-stack*, integrando o *front-end* Angular a uma API REST em *Spring Boot* com persistência em PostgreSQL, mantendo os mesmos módulos e padrões de tela, além de Matrícula com seleção de Aluno/Curso.

No *front-end*, a arquitetura organiza o domínio em módulos e componentes (Listar/Inserir/Editar/Visualizar via modal), com serviços centralizando a lógica de dados. Em WEB1, os serviços manipulam o *LocalStorage* (listarTodos, buscarPorId, inserir, atualizar, remover); em WEB2, consomem a API com HTTP/JSON e controle de carregamento por *spinner*. O roteamento foi refinado com *nesting* e *wildcards* específicos para evitar redirecionamentos indevidos (ex.: cursos/qualquer-coisa → cursos/listar) e um coringa geral para a *home*. A interface usa *Bootstrap*, ícones e tooltips para ações, e um *Shared Module* concentra constantes e elementos reutilizáveis.

A validação de formulários foi feita com *Reactive Forms*. Em WEB1, as máscaras (ngx-mask) de CPF e data de nascimento garantiram formatação e mensagens de erro no momento certo (obrigatoriedade e tamanho 11 para CPF). Em WEB2, a validação foi aprofundada: o CPF recebe máscara dinâmica e é submetido a verificação algorítmica (classe/função dedicadas), além de normalização (remoção de caracteres não numéricos) antes do envio ao servidor. O tratamento de erros foi centralizado em uma classe de *error handling* (mensagens consistentes e redirecionamento para a rota adequada), com uso de *pipe/finalize/subscribe* e limpeza de estados em *ngOnDestroy* para evitar efeitos residuais.

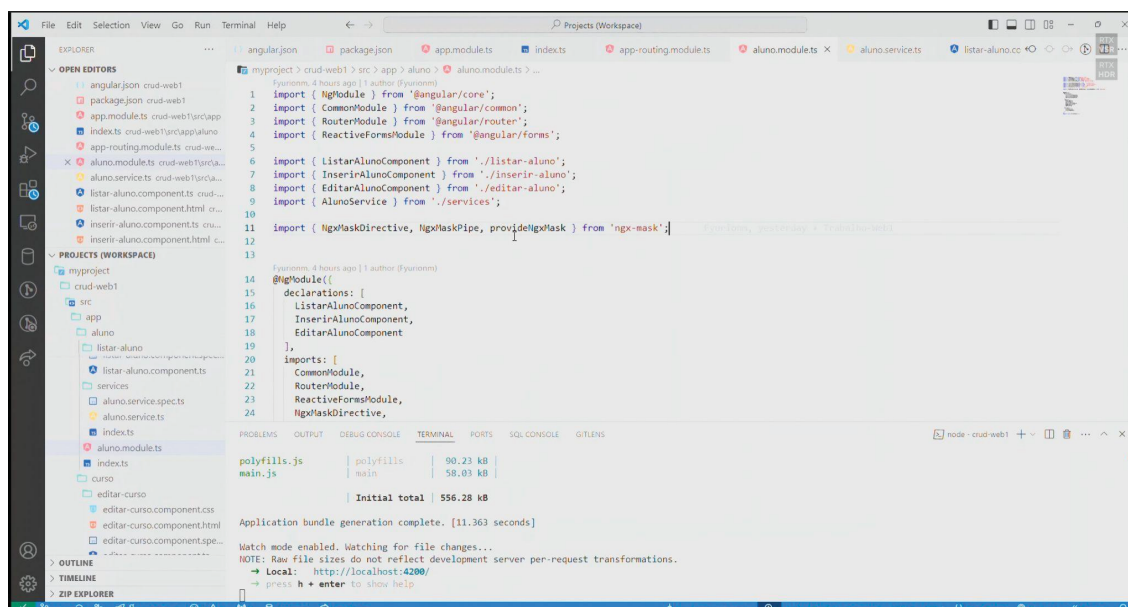
No *back-end* (WEB2), os *Controllers* expõem *endpoints GET/POST/PUT/DELETE*; *Services* concentram regras de negócio; e *Repositories* estendem *Spring Data JPA*, simplificando operações de persistência. A entidade Aluno trata *LocalDate* corretamente; os *payloads* trafegam em JSON com *headers* apropriados. Regras de integridade do banco são refletidas na experiência do usuário, a exclusão de Aluno/Curso traz confirmações em dois passos e comunica a remoção das matrículas relacionadas (exclusão em cascata). Consultas de

verificação (*SELECT*) demonstram dados persistidos com CPF normalizado, datas ISO e decimais para notas.

A construção das telas e do esquema dialogou com outras disciplinas do curso. Em MAG1 e MAG2 forneceu insumos conceituais para entidades e relacionamentos. Em BD sustentou o desenho físico e os *scripts* SQL. Já em MADS e TEST inspiraram práticas de qualidade (*feedback* rápido, validações e mensagens claras) e o controle de versão do código viabilizou integração com automação discutida em INFRA. Como síntese, os principais aprendizados foram, primeiro, centralizar validações e erros aumenta previsibilidade e reduz duplicação, segundo, modelar rotas e *wildcards* com cuidado evita desvios de navegação e terceiro, alinhar regras de integridade do banco ao comportamento da interface (ex.: cascata comunicada ao usuário) melhora a confiabilidade. O resultado é um sistema consistente e reutilizável, com UX mais amigável, serviços e componentes reaproveitados e uma base pronta para evoluções futuras.

8.1 ARTEFATOS DO PROJETO - WEB1

FIGURA 31 - FRAGMENTO DO CÓDIGO DO MÓDULO ALUNO



FONTE: O autor (2025).

FIGURA 32 - TELA DO EDITAR ALUNO

crud-web1

Editar Aluno

[← Voltar](#)

Nome:
Lucas B

CPF:
000.000.000-00

Email:
lucas@email.com

Data de Nascimento:
31/12/1991

[Atualizar](#)

FONTE: O autor (2025).

FIGURA 33 - TELA NOVO CURSO

crud-web1

Novo Curso

[← Voltar](#)

