

Universidade de Brasília

Faculdade de Tecnologia

Departamento de Engenharia de Software

GuardItens: Uma solução para o gerenciamento de ativos de TI.

Artur Seppa Reiman

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA DE SOFTWARE

Brasília, DF

2025

Artur Seppa Reiman

GuardItens: Uma solução para o gerenciamento de ativos de TI.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Henrique Gomes de Moura

Brasília, DF

2025

Artur Seppa Reiman GuardItens: Uma solução para o gerenciamento de ativos de TI./ Artur Seppa Reiman. – Brasília, DF, 2025- 101 p. : il. (algumas color.) ; 30 cm.
Orientador: Prof. Henrique Gomes de Moura
Trabalho de Conclusão de Curso – Universidade de Brasília - UnB Faculdade do Gama - FGA , 2024. – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2025.
1. . 2. Faculdade UnB Gama. I. Prof. Henrique Gomes de Moura. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. GuardItens: Uma solução para o gerenciamento de ativos de TI.
CDU

Resumo

Este trabalho propõe o desenvolvimento do GuardItens, um *web app* completo para gestão de ativos de TI destinado a empresas de médio e pequeno porte. A motivação surge da lacuna entre a importância estratégica do *ITAM*[1] e sua adoção desigual no mercado, onde PMEs frequentemente carecem de ferramentas acessíveis para gerenciar seus recursos tecnológicos. O desenvolvimento seguiu a metodologia *Lean Inception*, que permitiu alinhar requisitos técnicos com demandas do mercado através de workshops colaborativos, definindo visão do produto, *personas*[2] e jornadas de usuário. Arquiteturalmente, a aplicação web adotou uma abordagem híbrida com *Clean Architecture* no back-end e *Feature-Based Domain Architecture* no front-end, garantindo desacoplamento, manutenibilidade e evolução independente das camadas. Tecnologias como BullMQ com Redis foram empregadas para operações assíncronas, enquanto React e Node.js formaram a base do desenvolvimento. Os resultados demonstraram a implementação bem-sucedida das funcionalidades essenciais, incluindo autenticação segura, dashboard analítico com métricas em tempo real, gestão completa do ciclo de vida dos ativos, vinculação de assinaturas, geração de QR Codes para identificação física e módulos administrativos para usuários, centros de custo e perfis de acesso. A conclusão evidencia que o produto atingiu a maioria dos objetivos propostos, eliminando planilhas manuais, centralizando dados confiáveis e oferecendo transparência no gerenciamento patrimonial, com exceção do cálculo de depreciação, remanejado para versões futuras. O GuardItens comprova ser uma solução robusta e acessível que equilibra funcionalidades essenciais com usabilidade intuitiva, preenchendo uma lacuna significativa no mercado de gestão de TI para PMEs.

Palavras-chaves: Gestão de ativos de TI; *ITAM*; Tecnologia da Informação; *Lean Inception*; Arquitetura de Software.

[1] *Information Technology Asset Management* - Gerenciamento de Ativos de Tecnologia da Informação
[2] *Personas* - Perfis de usuários representativos

Lista de ilustrações

Figura 1 – Tamanho do mercado de gerenciamento de ativos de TI (2024-2029). . 12
Figura 2 – Ciclo de vida dos ativos de TI. 19
Figura 3 – Principais economias geradas por programas *SAM* (2024). 21
Figura 4 – Crescimento de mercado de *ITAM* por região (2023). 24
Figura 5 – Agenda de um *Lean Inception*. 29
Figura 6 – Visão do produto 30
Figura 7 – Quadro É/Não É - Faz/Não Faz. 31
Figura 8 – Objetivos do Produto. 32
Figura 9 – Persona 1. 35
Figura 10 – User Story 1. 35
Figura 11 – Persona 2. 37
Figura 12 – User Story 2. 37
Figura 13 – Persona 3. 38
Figura 14 – User Story 3. 39
Figura 15 – Brainstorm de Funcionalidades 40
Figura 16 – Brainstorm de Funcionalidades - Escopo Final 41
Figura 17 – Revisão das funcionalidades. 42
Figura 18 – Revisão das Funcionalidades - Escopo Final 43
Figura 19 – Sequenciador. Ondas 1, 2 e 3. 44
Figura 20 – Sequenciador. Ondas 4, 5 e 6. 45
Figura 21 – Sequenciador - Ondas 1, 2 e 3 (Versão reestruturada) 46
Figura 22 – Sequenciador - Ondas 4, 5 e 6 (Versão reestruturada) 46
Figura 23 – Canva MVP. 47
Figura 24 – Canvas MVP - Versão Final 49
Figura 25 – Framework Scrum. 50
Figura 26 – Método Kanban. 53
Figura 27 – Estrutura de épicos, histórias e iniciativas. 55
Figura 28 – Matriz de Roadmap. 62
Figura 29 – Quadro Kanban para gestão do fluxo de trabalho. 62
Figura 30 – Logotipo do GuardItens. 63
Figura 31 – Protótipo da tela de login. 64
Figura 32 – Protótipo da tela de *dashboard*. 65
Figura 33 – Protótipo da tela de equipamentos. 66
Figura 34 – Protótipo da tela de cadastro de equipamentos. 66
Figura 35 – Protótipo da tela de cadastro de equipamentos parte 2. 67
Figura 36 – Protótipo da tela de cadastro de equipamentos parte 3. 67

Figura 37 – Protótipo da tela de assinaturas com ativos desvinculados. 68
Figura 38 – Protótipo da tela de cadastro de vinculação parte 1. 69
Figura 39 – Protótipo da tela de cadastro de vinculação parte 2. 69
Figura 40 – Protótipo da tela de assinaturas com ativos vinculados. 70
Figura 41 – Protótipo da tela de histórico do ativo. 71
Figura 42 – Arquitetura de software. 83

Lista de abreviaturas e siglas

CARG	Taxa de Crescimento Anual Composta
CMDB	*Configuration Management Database*[3]
CRUD	*Create Read Update Delete*[4],
MVP	*Minimum Viable Product*[5]
PME	Pequenas e Médias Empresas
ISTM	*IT Service Management*[6]
ITAM	*Information Technology Asset Management*[7]
ITIL	*Information Technology Infrastructure Library*[8]
SAM	*Software Asset Management*[9]
TI	Tecnologia da Informação

[3] *Configuration Management Database* - Configuração de gerenciamento de Banco de Dados
[4] *Create Read Update Delete* - Criar Ler Atualizar Deletar
[5] *Minimum Viable Product* - Mínimo Produto Viável
[6] *IT Service Management* - Gerenciamento de Serviços de TI
[7] *Information Technology Asset Management* - Gerenciamento de Ativos de Tecnologia da Informação
[8] *Information Technology Infrastructure Library* - Biblioteca de Infraestrutura de Tecnologia da Informação
[9] *Software Asset Management* - Gerenciamento de Ativos de software

Sumário

1 INTRODUÇÃO . . . 11
1.1 Contexto . . . 11
1.2 Justificativa . . . 14
1.3 Objetivos . . . 15
1.3.1 Objetivo geral . . . 15
1.3.2 Objetivo específicos . . . 15
1.4 Estado da Realização dos Objetivos . . . 16
1.4.1 Objetivo 1: Elicitação de Requisitos . . . 16
1.4.2 Objetivo 2: Desenvolver um Protótipo Funcional . . . 16
1.4.3 Objetivo 3: Definir a Arquitetura do Sistema . . . 16
1.4.4 Objetivo 4: Procedimentos de Teste . . . 17

2 GESTÃO DE ATIVOS DE TI - *ITAM* . . . 18
2.1 Ativos de TI . . . 18
2.2 Ciclo de vida dos ativos de TI . . . 19
2.3 Vantagens de *ITAM* . . . 20
2.4 Implementação da gestão de ativos de TI . . . 22
2.5 Mercado de *ITAM* . . . 23

3 DEFINIÇÃO DO CLIENTE . . . 27
3.0.1 Descrição do cliente envolvido . . . 27

4 METODOLOGIAS . . . 28
4.1 Requisitos do Produto - *Lean Inception* . . . 28
4.2 Fases da *Lean Inception* Aplicadas ao Projeto . . . 29
4.2.1 1ª Etapa: Definição da Visão . . . 29
4.2.1.1 Visão do Produto . . . 29
4.2.1.2 Quadro É/Não É - Faz/Não Faz . . . 30
4.2.1.3 Objetivos do Produto . . . 31
4.2.2 2ª Etapa: Personas e Jornadas de Usuário . . . 33
4.2.2.1 Personas . . . 33
4.2.2.2 Jornada do Usuário . . . 33
4.2.2.2.1 Persona e jornada de usuário 1 . . . 33
4.2.2.2.2 Persona e jornada de usuário 2 . . . 35
4.2.2.2.3 Persona e jornada de usuário 3 . . . 37
4.2.3 3ª etapa: Brainstorm de Funcionalidades e Revisão das Funcionalidades . . . 39

4.2.3.1 Brainstorm de Funcionalidades Iniciais . . . 39
4.2.3.2 Revisão e Evolução Definitiva . . . 40
4.2.3.3 Revisão das Funcionalidades . . . 41
4.2.3.4 Revisão Final das Funcionalidades . . . 42
4.2.4 4ª etapa: Sequenciador e Canva MVP . . . 43
4.2.4.1 Sequenciador . . . 43
4.2.4.1.1 Sequenciador - Readequação das Ondas de Desenvolvimento . . . 45
4.2.4.2 Canva MVP . . . 46
4.2.4.2.1 Canvas MVP - Evolução e Definição Final . . . 47
4.3 Desenvolvimento do produto . . . 49
4.3.1 Scrum . . . 49
4.4 Kanban: Princípios, Funcionamento e Integração com Scrum . . . 52
4.5 Iniciativas, Épicos e Histórias de Usuário . . . 54
4.5.1 Histórias de usuário . . . 55
4.5.1.1 Trabalhando com Histórias de Usuário . . . 56
4.5.1.2 Boas Práticas para a Elaboração de Histórias de Usuário . . . 56

5 PROPOSTA DE SOLUÇÃO . . . 58
5.1 Backlog . . . 58
5.1.1 Épicos . . . 58
5.1.2 Histórias de Usuário . . . 59
5.2 Roadmap . . . 61
5.2.1 Elaboração do Roadmap . . . 61
5.3 Protótipos . . . 62
5.3.1 Diretrizes de design e identidade visual . . . 63
5.3.2 *Login* e tela inicial . . . 64
5.3.3 Gerenciamento de equipamentos . . . 65
5.3.4 Gestão de assinaturas . . . 68
5.3.4.1 Equipamentos desvinculados . . . 68
5.3.4.2 Equipamentos vinculados . . . 70
5.3.5 Histórico e status . . . 70