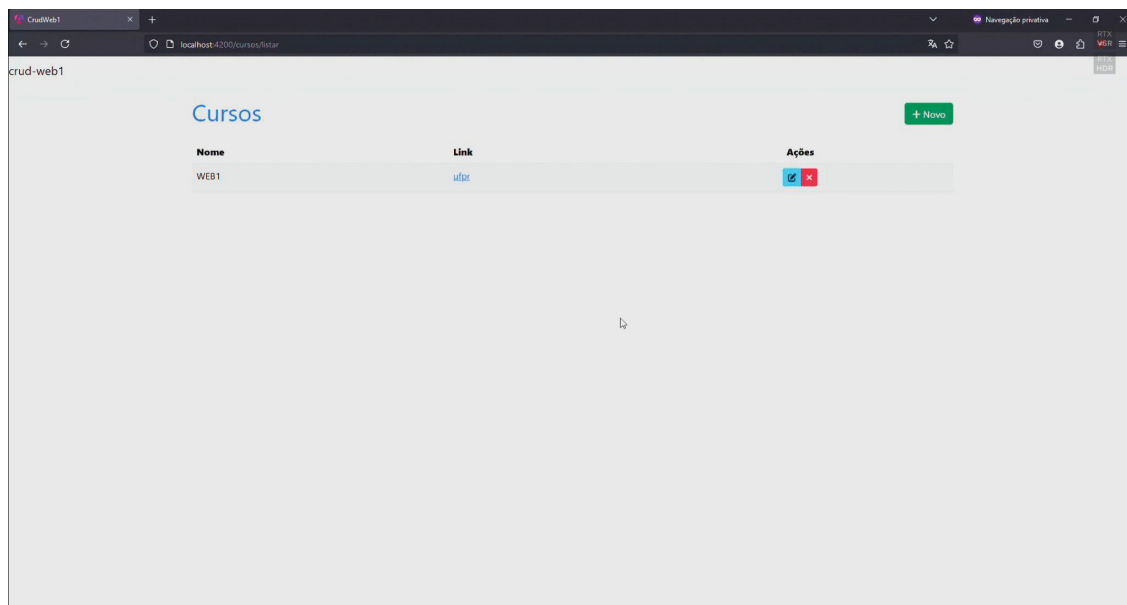
Nome:
Digite o nome do curso.

Link:
Digite o link do curso.

[Salvar](#)

FONTE: O autor (2025).

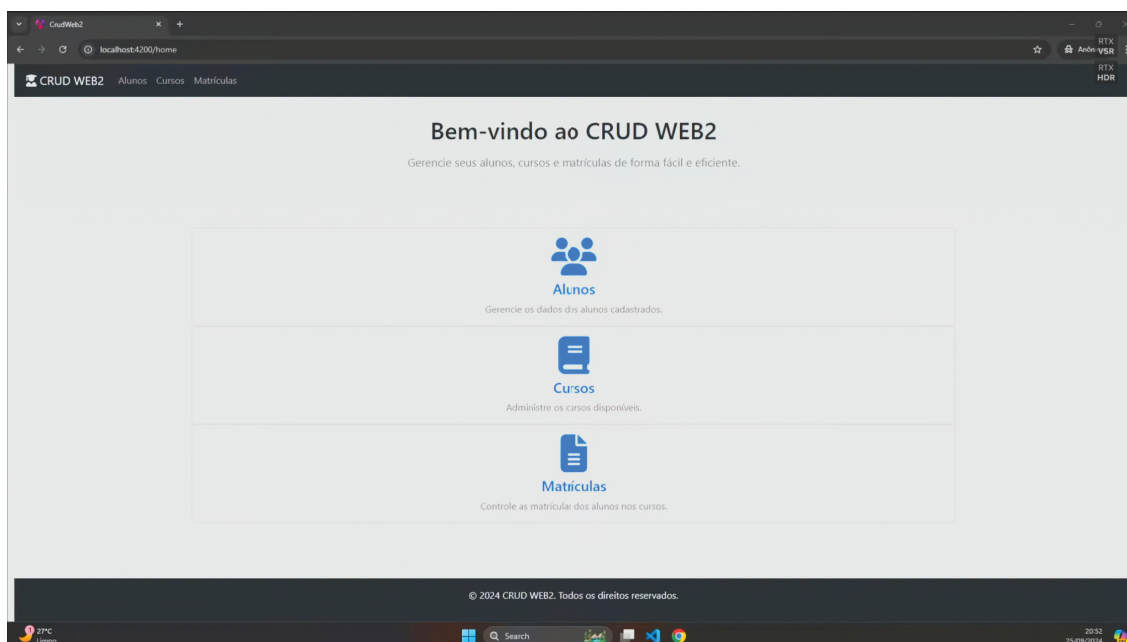
FIGURA 34 - TELA LISTAR CURSOS



FONTE: O autor (2025).

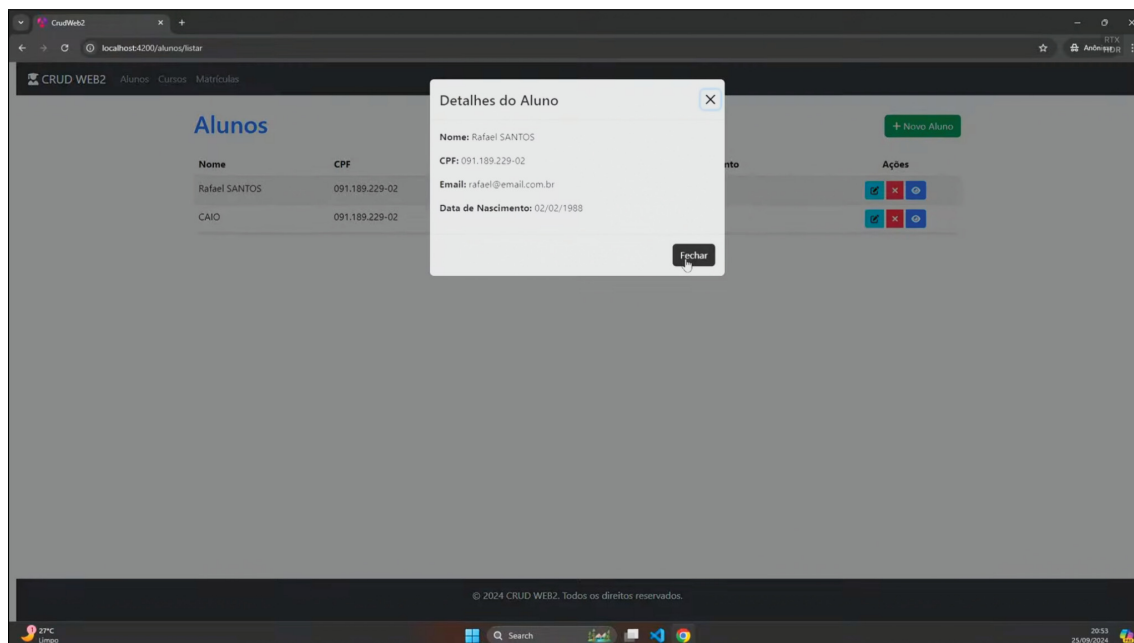
8.2 ARTEFATOS DO PROJETO - WEB2

FIGURA 35 - TELA INICIAL



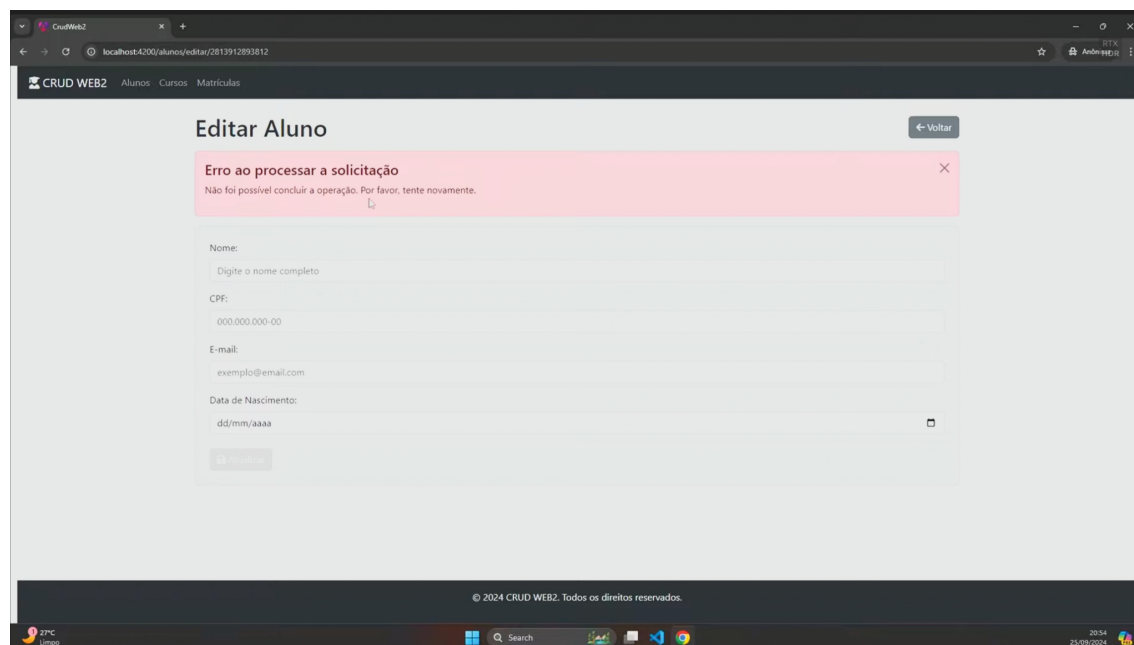
FONTE: O autor (2025).

FIGURA 36 - TELA DETALHES DO ALUNO



FONTE: O autor (2025).

FIGURA 37 - TELA EXEMPLO DE ERRO



FONTE: O autor (2025).

FIGURA 38 - FRAGMENTO DE CÓDIGO LISTAR ALUNO

The screenshot shows a Visual Studio Code interface. On the left, the Explorer sidebar displays a project structure for 'listar-aluno'. The main editor area shows the 'ListComponent.ts' file, which contains the following code:

```
// listar-aluno.component.ts
import { Component, OnInit, OnDestroy } from '@angular/core';
import { AlunoService } from '../services';
import { Aluno } from '../models/aluno.model';
import { NgModal } from '@ng-bootstrap/ng-bootstrap';
import { ModalHandlingComponent } from '../modal-handling/modal-handling.component';
import { finalize } from 'rxjs/operators';
import { ErrorHandlerService } from '../services/error-handler.service';

@Component({
  selector: 'app-listar-aluno',
  templateUrl: './listar-aluno.component.html',
  styleUrls: ['./listar-aluno.component.css']
})
export class ListarAlunoComponent implements OnInit, OnDestroy {
  alunos: Aluno[] = [];
  loading = false;
  showModal = false;
  private spinnerTimeout: any;
  successMessage: string | null = null;

  constructor(
    private alunoService: AlunoService,
    private modalService: NgModal,
    public errorHandlerService: ErrorHandlerService
  ) {}

  ngOnInit(): void {
    this.listarAlunos();
  }

  ngOnDestroy(): void {
    if (this.spinnerTimeout) {
      clearTimeout(this.spinnerTimeout);
    }
  }
}
```

FONTE: O autor (2025).

FIGURA 39 - FRAGMENTO CÓDIGO ALUNO CONTROLLER (BACKEND)

The screenshot shows an IDE window titled "AlunoController.java". The code defines several REST endpoints for managing students:

- @GetMapping("/{id}"): A GET endpoint that returns a `ResponseEntity<Aluno>`. It uses `Optional<Aluno>` to handle cases where the student might not exist.**
- @PostMapping: A POST endpoint that takes a `RequestEntity<Aluno>` and returns a `ResponseEntity<Aluno>`. It calls `alunoService.salvar(aluno)`.**
- @PutMapping("/{id}"): A PUT endpoint that takes a `RequestEntity<Aluno>` and returns a `ResponseEntity<Aluno>`. It checks if the student exists before updating it with `aluno.setId(id)`.**
- @DeleteMapping("/{id}"): A DELETE endpoint that takes a `RequestEntity<Void>` and returns a `ResponseEntity<Void>`. It calls `alunoService.excluirPorId(id)`.**

The IDE interface includes a sidebar with project files, a main editor area, and a bottom panel with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, GITLens, COMMENTS, and SQL CONSOLE. The terminal tab is active, displaying Hibernate log messages.

FONTE: O autor (2025).

QUADRO 05 - SCRIPT DA CRIAÇÃO DE TABLE (SQL)

```
CREATE DATABASE crud web backend;
```

```
CREATE TABLE tb_aluno (  
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    nome VARCHAR(255) NOT NULL,  
    cpf VARCHAR(11) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    data_nascimento DATE NOT NULL  
);  
  
CREATE TABLE tb_curso (  
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    nome VARCHAR(255) NOT NULL,  
    link VARCHAR(255)  
);  
  
CREATE TABLE tb_matricula (  
    id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    aluno_id INTEGER NOT NULL,  
    curso_id INTEGER NOT NULL,  
    data_matricula DATE NOT NULL,  
    nota NUMERIC(5,2),  
    CONSTRAINT fk_aluno  
        FOREIGN KEY (aluno_id)  
            REFERENCES tb_aluno(id)  
            ON DELETE CASCADE,  
    CONSTRAINT fk_curso  
        FOREIGN KEY (curso_id)  
            REFERENCES tb_curso(id)  
            ON DELETE CASCADE  
);
```

FONTE: O autor (2025).

9 DISCIPLINA: UX – UX NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

O projeto de UX consistiu na concepção e prototipação, em Figma, de um site institucional para a fisioterapeuta Poliane Faccin, com foco em apresentar serviços, transmitir credibilidade e facilitar o agendamento de avaliações. A solução priorizou clareza e acolhimento, adotando *layout* minimalista, hierarquia visual nítida e navegação direta, de modo a reduzir esforço cognitivo e orientar o usuário rapidamente às ações principais (conhecer serviços e agendar). Cores suaves (bege/coral) e tipografia legível foram escolhidas para reforçar a identidade profissional e a sensação de confiança, enquanto a organização em “*cards*” tornou a exploração dos serviços mais rápida e previsível. O resultado é um protótipo navegável que cobre páginas-chave como Página inicial, Quem sou eu, Serviços, Perguntas frequentes e Agende sua avaliação.