6 TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO . . . 72
6.1 Ferramentas de Apoio ao Desenvolvimento . . . 72
6.1.1 Github . . . 72
6.1.2 Miro . . . 72
6.1.3 Figma . . . 73
6.2 Tecnologias de Desenvolvimento . . . 73
6.2.1 Front-end: React . . . 73
6.2.1.1 Tecnologias Complementares para o Ecossistema React . . . 74

6.2.2 Back-end: Node.js . . . 74
6.2.2.1 Tecnologias Complementares para o Ecossistema Node.js . . . 75
6.2.3 Linguagem de Programação: TypeScript . . . 76
6.2.4 Validação de Dados com Zod . . . 76
6.2.5 Testes de Software com Vitest . . . 77
6.2.6 Containerização com Docker e Banco de Dados PostgreSQL . . . 77

7 ARQUITETURA DE SOFTWARE . . . 79
7.1 Arquitetura do Back-end: *Clean Architecture* . . . 79
7.1.1 Escolhas de Implementação para Tarefas Assíncronas . . . 80
7.2 Arquitetura do Front-end: *Feature-Based Domain Architecture* . . . 80
7.3 Comunicação e Integração . . . 81
7.4 Vantagens da Abordagem Proposta . . . 81
7.5 Diagrama de arquitetura . . . 82

8 CONTAINERIZAÇÃO E ORQUESTRAÇÃO DE SERVIÇOS . . . 84
8.1 Arquitetura de Containerização do *Backend* . . . 84
8.1.1 Docker Compose do *Backend* . . . 84
8.1.2 Configuração dos Serviços de *Backend* . . . 86
8.2 Arquitetura de Containerização do *Frontend* . . . 87
8.2.1 Docker Compose do *Frontend* . . . 87
8.2.2 Dockerfile do *Frontend* . . . 87
8.2.3 Configuração do Serviço de *Frontend* . . . 88
8.3 Vantagens da Abordagem Adotada . . . 89
8.4 Considerações sobre *GitHub Actions* . . . 89

9 DEMONSTRAÇÃO DAS FUNCIONALIDADES IMPLEMENTADAS 90
9.1 Sistema de Autenticação e Acesso . . . 90
9.1.1 Tela de Login . . . 90
9.1.1.1 Tela de Login - Campos Inválidos . . . 90
9.1.2 Fluxo de Primeiro Acesso . . . 91
9.1.3 Tela de Redefinição de Senha Obrigatória . . . 91
9.1.4 Troca de Senha . . . 91
9.1.5 Perfil do Usuário e Gestão de Permissões . . . 91
9.1.6 Segurança e Auditoria . . . 91
9.2 Dashboard e Visão Geral . . . 92
9.2.1 Dashboard Principal . . . 92
9.3 Gestão de Ativos de TI . . . 92
9.3.1 Cadastro de Ativos . . . 92
9.3.2 Histórico de Vinculações e Rastreamento . . . 93

9.3.3 Identificação por QR Code 93
9.3.4 Consulta e Pesquisa de Ativos 93
9.4 Gestão de Assinaturas de Equipamentos 93
9.4.1 Vinculação de Assinaturas a Ativos 93
9.5 Módulo de Adiministração 94
9.5.1 CRUD de Usuários 94
9.5.2 CRUD de Centros de Custo 94
9.5.3 CRUD de Papéis de Acesso 94
9.6 Considerações Finais da Implementação 95

10 CONCLUSÃO 96

REFERÊNCIAS 98

1 Introdução

1.1 Contexto

Nos últimos anos, o mercado tornou-se cada vez mais dinâmico e competitivo, exigindo que as empresas implementem estratégias eficazes para reduzir custos e garantir sua competitividade. Diante desse cenário, a Tecnologia da Informação (TI) evoluiu de um mero suporte operacional para um recurso estratégico essencial, cuja administração influencia diretamente o desempenho e os resultados organizacionais.

Como destaca Fagundes (2006), empresas de médio e grande porte hoje dependem de um parque de equipamentos e sistemas integrados em um ambiente gerenciável. Essa demanda surgiu como resposta à crescente pressão competitiva, que forçou organizações a repensarem seus investimentos em TI, buscando não apenas eficiência operacional, mas também redução de custos e melhor aproveitamento de recursos.

Ortolani (1995) reforça essa perspectiva ao abordar o valor da informação sob a ótica do custo de oportunidade: "quanto custa não tê-la". Essa visão mercadológica trata a informação e, por extensão, os ativos de TI, como um recurso mensurável, com impacto financeiro direto. Afinal, em um cenário onde a concorrência exige agilidade e inteligência de negócios, a falta de controle sobre hardwares, softwares e dados pode significar perda de competitividade e oportunidades.

Diante disso, o Gerenciamento de Ativos de TI surge como uma disciplina essencial, definido por Natal (2010) como o conjunto de "sistemas, processos e controles para medir e gerenciar os ativos de TI e seu ciclo de vida em uma organização". Sua função é equilibrar as demandas financeiras (custos, compliance e ROI) com as necessidades operacionais (inovação, desempenho e continuidade), garantindo que a TI seja não apenas um gasto, mas um diferencial estratégico.

O *ITAM* tornou-se uma peça fundamental no mercado e denotou um crescimento constante nos últimos anos. Segundo a Mordor Intelligence (2023), o mercado global de gestão de ativos de TI está avaliado em US$ 1,97 bilhões em 2024 e deve atingir US$ 2,68 bilhões até 2029, com uma taxa de crescimento anual (CAGR) de 6,32%, como demonstrado pela Figura 1.

Este crescimento é impulsionado pela digitalização de setores estratégicos (como saúde, finanças e manufatura), pela adoção de *cloud computing* que facilita a implantação de sistemas *ITAM*, pelas pressões regulatórias que exigem maior controle sobre licenças de software e *compliance*, e pela busca por eficiência no período pós-pandemia, já que a COVID-19 acelerou a necessidade de otimização de custos em TI.

Figura 1 – Tamanho do mercado de gerenciamento de ativos de TI (2024-2029).



Fonte: Mordor Intelligence (2023).

No entanto, apesar do mercado em expansão, a adoção de *ITAM* ainda é desigual. Estudos do Gartner Peer Insights (2023) revelam que empresas podem reduzir seus custos de TI em cerca de 20% através de um gerenciamento eficaz de seus ativos. Além disso, organizações que gerenciam sistematicamente o ciclo de vida dos ativos de TI alcançam reduções de 30% nos custos por ativo no primeiro ano, e de 5 a 10% nos anos subsequentes. Contudo, apesar desses benefícios evidentes, apenas 25% das empresas globais possuem ferramentas específicas para gestão de ativos de TI instaladas.

Estes dados evidenciam uma realidade preocupante: enquanto a gestão de ativos de

TI comprovadamente gera eficiência e redução de custos, a maioria das organizações ainda negligencia essa prática estratégica. Como alerta Natal (2010), essa lacuna representa não apenas um desperdício de recursos, mas também uma vulnerabilidade competitiva em um mercado cada vez mais dependente da otimização tecnológica.

A implementação de um sistema robusto de gestão de ativos de TI permite às organizações não apenas cortar custos, mas também aumentar a visibilidade sobre seu patrimônio tecnológico, melhorar a tomada de decisões e garantir conformidade com regulamentações - fatores críticos para a sustentabilidade dos negócios na era digital.

Apesar dos comprovados benefícios, a implementação efetiva da Gestão de Ativos de TI (*ITAM*) ainda enfrenta significativas barreiras organizacionais. Um estudo realizado pela Deloitte (2021) com as maiores corporações da indústria norte-americana revelou um cenário preocupante no qual 22% das empresas pesquisadas encontram-se em estágio inicial ou completamente ausentes de qualquer iniciativa estruturada de *ITAM*. Entretanto 82% dos respondentes demonstraram intenção de avançar significativamente nesta área nos próximos anos.

Este paradoxo entre a baixa adoção atual e o crescente interesse evidencia uma transição crítica no mercado. Como observa Natal (2010), a gestão de ativos de TI frequentemente é relegada a segundo plano frente a demandas operacionais imediatas, criando uma lacuna entre o potencial estratégico e a prática organizacional.

Os dados da Deloitte (2021) corroboram com as descobertas anteriores do Gartner Group, sugerindo que existe uma clara correlação entre maturidade em *ITAM* e vantagem competitiva e que o mercado está em um ponto de inflexão, com crescente reconhecimento do valor estratégico da gestão de ativos

Esta conjuntura apresenta uma janela de oportunidade única para organizações que desejam se antecipar às tendências do mercado. Como destacado por Ortolani (1995), o custo da inação - especialmente em ambientes competitivos - frequentemente supera em muito os investimentos necessários para implementação de soluções robustas de gestão de ativos.

Diante desse cenário, surge um desafio crucial: como democratizar o acesso a ferramentas eficientes de *ITAM* em um mercado onde 22% das empresas ainda não possuem nenhuma implementação? A resposta pode estar no desenvolvimento de uma plataforma que combine **acessibilidade**, com custos adaptáveis à realidade de pequenas e médias empresas; **essencialidade**, através do foco nas funcionalidades críticas do ciclo de vida de ativos; **intuitividade**, por meio de uma interface simplificada para gestores com limitações técnicas; e **competitividade**, pela incorporação das melhores práticas de mercado.

Esta proposta se alinha diretamente com as lacunas identificadas por Natal (2010) e Fagundes (2006), onde a ausência de soluções adaptáveis cria um paradoxo: empresas

reconhecem a importância do *ITAM* (82% pretendem avançar), mas esbarram em barreiras de custo e complexidade. Ao desenvolver uma plataforma que traduz os benefícios comprovados pela pesquisa do Gartner Group em uma solução tangível e economicamente viável, este trabalho busca transformar o custo da inação, destacado por (ORTOLANI, 1995), em oportunidades concretas de gestão estratégica.

1.2 Justificativa

A realização deste trabalho se justifica pela lacuna crítica entre o potencial estratégico do Gerenciamento de Ativos de TI (*ITAM*) e sua adoção desigual no mercado, conforme evidenciado pelos dados do Gartner Group e da Deloitte (2021). Apesar dos benefícios comprovados, como a redução de custos em até 30% por ativo e maior conformidade regulatória, 25% das empresas globais ainda carecem de ferramentas específicas para *ITAM*, e 22% sequer iniciam práticas estruturadas nessa área (DELOITTE, 2021; NATAL, 2010). Essa contradição sinaliza um problema urgente: a ausência de soluções acessíveis e adaptáveis, especialmente para pequenas e médias empresas, que representam a maioria do tecido empresarial global. Do ponto de vista organizacional, a justificativa se ancora em dois pilares:

a) **eficiência econômica**: como destacado por Ortolani (1995), o "custo de não ter informação"supera os investimentos em gestão de ativos. Empresas que negligenciam o ITAM enfrentam desperdícios médios de 20% em custos de TI (Gartner Peer Insights, 2023), além de riscos de compliance e perda de competitividade;

b) **democratização do *ITAM***: o crescimento do mercado (6,32% ao ano até 2029, segundo Mordor Intelligence (2023)) é impulsionado por grandes corporações, deixando PMEs em desvantagem. Este trabalho propõe uma plataforma com custos adaptáveis e funcionalidades essenciais, preenchendo uma lacuna identificada por Fagundes (2006) e Natal (2010).