O processo foi iterativo e ancorado em evidências. A cliente avaliou a navegação e a apresentação visual, confirmando a intuitividade e a aderência à sua marca. A partir do *feedback*, foram aplicadas melhorias tangíveis - ajuste de caixa-alta e tamanho em botões de cabeçalho, remoção de negritos desnecessários, revisão de espaçamentos e supressão de sombreados para manter o visual mais *clean*. Esse ciclo curto de prototipação → validação com a cliente → refino reduziu retrabalho e elevou a confiança nas decisões de *design*, antecipando questões que, se deixadas para o desenvolvimento, custariam mais caro para corrigir.

A disciplina mostrou-se estratégica para o desenvolvimento ágil com princípios de UX (clareza, consistência, feedback e foco na tarefa) foram traduzidos em critérios de aceitação e “*Definition of Ready*” para itens de *backlog*. O protótipo serviu como artefato compartilhado nas discussões com *stakeholders*, acelerando alinhamento e diminuindo ambiguidades - prática alinhada a ciclos curtos de inspeção e adaptação.

A integração com as demais disciplinas ocorreu da seguinte forma: em MADS, o protótipo alimentou a priorização do *Product Backlog* (por exemplo, páginas e fluxos críticos primeiro) e serviu de base para *Reviews*, nas quais a cliente validou incrementos de valor; em WEB1 e WEB2–2, as telas e componentes do protótipo foram transpostos para o *front-end* (Angular/*Bootstrap*) e para o *back-end* (*Spring*), guiando a padronização de formulários, mensagens de erro e máscaras (como validação de campos e *feedback* imediato ao usuário); em INTRO e BD, os

fluxos mapeados (por exemplo, “Agende sua avaliação”) ajudaram a esclarecer entidades e atributos necessários, influenciando a modelagem de dados e a definição de *endpoints* de suporte; e, em TEST, as microinterações previstas (estados de erro/sucesso e mensagens claras) orientaram casos de teste de interface e validações de entrada, favorecendo *design for testability* e a prevenção de defeitos desde a concepção.

Em síntese, o projeto de UX funcionou como “fio condutor” de usabilidade e valor ao usuário, diminuindo incertezas e orientando decisões de produto e de tecnologia ao longo do desenvolvimento.

9.1 ARTEFATOS DO PROJETO

FIGURA 40 - PÁGINA INICIAL PROTÓTIPO FIGMA



FONTE: O autor (2025).

FIGURA 41 - PÁGINA QUEM SOU EU


POLIANE FACCIN
FISIOTERAPEUTA

[INÍCIO](#)
[QUEM SOU EU](#)
[NOSSO ESPAÇO](#)
[SERVIÇOS](#)
[PERGUNTAS FREQUENTES](#)
[AGENDE SUA AVALIAÇÃO](#)

QUEM SOU EU



Olá! Eu sou Poliane Faccin, fisioterapeuta apaixonada por proporcionar mais qualidade de vida e bem-estar aos meus pacientes. Acredito que todos mereçam o melhor tratamento e a melhor estrutura, desde atletas e praticantes ou não de atividades físicas. Toda habilidade pode ser treinada.

Formação Acadêmica:

- Graduação em Fisioterapia - Universidade Federal de Santa Catarina (UFSC) - Conclusão: 2012
- Pós-graduação em Fisioterapia Ortopédica e Traumatológica - Universidade de São Paulo (USP) - Conclusão: 2015

Cursos e Certificações Complementares:

- Curso de Terapia Manual e Manipulação Articular - Instituto Brasileiro de Terapia Manual - Carga Horária: 80 horas - Conclusão: 2017

Certificação em Pilates Clínico - Instituto de Pilates Terapêutico

- Carga Horária: 60 horas
- Conclusão: 2018

Curso de Reabilitação Funcional Avançada - Reabilitação Brasil

- Carga Horária: 100 horas
- Conclusão: 2020



FONTE: O autor (2025).

FIGURA 42 - PÁGINA NOSSO ESPAÇO

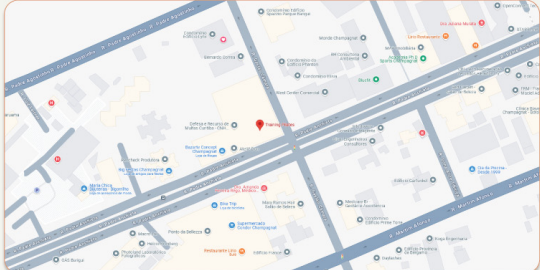

POLIANE FACCIN
FISIOTERAPEUTA

[INÍCIO](#)
[QUEM SOU EU](#)
[NOSSO ESPAÇO](#)
[SERVIÇOS](#)
[PERGUNTAS FREQUENTES](#)
[AGENDE SUA AVALIAÇÃO](#)

NOSSO ESPAÇO









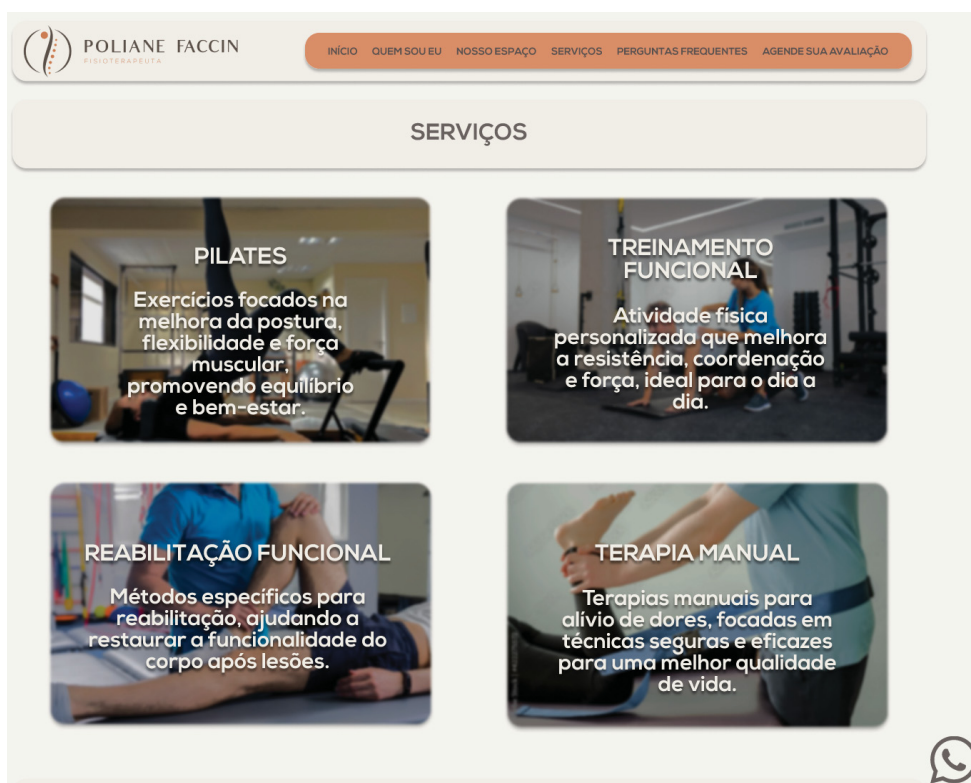
 Champagnat Office & Residence - R. Padre Anchieta, 2540 - Sala 1501 - Bigorriho, Curitiba - PR, 80730-000

Em frente ao Supermercado Condor Champagnat



FONTE: O autor (2025).