Para a academia, a relevância reside na formalização de processos de *ITAM*, ainda pouco explorados na literatura comparados a outras áreas de TI. Como observa Natal (2010), há uma carência de modelos reconhecidos e acessíveis, o que limita a visibilidade do tema. Este trabalho contribui ao sistematizar práticas existentes, integrando frameworks como o *ITIL*[1] às demandas reais de PMEs, ao oferecer uma base empírica que pode servir de referência para estudos sobre maturidade em *ITAM*, e ao promover uma abordagem interdisciplinar que conecta gestão tecnológica, economia (com foco no custo

[1] *Information Technology Infrastructure Library* - Biblioteca de Infraestrutura de Tecnologia da Informação

de oportunidade) e estratégia organizacional, em consonância com a visão mercadológica proposta por Ortolani (1995).

1.3 Objetivos

Esta seção define os objetivos desta pesquisa, divididos em objetivo geral e objetivos específicos. O objetivo geral apresenta uma visão ampla do propósito do projeto, enquanto os objetivos específicos delineiam as etapas necessárias para sua implementação.

Dessa forma, este capítulo estabelece o direcionamento a ser estabelecido ao longo do trabalho, definindo de forma clara os resultados pretendidos e o caminho para alcançá-los.

Neste TCC 1, serão abordadas as bases teóricas e práticas relacionadas à gestão de ativos de TI (*ITAM*), incluindo a definição do escopo da área de estudo, mapeamento de vantagens, soluções disponíveis no mercado e boas práticas de implementação. Além de ser realizado a implementação do Lean Inception para definição estrutural do *MVP*.

Já no TCC 2, o foco será na evolução da solução proposta, com o seu desenvolvimento e validação, tendo como base nas diretrizes estabelecidas nesta primeira fase para a construção de um produto de *ITAM* satisfatório tanto para o cliente como o mercado.

1.3.1 Objetivo geral

O objetivo geral deste projeto é desenvolver um sistema de gestão de ativos de TI que permita monitorar, rastrear e otimizar o uso de recursos tecnológicos dentro das organizações. O sistema deve possibilitar um controle eficiente do ciclo de vida dos ativos, incluindo aquisição, utilização, manutenção e descarte.

1.3.2 Objetivo específicos

Com o objetivo central já delineado, é essencial determinar os objetivos específicos que irão guiar o desenvolvimento do aplicativo. Esses objetivos buscam definir as etapas para que o objetivo geral seja alcançado, proporcionando uma experiência eficiente e agradável. Abaixo, são descritos os objetivos específicos definidos para o aplicativo:

1. **Elicitação de requisitos**: Levantar os requisitos do sistema de gestão de ativos de TI em conjunto com o cliente envolvido, buscando considerar necessidades empresariais e boas práticas de governança.
2. **Desenvolver um protótipo funcional**: Criar um software que permita o registro, monitoramento e gestão de ativos de TI, incluindo funcionalidades de rastreamento, controle e analise de dados.

3. **Definir a arquitetura do sistema**: Estruturar a arquitetura do software, descrevendo padrões e abordagens adotadas para garantir eficiência e escalabilidade.

4. **Procedimentos de teste**: Aplicar testes unitários e de integração para garantir a qualidade e confiabilidade do sistema, além de garantir a conformidade com os requisitos definidos.

1.4 Estado da Realização dos Objetivos

Esta seção apresenta o acompanhamento da execução de cada objetivo específico ao longo do desenvolvimento do projeto, detalhando o estado de conclusão e eventuais ajustes realizados.

1.4.1 Objetivo 1: Elicitação de Requisitos

Estado: Concluído integralmente. **Realização:** A elicitação de requisitos foi realizada através do workshop de *Lean Inception*, que envolveu sessões colaborativas com os stakeholders. Foram definidas as personas principais (gerentes de TI, analistas de suporte, auditores), mapeadas as jornadas do usuário e priorizadas as funcionalidades para o MVP. O processo resultou na especificação de requisitos que inicialmente foram ajustados de acordo com o desenvolvimento do projeto.

1.4.2 Objetivo 2: Desenvolver um Protótipo Funcional

Estado: Concluído. **Realização:** O sistema GuardItens foi desenvolvido e implementado com a maioria das funcionalidades essenciais operacionais. A aplicação inclui: sistema de autenticação seguro, dashboard com métricas em tempo real, gestão do ciclo de vida dos ativos, cadastro e consulta de equipamentos, vinculação de assinaturas, geração de QR Codes para identificação física, e módulos administrativos para usuários, centros de custo e perfis de acesso. No entanto, o produto final sofreu adaptações em relação ao protótipo inicial, com ajustes no escopo funcional realizados durante o desenvolvimento para adequação a restrições técnicas e de prazo. A funcionalidade de cálculo automatizado de depreciação, originalmente prevista, foi remanejada para versões futuras devido à sua complexidade contábil específica, assim como outras funcionalidades especificadas posteriormente.

1.4.3 Objetivo 3: Definir a Arquitetura do Sistema

Estado: Concluído e validado. **Realização:** A arquitetura do sistema foi definida seguindo uma abordagem híbrida, com *Clean Architecture* no back-end e *Feature-Based*

Domain Architecture no front-end. Esta escolha arquitetural demonstrou-se eficaz durante o desenvolvimento, proporcionando desacoplamento, manutenibilidade e evolução independente das camadas. Foram especificados os padrões tecnológicos incluindo BullMQ com Redis para operações assíncronas, React com TypeScript no front-end e Node.js no back-end.

1.4.4 Objetivo 4: Procedimentos de Teste

Estado: Não realizado - trabalho futuro. **Realização:** A implementação formal de procedimentos de teste não foi executada devido a restrições de escopo e prazo. Durante o desenvolvimento, foram realizados testes manuais exploratórios para validação das funcionalidades principais e verificação da integração entre os módulos. A ausência de uma suíte automatizada de testes unitários e de integração foi identificada como uma lacuna significativa, sendo priorizada como melhoria crítica para versões futuras do sistema, onde se planeja implementar testes automatizados para garantir maior confiabilidade e facilitar a manutenção evolutiva.

2 Gestão de Ativos de TI - *ITAM*

Este capítulo explora os fundamentos da gestão de ativos de TI, divididos em quatro eixos principais: (1) a definição e classificação dos ativos de TI, (2) as etapas do seu ciclo de vida desde a aquisição até a descontinuação, (3) os prós e contras da gestão sistemática desses ativos, e (4) um panorama do mercado de soluções de *ITAM*.

A análise destes tópicos visa demonstrar como uma gestão eficiente de ativos pode otimizar custos e operações, ao mesmo tempo que revela as lacunas existentes entre as necessidades das organizações e as soluções disponíveis no mercado, problema que motiva o desenvolvimento desta proposta.

2.1 Ativos de TI

Os ativos de TI são definidos como “itens ou entidades que possuem valor real ou potencial para uma organização” (International Organization for Standardization, 2014), que têm como objetivo armazenar e distribuir informações que podem contribuir para a entrega de um produto ou serviço de TI. Conforme Bonham (2005), esses ativos se caracterizam por:

1. **Hardware**: inclui todas as plataformas nas quais os ativos de software podem ser executados, como notebooks, desktops, tablets, smartphones e servidores físicos ou virtuais (ABES, 2020).
2. **Software**: abrange os programas autorizados para instalação e execução do negócio, organizados por plataforma (Windows, UNIX, MacOS etc.) (ABES, 2020).
3. **Contratos**: SLAs (Service Level Agreements), contratos de manutenção e acordos de suporte técnico.
4. **Licenças**: direitos de uso com modelos variados, como perpétuos, por assinatura ou baseados em consumo.

Conforme Green e Helstrom (2011), "o valor dos ativos não deve ser medido apenas financeiramente, mas pela sua importância estratégica para a organização". Essa premissa é particularmente relevante para ativos de TI, que possuem ciclo de vida mais curto devido à rápida obsolescência tecnológica (AXELOS, 2019), exigindo processos de refresh planejados e políticas de end-of-life claras.

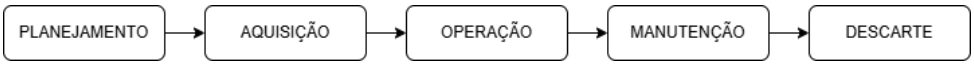
2.2 Ciclo de vida dos ativos de TI

O ciclo de vida dos ativos de TI representa um conceito fundamental para a gestão eficaz dos recursos tecnológicos de uma organização. Conforme destacado pelo Institute of Asset Management (2008), esse ciclo compreende todas as etapas desde a identificação da necessidade do ativo até seu descarte final, quando deixa de ter utilidade para a empresa. Embora existam diferentes modelos para representar esse processo, quatro atividades essenciais devem ser sempre consideradas: aquisição do recurso, sua utilização operacional, manutenção durante o período de vida útil e finalmente a desmobilização ou aposentadoria do ativo.

Todos os ativos de TI, sejam eles *hardware* ou *software*, físicos ou virtuais, passam por esse ciclo de forma repetitiva. Como explica Boerger (2021), esses recursos são inicialmente adquiridos, depois preparados e configurados para uso, mantidos durante sua operação e, ao final de sua vida útil, recuperados para reutilização, reciclados ou descartados adequadamente. Cada uma dessas etapas representa um estado específico na vida do ativo, sendo importante destacar que a metodologia e terminologia empregadas podem e devem ser adaptadas à realidade de cada organização.

Uma visão mais detalhada desse ciclo de vida, pode ser ilustrada por uma abordagem holística em cinco fases interconectadas, como ilustrada na Figura 2. (Atlassian, 2023a)

Figura 2 – Ciclo de vida dos ativos de TI.



Fonte: Adaptado de Atlassian (2023a)

A fase de planejamento, por exemplo, vai além da simples identificação de necessidades - quando uma empresa decide implementar inteligência artificial em seu processo de vendas, é crucial avaliar não apenas os requisitos técnicos do software, mas também sua capacidade de integração com os sistemas existentes, o retorno sobre o investimento esperado e os possíveis riscos de adoção.

Na sequência, a fase de aquisição envolve processos complexos que vão desde a seleção cuidadosa de fornecedores até a negociação de contratos que contemplem futuras atualizações tecnológicas. Um caso prático pode ser observado na aquisição de laptops corporativos, onde especificações técnicas, prazos de garantia e cláusulas de suporte devem ser minuciosamente analisados para evitar custos adicionais não previstos.

Durante a operação, que corresponde ao período de maior geração de valor, destaca-se a importância de treinamentos periódicos para os usuários e monitoramento constante

de desempenho. No exemplo dos laptops corporativos, isso se traduz em programas de capacitação para a equipe e acompanhamento sistemático de indicadores de utilização.