FIGURA 43 - PÁGINA SERVIÇOS



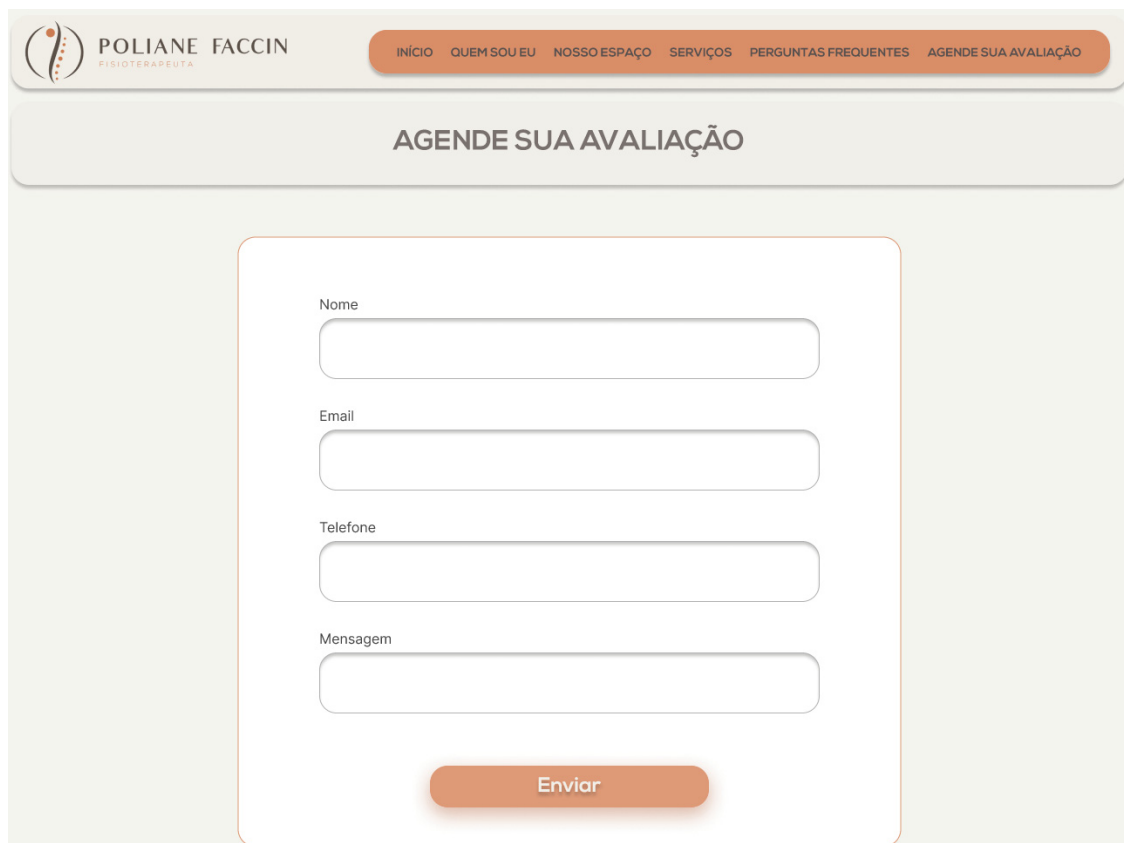
FONTE: O autor (2025).

FIGURA 44 - PÁGINA PERGUNTAS FREQUENTES



FONTE: O autor (2025).

FIGURA 45 - PÁGINA AGENDE SUA AVALIAÇÃO



POLIANE FACCIN
FISIOTERAPEUTA

INÍCIO QUEM SOU EU NOSSO ESPAÇO SERVIÇOS PERGUNTAS FREQUENTES **AGENDE SUA AVALIAÇÃO**

AGENDE SUA AVALIAÇÃO

Nome

Email

Telefone

Mensagem

Enviar

FONTE: O autor (2025).

FIGURA 46 - RODAPÉ

FONTE: O autor (2025).

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

Os trabalhos de Desenvolvimento Mobile I e II foram conduzidos como um ciclo contínuo de aprendizado incremental. Em MOB1, o projeto FinApp implementou um aplicativo Android simples voltado a apoiar a organização financeira pessoal. A solução apresentou um *dashboard* com navegação direta e duas funcionalidades centrais: cadastro de operações (marcadas como débito ou crédito, com breve descrição e valor) e extrato para a listagem das operações registradas. A persistência não fez parte do escopo, já que os dados foram mantidos apenas em memória durante a execução, priorizando a prática de *layout*, componentes de interface e fluxo entre *activities*. Esse recorte possibilitou treinar a estruturação de telas, o uso de *ListView* com adapter customizado, a navegação por *intents* e o tratamento básico de entradas, com foco em clareza e legibilidade do código.

Em MOB2, a complexidade aumentou com a introdução do consumo de serviços REST. O trabalho integrou a HP-API, disponibilizando um menu que direciona para consultas específicas: buscar personagem por ID, listar professores e listar estudantes por casa. As requisições foram implementadas com *Retrofit* e *Gson*, utilizando corrotinas para executar chamadas de rede fora da thread principal com *lifecycleScope* e mudança de contexto para processamento em *Dispatchers.IO*, retornando à interface para exibir os resultados. Quando a busca por nome não estava disponível no *endpoint*, o app carregou a lista completa e aplicou filtro local em memória. O tratamento de falhas foi feito de forma simples, com *try/catch* e mensagens informativas na tela. Não foram adotadas camadas ou mecanismos adicionais de armazenamento, mantendo o foco na integração de rede e na atualização segura da UI.

A importância desse eixo no desenvolvimento ágil de *software* foi dupla. Primeiro, pela prática de entregas incrementais: em MOB1, um núcleo funcional enxuto de UI e navegação; em MOB2, incrementos que acrescentam integração com serviço externo e concorrência sem bloquear a interface. Segundo, pelo cuidado com qualidade incorporada: entradas tratadas de forma básica, mensagens claras ao usuário e organização do código suficiente para facilitar a evolução, sempre alinhada ao objetivo de cada incremento.

A modelagem e o raciocínio de BD auxiliaram a delimitar entidades e atributos de forma simples (por exemplo, a estrutura de transações e a de personagens). As convenções REST e o desenho de camadas discutidos em WEB1 e WEB2 influenciaram a definição de *endpoints* e o tratamento das respostas. Conceitos de INTRO (sintaxe, POO e exceções) foram aplicados no controle de fluxo e na organização do código. O trabalho de UX orientou decisões de *layout* e microinterações (navegação por botões e mensagens de *feedback*), mantendo a interface objetiva e coerente com os objetivos dos aplicativos.