Já a manutenção preventiva e corretiva garante que os ativos operem com máxima eficiência ao longo de seu ciclo de vida. Práticas como atualizações regulares de software, substituição preventiva de componentes e documentação de incidentes são essenciais para prolongar a vida útil dos equipamentos e sistemas.

Por fim, a fase de descarte requer cuidados específicos na TI, particularmente no que diz respeito à sanitização de dados sensíveis e ao descarte ambientalmente correto de componentes eletrônicos. Uma gestão adequada nesta etapa final pode gerar benefícios como a recuperação de valor residual através da revenda de equipamentos ou a mitigação de riscos legais associados ao descarte inadequado.

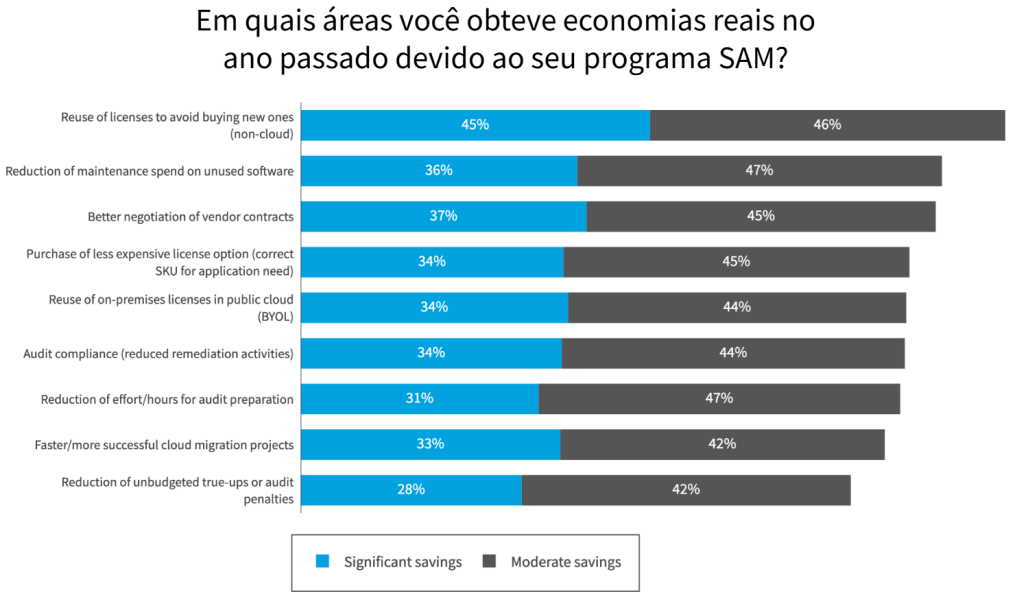
Essa abordagem integrada do ciclo de vida, quando aplicada a projetos reais como a implantação de uma frota de laptops corporativos, demonstra como as fases se inter-relacionam. Desde o planejamento inicial que define as especificações técnicas adequadas às necessidades dos usuários de vendas, passando pelo treinamento contínuo, até os protocolos de descarte seguro que protegem os dados da empresa. Cada etapa contribui para maximizar o valor gerado pelos ativos enquanto minimizam custos e riscos associados. (Atlassian, 2023a)

Segundo Boerger (2021), os ativos de tecnologia demandam uma descrição mais detalhada de seu ciclo de vida quando comparados a outros tipos de ativos corporativos. Essa particularidade justifica a necessidade de etapas específicas para esses recursos. O autor ressalta ainda que as organizações têm flexibilidade para combinar etapas, ajustar a extensão do ciclo ou mesmo adaptar a nomenclatura dos processos envolvidos, desde que os princípios fundamentais de gestão ao longo do ciclo de vida sejam preservados.

2.3 Vantagens de *ITAM*

A implementação sistemática da gestão de ativos de TI oferece benefícios tangíveis para organizações de todos os portes. Conforme demonstrado por Natal (2010), empresas que adotam práticas estruturadas de *ITIL* alcançam reduções médias de 20% nos custos de TI, com economias específicas de 30% por ativo no primeiro ano e 5-10% nos anos subsequentes. Esses resultados alinham-se com dados recentes da Flexera (2024) (Figura 3), cujo relatório com 503 organizações globais (91% acima de 2.000 funcionários) revelou que:

Figura 3 – Principais economias geradas por programas *SAM* (2024).



Fonte: Flexera (2024)

A gestão diligente proporciona visibilidade integral sobre status, uso e desempenho dos ativos, conforme destacado pela Atlassian (2023a). Entre os principais benefícios comprovados estão:

1. **Melhoria de desempenho**: análises contínuas de dados operacionais permitem otimizações que reduzem em 12% o desperdício em software de *data center* Flexera (2024), além de diminuir tempo de inatividade e frequência de reparos.

2. **Extensão da vida útil**: o monitoramento proativo durante todo o ciclo de vida, desde a aquisição de ativos de qualidade até manutenções preventivas, pode ampliar sua utilização efetiva em até 40% (NATAL, 2010).

3. **Conformidade e mitigação de riscos**: protocolos estruturados de *ITAM* reduzem penalidades por auditorias (28%), esforço na preparação para compliance (31%) e riscos operacionais por meio do mapeamento prévio de vulnerabilidades (Atlassian, 2023a).

4. **Sinergia organizacional**: conforme o relatório da Flexera (2024), equipes de *SAM* com interação frequente apresentam 65% de alinhamento com FinOps e gestão de fornecedores, e 63% de integração com arquitetura corporativa e segurança.

Estes resultados corroboram a afirmação de Natal (2010) sobre o caráter estratégico do *ITAM*: embora apenas 25% das empresas globais adotem ferramentas especiali-

zadas, aquelas que o fazem transformam seu parque tecnológico de centro de custo em vantagem competitiva.

2.4 Implementação da gestão de ativos de TI

A implementação da gestão de ativos de TI (*ITAM*) é um processo estruturado que exige planejamento, políticas claras, adoção de ferramentas adequadas e engajamento institucional. O objetivo central é garantir o controle, a conformidade e o uso eficiente dos recursos tecnológicos ao longo de todo o seu ciclo de vida.

Segundo a ABES (2020), o processo de *ITAM* deve ser construído em seis etapas fundamentais:

1. **Planejamento e Diagnóstico:** O primeiro passo é realizar um diagnóstico do ambiente atual. Isso inclui a identificação dos ativos existentes, suas condições, a forma como são gerenciados, e os principais riscos envolvidos. O manual da ABES (2022) destaca que muitas organizações operam com baixa maturidade em *ITAM*, utilizando planilhas manuais e controles descentralizados. Essa etapa visa mapear o estado atual da organização e estabelecer metas realistas de evolução.

2. **Definição de Políticas e Procedimentos:** A governança em *ITAM* requer a criação de políticas formais para aquisição, uso, descarte e substituição de ativos. É fundamental definir papéis e responsabilidades: quem aprova compras, quem realiza inventários, quem audita licenças, entre outros. O manual recomenda que essas políticas estejam alinhadas com normas como ISO/IEC 19770 e com legislações nacionais, como a Lei nº 13.709 (2018), especialmente no que tange ao tratamento de dados contidos em ativos descartados e evitando o acarretamento de multas, sanções legais e riscos reputacionais com o descumprimento dessas diretrizes.

3. **Inventário e Classificação de Ativos:** A etapa de inventário envolve identificar todos os ativos de hardware e software em uso, classificando-os por tipo, localização, usuário responsável e ciclo de vida. Soluções de *ITAM* modernas permitem realizar esse processo de forma automatizada, integrando com sistemas de ITSM (IT Service Management). O inventário contínuo permite não apenas controle, mas também a visibilidade necessária para auditorias, conformidade com contratos de licença e prevenção de uso indevido.

4. **Adoção de Ferramentas de *ITAM*:** A escolha da ferramenta deve levar em consideração o porte da empresa, a complexidade da infraestrutura e o grau de automação necessário. O manual recomenda soluções que ofereçam controle de licenças, auditoria, inventário automatizado, gerenciamento de ciclo de vida dos ativos

e integração com ERP ou sistemas de chamados. Ferramentas como ServiceNow, Lansweeper, ManageEngine e Ivanti são citadas em outros relatórios (como o da Gartner) como líderes nesse segmento.

5. **Treinamento e Conscientização:** A implementação de *ITAM* só terá sucesso se houver envolvimento e capacitação dos colaboradores. A ABES reforça que a conscientização sobre a importância do controle de ativos deve abranger todas as áreas da organização e não apenas a TI. Isso inclui treinamentos periódicos, comunicação interna clara e integração com políticas de segurança da informação.

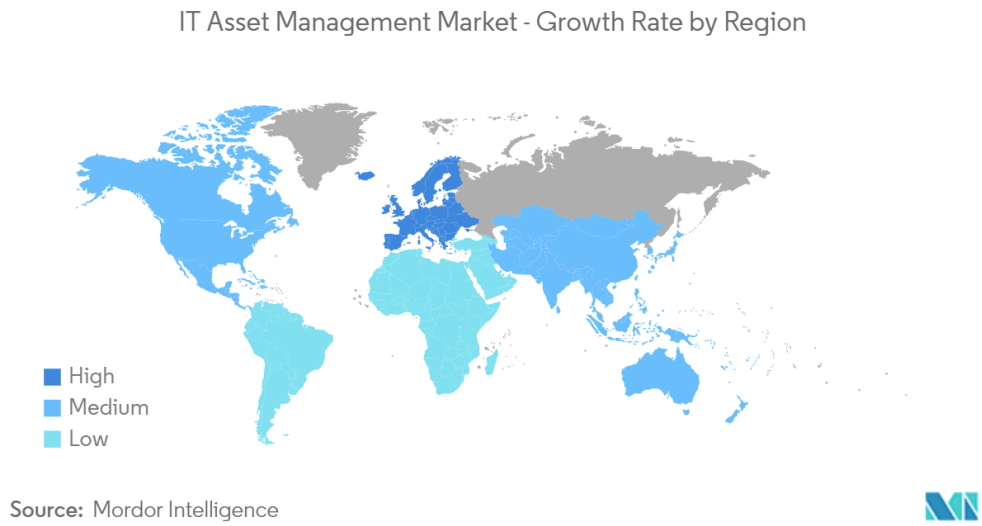
6. **Monitoramento Contínuo e Melhoria:** Por fim, a gestão de ativos não termina com a implementação inicial. O manual recomenda a adoção de indicadores de desempenho (KPIs), revisões periódicas de conformidade e ciclos de melhoria contínua. Um programa de *ITAM* bem estruturado contribui diretamente para a redução de custos (até 30%), mitigação de riscos jurídicos, aumento da produtividade da equipe de TI e maior alinhamento com objetivos estratégicos da organização.

Dessa forma, é essencial compreender que a gestão de ativos de TI não é uma tarefa exclusivamente técnica. Trata-se de uma prática multidisciplinar que exige o alinhamento entre tecnologia, governança e estratégia. Quando bem implementado, o *ITAM* oferece ganhos expressivos em conformidade legal, redução de custos, previsibilidade orçamentária e segurança da informação.