Como resultado, o capítulo Mobile consolidou um *pipeline* de aprendizado do “*layout* ao dado remoto”: começou por UI e navegação (MOB1) e evoluiu para acesso a serviços externos com corrotinas (MOB2). Essa progressão, combinada ao ritmo de inspeção e adaptação, estabelece base sólida para releases futuros - como inclusão de persistência, listas mais ricas com componentes específicos e camadas arquiteturais - e para a reutilização de componentes nos demais projetos do curso.

10.1 ARTEFATOS DO PROJETO - MOB1

FIGURA 47 - TELA INICIAL

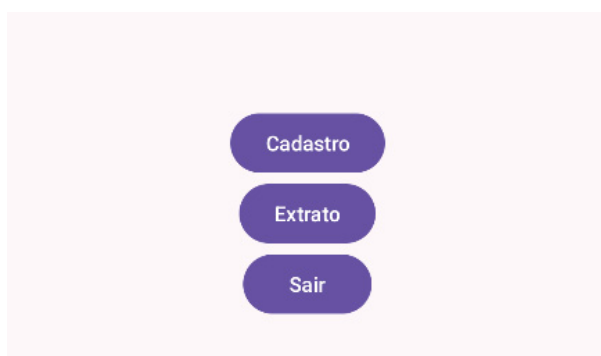


FIGURA 48 - EXEMPLO DE DÉBITO

FIGURA 49 - EXEMPLO DE CRÉDITO

FIGURA 50 - VALORES CADASTRADOS

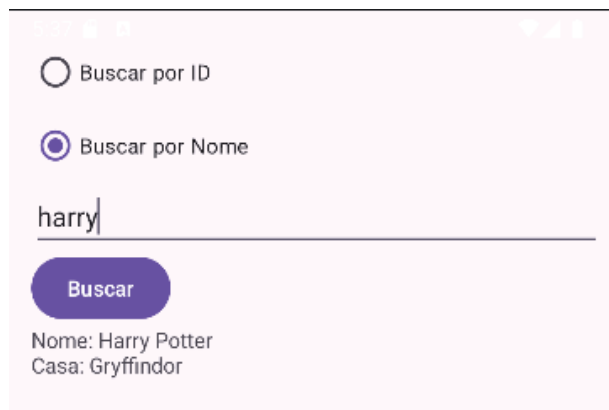
Débito	
Mercado	R\$ 399.90
Crédito	
Internet	R\$ 167.80

10.2 ARTEFATOS DO PROJETO - MOB2

FIGURA 51 - TELA INICIAL



FIGURA 52 - BUSCAR POR NOME



FONTE: O autor (2025) telas do HarryPotterAPI app.

FIGURA 53 - RESULTADO DA PESQUISA POR "HARRY"



FONTE: O autor (2025).

FIGURA 54 - LISTA DE ESTUDANTES POR CASA

☒ Gryffindor

☐ Slytherin

☐ Ravenclaw

☐ Hufflepuff

Listar Estudantes

Harry Potter
Hermione Granger
Ron Weasley
Minerva McGonagall
Rubeus Hagrid
Neville Longbottom
Ginny Weasley
Sirius Black
Remus Lupin
Arthur Weasley
Lily Potter
James Potter
Albus Dumbledore
Molly Weasley
Percy Weasley
Fred Weasley
George Weasley
Lee Jordan
Bill Weasley
Charlie Weasley
Lavender Brown
Seamus Finnigan
Parvati Patil
Nearly Headless Nick
Angelina Johnson
Alicia Spinnet
Katie Bell
Celestina Warbeck
Colin Creevey
Peter Pettigrew
Natalie McDonald
Eloise Midgen
Aberforth Dumbledore
Euan Abercrombie
Kenneth Towler
Vicky Frobisher
Geoffrey Hooper
Andrew Kirke
Jack Sloper

FONTE: O autor (2025).

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

O trabalho de INFRA teve como objetivo montar, validar e utilizar um ambiente local de controle de versões e colaboração por meio de *container*. O exercício consistiu em baixar e executar a imagem `dfwandarti/gitlab_jenkins:3` no *Docker*, criando o *container* com o nome da matrícula e publicando as portas 22, 80, 443 e 9091. Foram registrados *prints* do comando de execução e da listagem do *container* em funcionamento, evidenciando o mapeamento das portas e o *healthcheck*. Em seguida, realizou-se o acesso ao GitLab pelo usuário *root*, recuperando a senha no arquivo `/etc/gitlab/initial_root_password` dentro do *container*; o acesso via navegador foi documentado por *print screen*. Por fim, foi feito um *commit* e *push* em um projeto do Git hospedado nessa instância local: criação do arquivo `README.md`, adição ao repositório, confirmação da alteração e envio ao *remote*, com registro do *log* do Git mostrando o *hash* e a mensagem do *commit*.

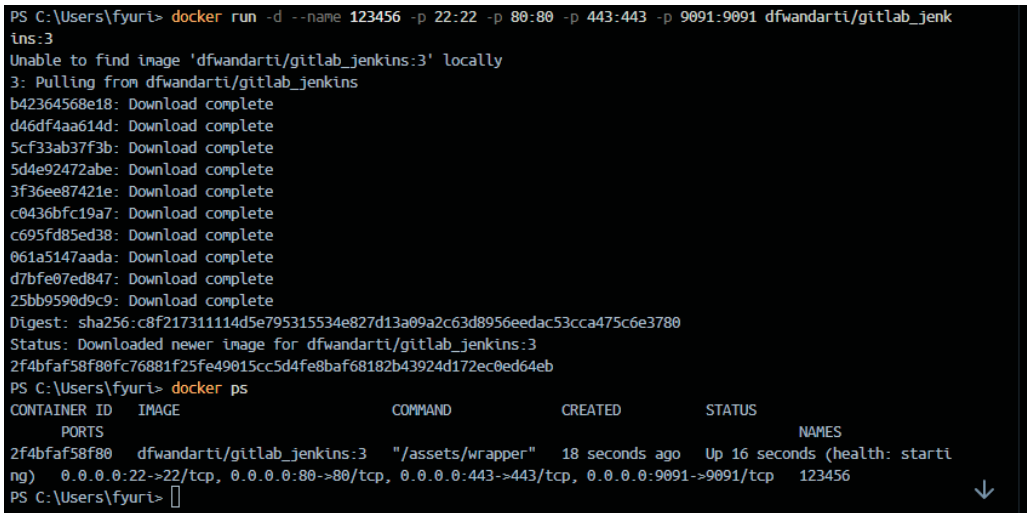
A atividade enfatiza práticas essenciais para o desenvolvimento ágil de *software*. Primeiro, reprodutibilidade do ambiente por meio de *container* - qualquer máquina capaz de rodar *Docker* consegue subir o mesmo serviço com um único comando. Segundo, rapidez de onboarding e de experimentação - a instância local permite criar e controlar versões de artefatos sem dependências externas. Terceiro, controle de mudanças - o ciclo *add/commit/push* reforça rastreabilidade, histórico e colaboração entre integrantes. A publicação das portas e a validação do acesso demonstram atenção à operacionalização do serviço (acesso por HTTP/HTTPS e canal de comunicação adicional), aproximando o estudante de rotinas cotidianas de configuração e verificação em times ágeis.