2.5 Mercado de *ITAM*

O mercado de IT Asset Management tem demonstrado um crescimento relevante, impulsionado pelas crescentes demandas por governança tecnológica, conformidade regulatória e controle de custos operacionais. Segundo a consultoria Mordor Intelligence (2023), o mercado global de *ITAM* está projetado para expandir de US$ 1,97 bilhão em 2024 para US$ 2,68 bilhões até 2029, com uma taxa composta de crescimento anual (CAGR) de 6,32% durante esse período, impactando diversas regiões nesse mercado (Figura 4). Este crescimento reflete a crescente valorização do controle dos ativos tecnológicos como uma estratégia essencial para a sustentabilidade dos negócios em ambientes altamente digitalizados.

Figura 4 – Crescimento de mercado de *ITAM* por região (2023).



Fonte: Mordor Intelligence (2023).

Paradoxalmente, apesar do avanço do setor, muitos ambientes corporativos ainda não adotaram sistemas formais de *ITAM*. De acordo com a Deloitte (2021), um número considerável de empresas ainda recorre a métodos não automatizados, como planilhas e controles descentralizados, o que evidencia um descompasso entre a evolução do mercado e a maturidade interna de gestão desses ativos. A pesquisa aponta que mais de 30% das empresas entrevistadas não possuem visibilidade clara sobre os ativos de software instalados ou em uso, o que pode gerar não apenas desperdício financeiro com licenças subutilizadas, mas também riscos legais decorrentes de não conformidade com contratos de software.

No cenário brasileiro, a ABES (2022) destaca que o país movimentou mais de R$ 240 bilhões no setor de tecnologia da informação em 2022, consolidando-se como o 10º maior mercado de TI do mundo. Dentro desse contexto, a gestão de ativos surge como uma prática crítica, principalmente considerando o aumento da complexidade dos ambientes híbridos que envolvem a coexistência de estruturas locais (on-premises), nuvem e dispositivos móveis. Segundo o Manual de Boas Práticas de Gestão de Ativos de Software dessa Associação, a implementação de políticas de *ITAM* pode reduzir custos em até 30% ao longo do ciclo de vida dos ativos, além de melhorar os níveis de segurança e governança.

Atualmente, o mercado de *IT Asset Management* é disputado por diversas soluções que atendem desde pequenas empresas até grandes corporações com infraestruturas altamente complexas. Entre os principais concorrentes, destacam-se plataformas como ServiceNow, Ivanti Neurons, Freshservice, ManageEngine AssetExplorer e Lansweeper, cada uma com abordagens distintas para a gestão de ativos.

A ServiceNow é amplamente reconhecida por sua integração nativa com módu-

los de *IT Service Management (ITSM)*, *Configuration Management Database (CMDB)* e automações inteligentes, sendo considerada líder no mercado corporativo por instituições como o Gartner Peer Insights (2023). Sua robustez e escalabilidade a tornam especialmente atrativa para organizações de grande porte, com operações distribuídas e altos requisitos de compliance.

O Ivanti Neurons for ITAM se destaca pelo uso intensivo de automação e inteligência artificial para detectar, rastrear e gerenciar ativos em tempo real, incluindo recursos voltados para segurança e correção automática de vulnerabilidades. Sua proposta de valor está voltada à eficiência operacional e à redução de custos por meio da visibilidade unificada dos ativos de software e hardware.

Por sua vez, o Freshservice, solução da Freshworks, oferece uma abordagem mais simplificada e acessível, voltada para empresas de médio porte ou equipes que buscam uma curva de aprendizado menor. Sua interface intuitiva e integração com ferramentas colaborativas favorecem ambientes mais dinâmicos e menos burocráticos.

O ManageEngine AssetExplorer, parte da suíte da Zoho Corporation, combina funcionalidades de inventário automatizado, detecção de rede, gerenciamento de ciclo de vida e integração com *help desk*. Seu diferencial está no custo-benefício e na facilidade de implantação em infraestruturas de médio porte.

Já o Lansweeper é bastante utilizado por sua capacidade de realizar escaneamentos detalhados de rede sem a necessidade de instalação de agentes, oferecendo inventário abrangente e relatórios personalizáveis. É uma alternativa popular em ambientes on-premises com foco em controle granular dos ativos físicos.

De acordo com o Gartner Peer Insights (2023), soluções como ServiceNow e Ivanti obtiveram os maiores índices de satisfação dos usuários, especialmente nos quesitos escalabilidade, personalização e suporte técnico. O relatório também destaca que empresas que adotam ferramentas consolidadas reportam uma economia média de até 30% em gastos com licenciamento e uma redução de até 25% no tempo dedicado ao inventário. Esses dados reforçam a importância estratégica da escolha adequada do sistema de *ITAM* conforme as necessidades e maturidade tecnológica da organização.

Dessa forma, o mercado de *ITAM* se apresenta não apenas como uma tendência de crescimento, mas como uma resposta necessária à complexidade atual da gestão de tecnologia. Empresas que adotam práticas eficazes de gestão de ativos não apenas otimizam seus recursos, mas também se posicionam de forma estratégica em um ambiente digital cada vez mais regulado, competitivo e dinâmico.

Ademais, o momento atual é particularmente propício para a inovação em gestão de ativos de TI. A crescente digitalização dos processos organizacionais, aliada à pressão por conformidade regulatória, como a exigida pela Lei nº 13.709 (2018), tem intensificado

a demanda por soluções que proporcionem visibilidade e controle sobre os ativos tecnológicos. Segundo relatório International Data Corporation (2023), cerca de 40% das empresas na América Latina estão em fase avançada de modernização de sua infraestrutura, com foco na substituição de sistemas legados por soluções baseadas em nuvem. Essa transformação amplia a complexidade da gestão de ativos e torna o *ITAM* uma peça central para a eficiência e a segurança operacional. Como destaca a ABES (2020), a escolha de ferramentas deve considerar a capacidade de adaptação ao contexto organizacional, priorizando soluções que aliem automação à simplicidade de uso, o que se mostra essencial para organizações com maturidade tecnológica variada.

3 Definição do cliente

A definição clara do cliente é uma etapa fundamental no gerenciamento de projetos, pois estabelece quem é o principal interessado e facilita a comunicação contínua durante o desenvolvimento. Essa prática permite a coleta de feedbacks iterativos, a incorporação de insights valiosos e a adaptação ágil a mudanças, garantindo que o produto final esteja alinhado com as expectativas reais.

Conforme SILVA (2021), o envolvimento ativo do cliente no desenvolvimento de software é estratégico não apenas para validar requisitos, mas também para reduzir retrabalhos e aumentar a satisfação do usuário final. O autor destaca que essa abordagem minimiza riscos de desvios de escopo e assegura que a solução entregue atenda efetivamente às necessidades do mercado.

3.0.1 Descrição do cliente envolvido

O cliente deste projeto é Guilherme Ferreira, empreendedor e Tech Lead com certificações AWS (Solutions Architect, Developer e SysOps Administrator), além de experiência em governança de sistemas em larga escala. Ele é um especialista em arquitetura de cloud (AWS) e governança de sistemas, com expertise em liderança técnica na Amazon e em sua própria empresa , Software Sob Medida, focada na implementação de produtos como a gestão de operações financeiras (sistemas de reembolso) e em entre outros serviços.

Com perfil técnico e estratégico, o cliente busca uma solução de ITAM, que garanta a rastreabilidade de ativos, segurança de dados (ex.: conformidade com LGPD ou GDPR) e análise preditiva para tomada de decisão. Seu objetivo é consolidar o projeto a sua empresa como referência em gestão inteligente de ativos, combinando sua experiência em AWS com frameworks ágeis para entregar produtos escaláveis.

4 Metodologias

4.1 Requisitos do Produto - *Lean Inception*

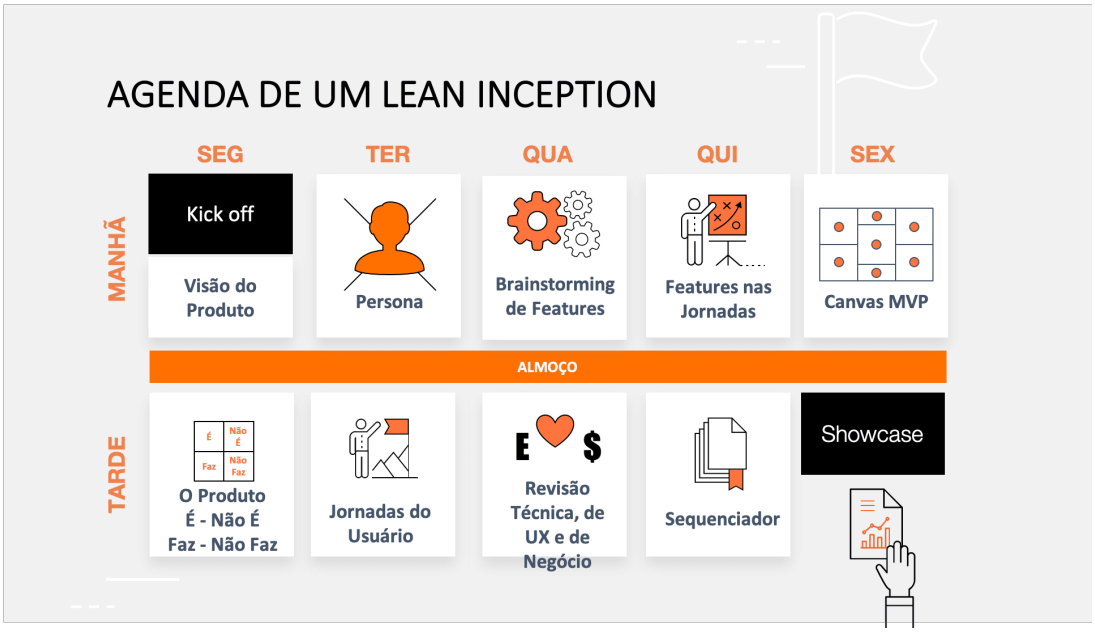
A *Lean Inception*, metodologia cunhada por CAROLI (2024), surge como uma abordagem híbrida entre o *Lean Startup* e o *Design Thinking*, projetada para alinhar equipes multidisciplinares na definição do Produto Mínimo Viável (MVP). Diferente de processos tradicionais de levantamento de requisitos, frequentemente lineares e documentais, esse método propõe um *workshop* colaborativo de curta duração (geralmente uma semana), onde decisões são tomadas de forma visual, iterativa e centrada no usuário. Para o desenvolvimento do *IT Asset Management*, essa dinâmica é particularmente eficaz, pois permite condensar discussões complexas sobre gestão de ativos de TI, integração de sistemas e expectativas de *stakeholders* em um fluxo estruturado e evitando os riscos de escopo inflado, comuns em projetos de software empresarial.