A integração com as demais disciplinas ocorre de forma direta: em MADS e GAP, a instância local apoia o ritmo de entregas incrementais e a possibilidade de trabalho colaborativo sobre o mesmo código; em WEB1, WEB2, MOB1 e MOB2, o repositório controla versões de exemplos, *scripts* e projetos, permitindo revisar *commits*, comparar mudanças e sustentar pequenas liberações frequentes; em TEST, o histórico de versões viabiliza a evolução controlada de casos de teste e a análise de regressões. Assim, o capítulo de Infraestrutura consolida o entendimento

de como ambiente, controle de versão e evidências de execução dão suporte prático ao ciclo de inspeção e adaptação que caracteriza o desenvolvimento ágil.

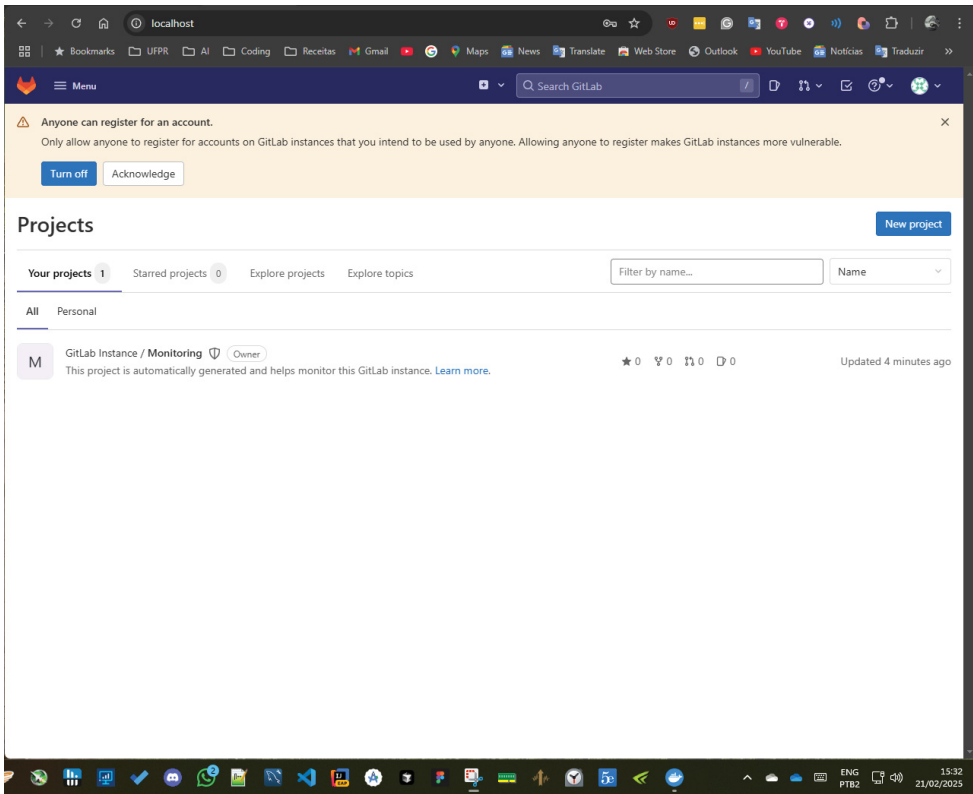
11.1 ARTEFATOS DO PROJETO

FIGURA 55 - CRIAÇÃO DE UM CONTAINER NO GITLAB



FONTE: O autor (2025).

FIGURA 56 - PUBLICAÇÃO DO GITLAB



FONTE: O autor (2025).

FIGURA 57 - PRIMEIRO COMMIT NO REPOSITÓRIO GITLAB

```

Terminal

PS C:\Users\fyuri\OneDrive\Documentos\infra> ^C
PS C:\Users\fyuri\OneDrive\Documentos\infra> New-Item README.md -ItemType File

Directory: C:\Users\fyuri\OneDrive\Documentos\infra

Mode                LastWriteTime         Length Name
----                -
-a---             21/02/2025   15:57              0 README.md

PS C:\Users\fyuri\OneDrive\Documentos\infra> touch README.md
touch: The term 'touch' is not recognized as a name of a cmdlet, function, script file, or executable program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
PS C:\Users\fyuri\OneDrive\Documentos\infra> git add README.md
PS C:\Users\fyuri\OneDrive\Documentos\infra> git commit -m "add README"
[main (root-commit) f8174d5] add README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\fyuri\OneDrive\Documentos\infra> git push -u origin main
fatal: Failed to write item to store. [0x8]
fatal: Not enough memory resources are available to process this command
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 219 bytes | 219.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To http://localhost/gitlab-instance-5cfec231/infra.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\fyuri\OneDrive\Documentos\infra> git log --oneline
f8174d5 (HEAD -> main, origin/main) add README
PS C:\Users\fyuri\OneDrive\Documentos\infra>

```

FONTE: O autor (2025).

12 DISCIPLINA: TEST – TESTES AUTOMATIZADOS

Na disciplina TEST, o exercício pediu a criação de um código que automatizasse a escrita de uma nota no serviço online *aNotepad* (pt.anoypad.com). A automação deveria definir o título exatamente como “Entrega trabalho TEST DAS 2024” e preencher o corpo da nota com nome e matrícula. O resultado esperado foi a nota criada com o título correto e o conteúdo preenchido, evidenciando que o código conseguiu acessar a página, acionar os elementos da interface e registrar as informações solicitadas.

Do ponto de vista de desenvolvimento ágil, o trabalho reforçou três aspectos práticos. Primeiro, clareza de critério de aceitação com título e corpo precisos funcionaram como *Definition of Done* objetivo (ou passa, ou não passa). Segundo, *feedback* rápido com a execução da automação fornecendo evidência imediata (nota gerada), permitindo ciclos curtos de inspeção e adaptação - ajustar *selectors*, tratar pequenos erros de digitação e repetir. Terceiro, reprodutibilidade ao encapsular o procedimento em código, o mesmo passo-a-passo pode ser executado inúmeras vezes, reduzindo variação manual e tornando o resultado verificável.

A integração com as demais disciplinas é direta. Em MADS e GAP, a atividade se encaixa no ritmo de entregas incrementais e na validação objetiva de requisitos. Em WEB1 e WEB2, dialoga com a manipulação de elementos de página e o entendimento de interações básicas de formulários. Em INTRO e AAP, reutiliza fundamentos de lógica, tratamento de erros e organização de código para estruturar a automação de forma legível. Ao lado de INFRA, a evidência da execução (nota criada) pode ter o controle de versões no repositório com os arquivos do exercício e *prints* ilustrativos, preservando histórico e rastreabilidade. Assim, TEST consolidou a ideia de que automatizar uma verificação simples, porém bem especificada, é um passo pequeno e valioso que reduz ambiguidade, encurta *feedback* e pavimenta a evolução para cenários mais complexos quando necessário.

12.1 ARTEFATOS DO PROJETO

QUADRO 06 - SCRIPT DE AUTOMAÇÃO

```
import { test, expect } from '@playwright/test';
```

```

test.beforeEach(async ({ page }) => {
  // Bloqueia arquivos .gif (incluindo query params)
  // await page.route('**/*.gif*', route => route.abort());

  // Bloqueia domínios indesejados
  await page.route('**/*', (route) => {
    const url = route.request().url();

    const blockedDomains = [
      'ads.pubmatic.com',
      '3lift.com',
      'pa.openx.net',
      'tlx.3lift.com',
    ];

    if (blockedDomains.some(domain => url.includes(domain))) {
      return route.abort();
    }

    route.continue();
  });
});

test('Escreve título e conteúdo no Anotepad', async ({ page }) => {
  await page.goto('https://pt.anotepad.com/', { waitUntil:
'domcontentloaded' });

  // Preenche os campos
  const titleInput = page.getByRole('textbox', { name: 'Título da Nota' });
  await titleInput.fill('Entrega trabalho TEST DAS 2024');

  const contentTextArea = page.getByRole('textbox', { name: 'Conteúdo da
Nota' });
  await contentTextArea.fill('Lucas Norberto Bueno dos
Santos\n202400184602');

  // Asserções
  const titleValue = await titleInput.inputValue();
  const contentValue = await contentTextArea.inputValue();
  expect(titleValue).toBe('Entrega trabalho TEST DAS 2024');
  expect(contentValue).toContain('Lucas Norberto Bueno dos Santos');
  expect(contentValue).toContain('202400184602');

  // Salvar e aguardar redirecionamento
  const saveButton = page.getByRole('button', { name: 'Salvar' });
  await Promise.all([

```

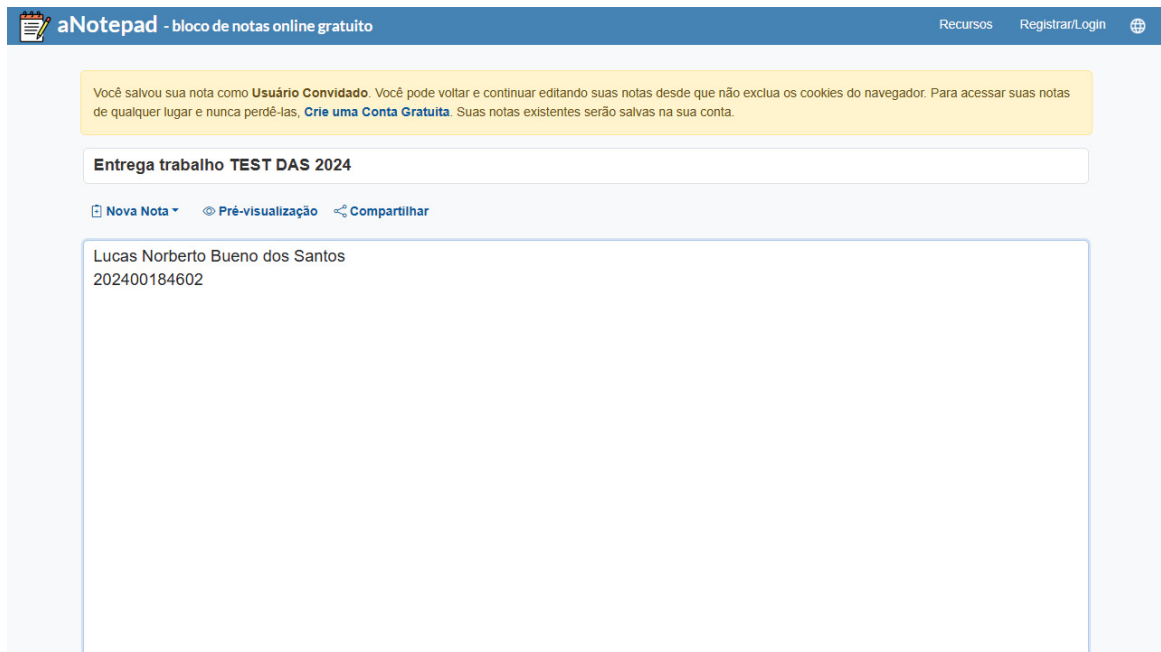
```

        saveButton.click(),
        page.waitForURL('**/notes/**', { waitUntil: 'domcontentloaded' })
    });

    // Captura a URL e tira screenshot
    console.log('Link da nota:', page.url());
    await page.screenshot({ path: 'anotepad_test_screenshot_fill.png' });
    console.log('Screenshot "anotepad_test_screenshot_fill.png" tirado para
confirmação.');
```

FONTE: O autor (2025).

FIGURA 58 - AUTOMAÇÃO PUBLICANDO OS DADOS



FONTE: O autor (2025).

13 CONCLUSÃO

Este memorial sintetiza, em formato de parecer, os projetos desenvolvidos ao longo das disciplinas e os integra conceitualmente em um único documento.

Observa-se uma progressão deliberada por incrementos. Em INTRO e AAP, foram reforçados fundamentos de programação e organização do código; MAG1 e MAG2 orientaram a modelagem e a comunicação visual de soluções; MADS e GAP1 e GAP2 amarraram prioridades, critérios de aceitação e ciclos curtos de inspeção/adaptação; WEB1 e WEB2 estruturaram a construção de interfaces e a prática de consumo de dados; UX sustentou decisões de *layout* e microinterações. Em MOB1, o foco recaiu sobre UI, navegação entre *activities* e listas com *adapter* em memória - um recorte mínimo, porém funcional. Em MOB2, o passo seguinte foi a integração com serviço REST usando corrotinas, mantendo a UI responsiva e filtrando resultados quando necessário. Nas atividades de INFRA, a execução de uma instância local de colaboração em *container Docker* e sua validação prática (execução e publicação de portas) reforçaram a reprodutibilidade e operacionalização do ambiente. A atividade de TEST demonstrou a tradução do requisito em verificação automática - geração da nota com título e conteúdo exigidos e comprovação objetiva por artefato e captura de tela.

Como conjunto, os projetos mostram domínio crescente de *feedback* rápido, rastreabilidade (controle de versão e *commits*), reprodutibilidade (ambientes containerizados) e verificação automática de resultados. Essa costura transversal - do “*layout* e lógica” ao “ambiente e evidência de execução” - traduz bem a mentalidade ágil exercitada, sendo, entregar valor em pequenos passos, medir, aprender e evoluir.

REFERÊNCIAS

BECK, K. *Extreme Programming Explained: Embrace Change*. 2. ed. Addison-Wesley, 2000.

FOWLER, M. *Continuous Integration*. ThoughtWorks/Prentice Hall, 2006.

HUMBLE, J.; FARLEY, D. *Continuous Delivery*. Addison-Wesley, 2010.

MARTIN, R. C. *Clean Code*. Prentice Hall, 2008.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. *CHI '90 Proceedings*, 1990.

SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide*. 2020.