O cerne da *Lean Inception* reside em três princípios: (1) a colaboração radical, que envolve desde desenvolvedores até usuários finais em todas as etapas; (2) a validação contínua, com hipóteses testadas rapidamente por meio de artefatos visuais (como canvas e jornadas do usuário); e (3) a ênfase no valor entregue, descartando funcionalidades supérfluas desde o início.

O método organiza-se em fases sequenciais (Figura 5), cada uma com entregas tangíveis. A primeira etapa, a Definição da Visão do Produto, estabelece o propósito macro do *ITAM*. Em seguida, a identificação de Personas e Jornadas do Usuário revela cenários fundamentais para a solução do produto.

Nas fases seguintes, priorizam-se funcionalidades essenciais por meio de técnicas como o sequenciador, que promove apenas o indispensável ao MVP. Por fim, o Canvas MVP consolida essas decisões em um plano visual, que serve como bússola para o desenvolvimento iterativo.

Figura 5 – Agenda de um *Lean Inception*.



Fonte: CAROLI (2024).

A aplicação desta metodologia no projeto garantirá o alinhamento claro entre equipe e stakeholders, um MVP focado em funcionalidades essenciais e, por fim, a redução de retrabalho nas fases posteriores de desenvolvimento.

4.2 Fases da *Lean Inception* Aplicadas ao Projeto

4.2.1 1ª Etapa: Definição da Visão

A primeira fase da *Lean Inception* é essencial para estabelecer um entendimento comum sobre o escopo do produto e consiste no *Kick-off* do *workshop*, onde são alinhados: Os três primeiros artefatos críticos: Visão do Produto, Quadro É/Não É/Faz/Não Faz e Objetivos do Produto.

4.2.1.1 Visão do Produto

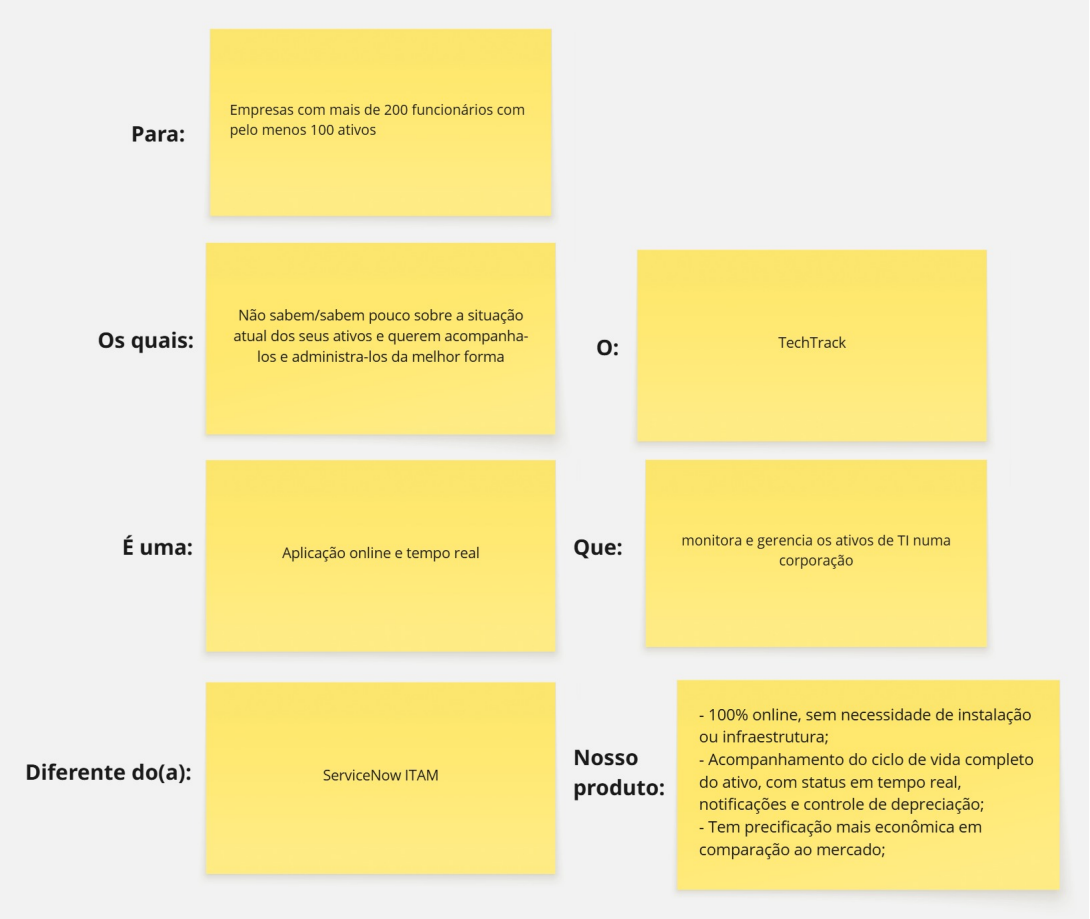
A Visão do Produto, construída coletivamente durante o *workshop*, segue uma estrutura padronizada para definir o valor central da solução. Essa estrutura é guiada por sete elementos principais:

1. **Para**: identifica o público-alvo do produto.
2. **Os quais**: descreve o problema enfrentado por esse público.
3. **O**: apresenta o nome da solução proposta.

4. **É um**: define a natureza da solução.
5. **Que**: especifica as funcionalidades ou ações da solução.
6. **Diferente do(a)**: aponta os concorrentes existentes no mercado.
7. **Nosso produto**: destaca o diferencial competitivo da solução.

Desta forma, como visão de produto idealizada para o produto, foi definida a seguinte declaração que sintetiza o diferencial competitivo do projeto e direciona as próximas fases do workshop (Figura 6):

Figura 6 – Visão do produto



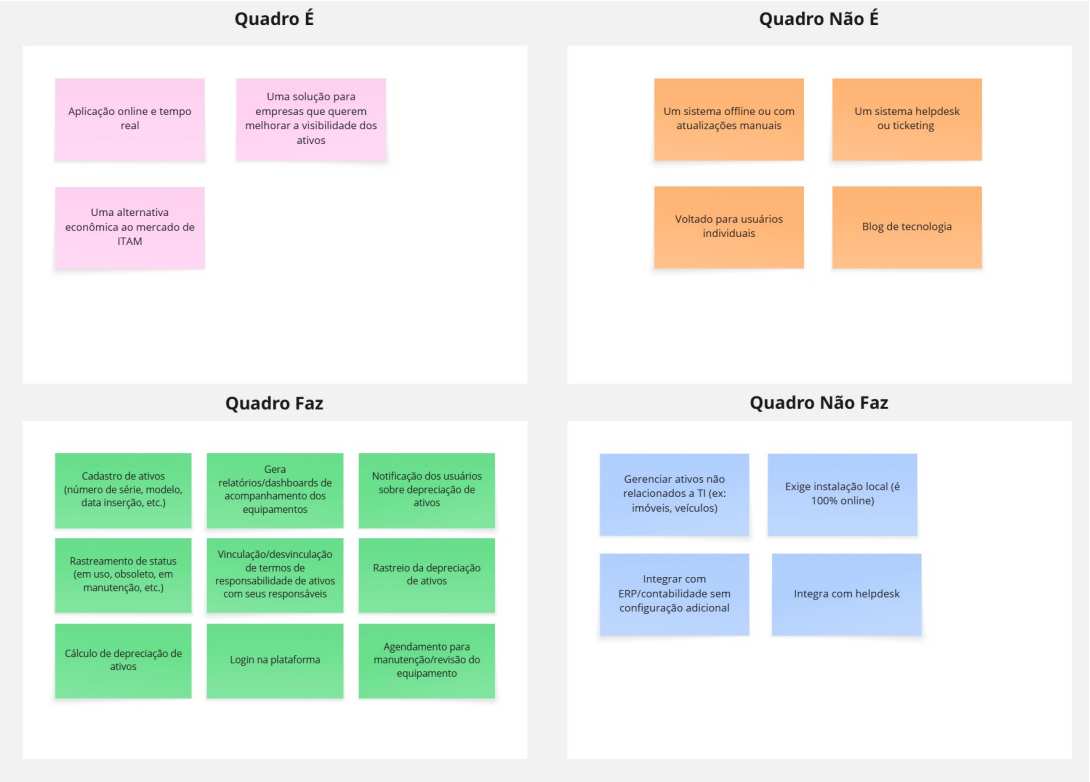
Fonte: Autoria própria.

4.2.1.2 Quadro É/Não É - Faz/Não Faz

O Quadro É/Não É - Faz/Não Faz é um instrumento fundamental para delimitação precisa do escopo do projeto. Sua eficácia reside na dualidade que apresenta: enquanto define claramente o que o produto **É/Faz** (aspectos geralmente consolidados na equipe), explicita também o que **Não É/Não Faz**, evitando mal-entendidos e escopo inflado.

Para o desenvolvimento do projeto, este artefato foi construído considerando as perspectivas técnicas da equipe de desenvolvimento, os requisitos regulatórios dos *stakeholders* e as limitações identificadas durante o *workshop*. Como resultado, obteve-se a seguinte estrutura na Figura 7:

Figura 7 – Quadro É/Não É - Faz/Não Faz.



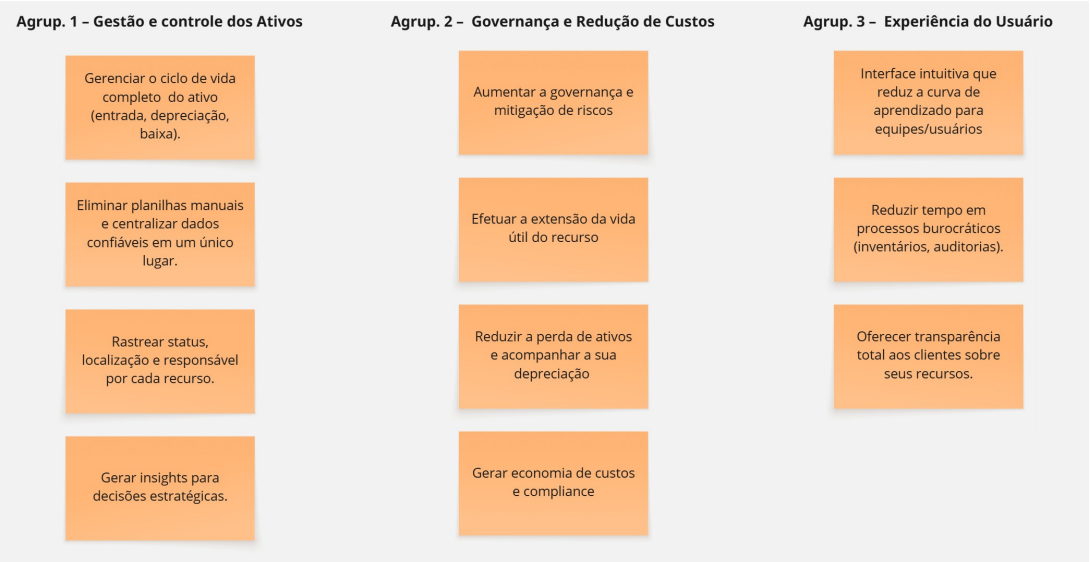
Fonte: Autoria própria.

4.2.1.3 Objetivos do Produto

A definição dos objetivos do produto constitui uma etapa fundamental no processo de *Lean Inception*, pois estabelece os resultados estratégicos que a solução deve alcançar. Esses objetivos representam as metas de negócio mensuráveis que justificam o desenvolvimento do produto, alinhando expectativas entre equipe técnica e *stakeholders*.

Durante o *workshop*, após a discussão e elicitação dos objetivos, estes são organizados por agrupamentos de objetivos dentro do mesmo escopo, como pode ser ilustrado na Figura 8:

Figura 8 – Objetivos do Produto.



Fonte: Autoria própria.

Após a conclusão do desenvolvimento e implantação do produto, é possível afirmar que a grande maioria dos objetivos de produto elencados durante a *Lean Inception* foi plenamente atingida. No entanto, um objetivo específico do **Agrupamento 1 – Gestão e Controle dos Ativos** não foi implementado em sua totalidade, conforme a análise a seguir.

O objetivo de **"Gerenciar o ciclo de vida completo do ativo (entrada, depreciação, baixa)"** foi parcialmente atendido. A aplicação web efetivamente gerencia a entrada (cadastro) e a baixa de ativos, contemplando a maior parte do ciclo de vida. Contudo, o módulo de **cálculo automatizado de depreciação** não foi desenvolvido, ficando como uma funcionalidade proposta para trabalhos futuros. A depreciação é um processo contábil complexo que demanda a integração com métodos fiscais (como Linha Reta ou Saldo Decrescente) e a consolidação de relatórios específicos, o que exigiria um escopo e um prazo adicionais que extrapolavam os limites deste projeto.

Quanto aos demais objetivos, todos foram atingidos com sucesso. No Agrupamento 1, o produto elimina planilhas manuais ao centralizar todos os dados dos ativos, permite o rastreamento completo de status, localização e responsável, e gera insights através de seus relatórios e dashboards. No Agrupamento 2, a governança é aumentada pelo controle centralizado e pelo histórico de movimentações, enquanto a extensão da vida útil é facilitada pelo acompanhamento de manutenções e condições do ativo, o que, somado à redução de perdas, gera economia e auxilia no compliance. Por fim, no Agrupamento 3, a interface foi desenvolvida priorizando a intuição e a usabilidade, reduzindo o tempo despendido em processos burocráticos de inventário e auditoria, e oferecendo transparência às partes interessadas.

Portanto, considerando o escopo definido, o produto entregue atende de forma robusta e eficaz aos seus objetivos principais, ficando o cálculo de depreciação como uma evolução natural e de grande valor para versões subsequentes.

4.2.2 2ª Etapa: Personas e Jornadas de Usuário

A segunda fase da *Lean Inception* dedica-se à elaboração de Personas e Jornadas do Usuário, artefatos essenciais para compreender profundamente os usuários finais e antecipar suas interações com a solução proposta. Nesta etapa é mapeado perfis reais de usuários, seus comportamentos, necessidades específicas e pontos de dor no fluxo atual de trabalho.

4.2.2.1 Personas

Persona é uma representação fictícia do cliente ideal, construída a partir de dados reais sobre o comportamento e as características demográficas dos seus consumidores. Além disso, inclui elementos como histórias pessoais, motivações, objetivos, desafios e preocupações, ajudando a entender melhor quem é o público que se deseja alcançar. Cada Persona foi documentada com os seguintes elementos:

1. Nome e perfil demográfico, como cargo e tempo de experiência;
2. Objetivos profissionais;
3. Frustrações com soluções atuais, como o uso de planilhas manuais para inventário;
4. Expectativas em relação à nova aplicação web.

Essa abordagem adaptada assegurou que o desenvolvimento do *ITAM* mantivesse foco nas reais demandas dos usuários finais.

4.2.2.2 Jornada do Usuário

Além disso, as jornadas do usuário descrevem os processos atuais e ideais na interação com o produto. Ao longo dessa jornada, são destacadas as funcionalidades da solução e sua contribuição positiva para a experiência do usuário.

4.2.2.2.1 Persona e jornada de usuário 1

A 1ª Persona apresentada é o Carlos, um profissional de 40 anos, gerente de TI em uma empresa de médio porte com cerca de 500 funcionários, que valoriza precisão e *compliance* na gestão de ativos tecnológicos. Porém, devido à descentralização dos equipamentos em múltiplas filiais e à dependência de planilhas manuais, enfrenta desafios para

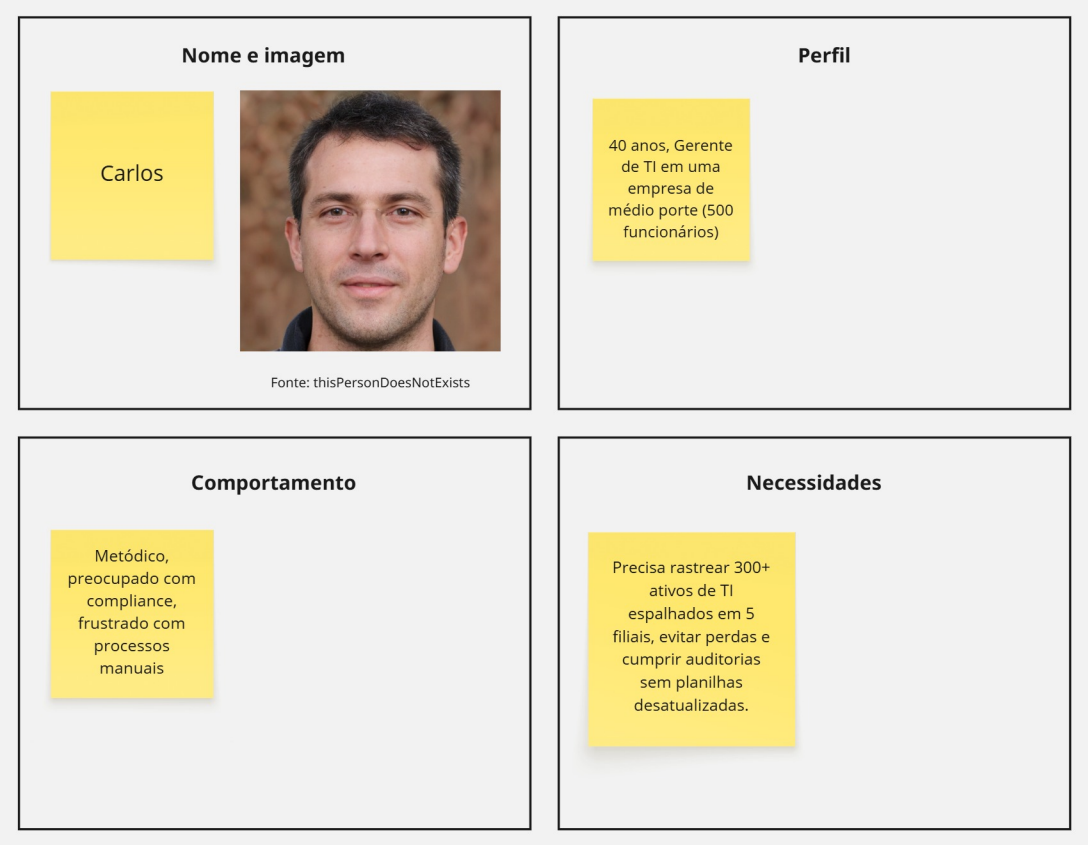
manter o controle preciso sobre os mais de 300 ativos de TI sob sua responsabilidade (Figura 9).

O Carlos representa um perfil estratégico para o público-alvo do GuardItens, pois está entre os gestores que necessitam de soluções robustas para rastreamento de ativos, conformidade com políticas internas e otimização de custos. Sua rotina exige ferramentas que permitam monitorar em tempo real a localização e status dos equipamentos, gerar relatórios precisos para auditorias e tomar decisões baseadas em dados sobre renovação ou baixa de ativos.

Nesse contexto, a Jornada de Usuário do Carlos demonstra como o aplicativo web GuardItens transforma sua gestão estratégica de ativos. Ao receber o alerta do CFO sobre custos excessivos, ele utiliza o dashboard integrado da aplicação para identificar oportunidades imediatas: 20% dos ativos subutilizados que podem ser redistribuídos e 5 servidores com depreciação crítica que exigem ação. Com base nesses *insights*, Carlos toma decisões ágeis: substitui compras desnecessárias pela realocação de equipamentos ociosos e prioriza a revisão técnica dos ativos mais depreciados

Dessa forma, obtém redução comprovada nos custos de TI no próximo trimestre, com transparência total perante a diretoria. O GuardItens não só fornece os dados, mas estrutura o processo decisório, seja do diagnóstico à execução (Figura 10).

Figura 9 – Persona 1.



Fonte: Autoria própria.

Figura 10 – User Story 1.



Fonte: Autoria própria.

4.2.2.2.2 Persona e jornada de usuário 2

Joana Silva, 28 anos, é analista de suporte de TI na mesma empresa onde Carlos atua como gerente. Enquanto ele se preocupa com estratégia, *compliance* e redução de

custos, Joana lida diretamente com a movimentação dos ativos, o registro de responsabilidades e a resolução de conflitos no dia a dia. Ela é meticulosa, sabe que detalhes fazem a diferença e está sempre atenta para evitar falhas que possam cair no sua jurisdição depois (Figura 11).

Seu maior desafio: provar quem realmente está com um equipamento quando algo dá errado. Antes do GuardItens, ela dependia de planilhas compartilhadas, e-mails perdidos e termos de responsabilidade em papel, quando existiam. Muitas vezes, um funcionário alegava nunca ter recebido um notebook caro, e não havia como comprovar o contrário. A empresa acabava arcando com o prejuízo, e Joana ficava frustrada por não ter ferramentas para resolver essas situações de forma limpa.

Agora, com o GuardItens sempre que um equipamento é entregue, ela registra: o termo de responsabilidade assinado digitalmente, com data, hora e localização e fotos para comprovação do estado do equipamento anexadas ao registro, evitando disputas futuras sobre danos.

Recentemente, um colaborador do setor de *marketing* insistiu que nunca havia recebido um *tablet* de última geração. No passado, isso significaria horas de investigação, emails cruzados e, no fim, a empresa provavelmente reporia o equipamento sem ter certeza do que realmente aconteceu. Mas, com o GuardItens, Joana resolveu a questão em minutos:

1. **Puxou o histórico do ativo**: o *tablet* havia sido formalmente entregue ao setor do colaborador três meses antes;
2. **Exibiu o termo de responsabilidade**: assinado digitalmente pelo próprio funcionário, com registro de IP e localização.

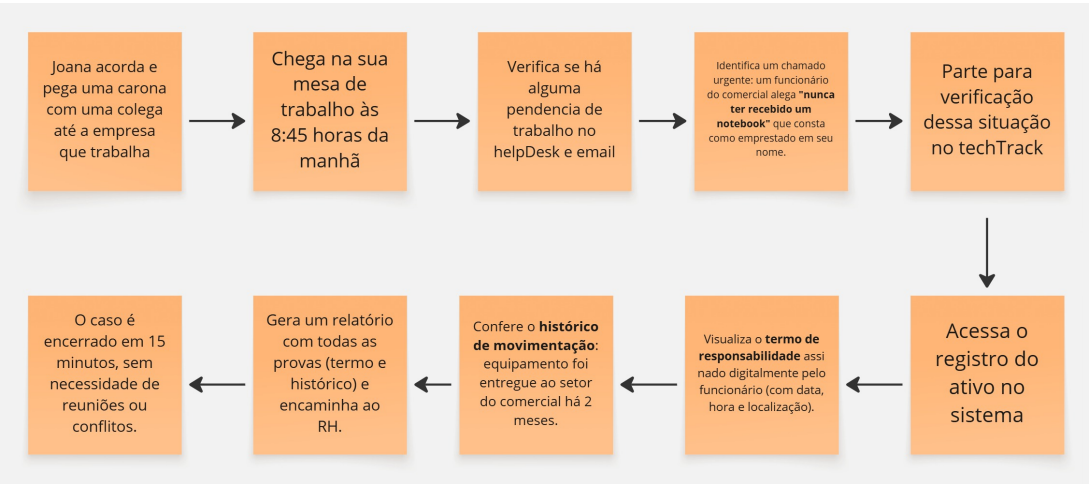
O caso foi encerrado rapidamente, sem discussões. O RH foi notificado e o colaborador assumiu a responsabilidade. Para Joana, isso significou menos estresse, menos retrabalho e a certeza de que a aplicação a apoia em situações que antes dependiam apenas da sua palavra contra a do usuário (Figura 12).

Figura 11 – Persona 2.



Fonte: Autoria própria.

Figura 12 – User Story 2.



Fonte: Autoria própria.

4.2.2.2.3 Persona e jornada de usuário 3

A 3ª Persona identificada é o Rafael Costa, um auditor de TI de 35 anos que vive para desvendar inconsistências em sistemas corporativos (Figura 13).

Durante seus finais de semana, Rafael é um fotógrafo amador que adora registrar paisagens urbanas, hobby que exige a mesma precisão que seu trabalho: "Se a foto não tem o enquadramento perfeito, não vale."

Contudo, quando assume auditorias em empresas com gestão caótica de ativos, enfrenta dias frustrantes. Planilhas desatualizadas, colaboradores que "esquecem"termos de responsabilidade e equipamentos que desaparecem sem deixar rastro consomem seu tempo e paciência.

Nessa circunstância, a Jornada de Usuário do Rafael demonstra como o GuardItens transforma processos de auditoria antes caóticos em fluxos eficientes. Isso acontece porque Rafael não precisa mais perder tempo cruzando planilhas manuais ou investigando inconsistências sem provas. Em vez disso, ele acessa um painel centralizado com: Histórico imutável de todas as movimentações de ativos; Termos de responsabilidade digitalmente assinados; Alertas automáticos de irregularidades (como equipamentos prestes a depreciar).

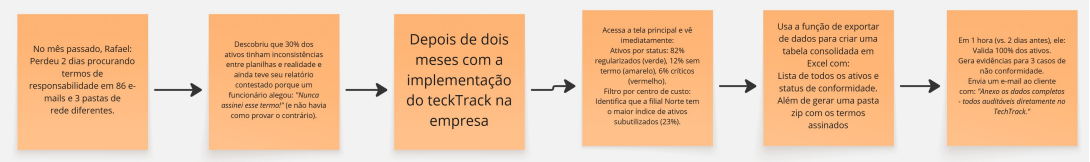
Enquanto revisa esses dados em tempo real, ele consegue até mesmo aproveitar pausas para seu hobby fotográfico. Sabendo que o sistema garante a rastreabilidade que sua profissão exige (Figura 14).

Figura 13 – Persona 3.



Fonte: Autoria própria.

Figura 14 – User Story 3.



Fonte: Autoria própria.

4.2.3 3ª etapa: Brainstorm de Funcionalidades e Revisão das Funcionalidades

A terceira fase da *Lean Inception* concentra-se na especificação detalhada das funcionalidades do produto, combinando criatividade e critérios estratégicos. Esta etapa é dividida em dois momentos chave: brainstorm de Funcionalidades com a geração livre e colaborativa de ideias e a revisão técnica que conta com avaliação de viabilidade e alinhamento com os objetivos do MVP.

4.2.3.1 Brainstorm de Funcionalidades Iniciais

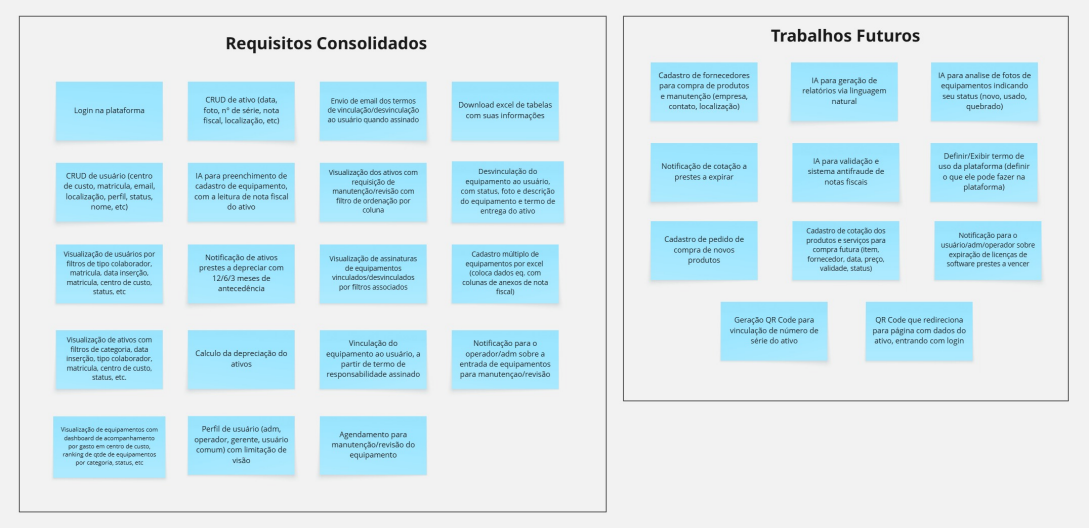
Nesta etapa, os integrantes do projeto realizaram um *brainstorming* colaborativo para levantar funcionalidades potenciais do *ITAM*. Foram consolidadas 30 ideias de funcionalidades após a remoção de duplicatas e discussão coletiva.

O Brainstorm de Funcionalidades na *Lean Inception* é uma etapa essencial em que cada participante, primeiro individualmente, propõe o maior número possível de ideias para as funcionalidades do produto. Em seguida, todas as sugestões são debatidas para definir quais permanecerão no escopo e quais serão descartadas.

Essa abordagem é altamente eficaz para estimular a geração de novas funcionalidades, que depois são filtradas com base em critérios como alinhamento com os objetivos de negócio, necessidades das personas e etapas da jornada do usuário.

Dessa forma, foram identificadas 30 funcionalidades distintas, sendo 19 selecionadas para o escopo inicial e 11 classificadas como propostas para trabalhos futuros (Figura 15).

Figura 15 – Brainstorm de Funcionalidades



Fonte: Autoria própria.

4.2.3.2 Revisão e Evolução Definitiva

Durante o desenvolvimento do projeto, o escopo de funcionalidades passou por uma reavaliação baseada na viabilidade técnica, complexidade de implementação e feedback dos usuários. O brainstorm foi revisitado e atualizado, resultando no escopo final apresentado na Figura 16.

As principais alterações em relação ao escopo original do TCC 1 podem ser categorizadas em três grupos principais:

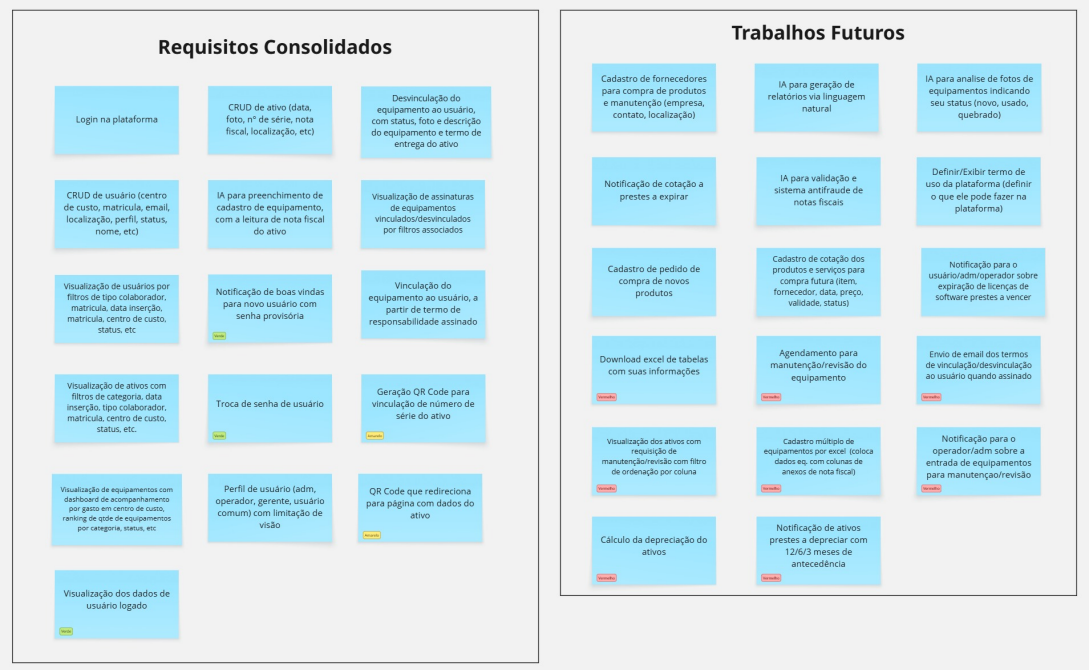
Remoções do Escopo: Foram identificadas funcionalidades que, após análise mais aprofundada, foram consideradas de menor prioridade ou complexidade excessiva para o escopo atual. Estas foram remanejadas para trabalhos futuros, estando demarcadas com a tag vermelha no quadro direito da Figura

Adições ao Escopo: Duas novas funcionalidades foram incorporadas ao produto a partir do consenso entre a equipe de desenvolvimento e o cliente envolvido, visando atender necessidades identificadas durante a fase de implementação. Sendo demarcadas pela tag verde do quadro esquerdo da Figura.

Promoções de Escopo: Dois itens originalmente classificados como trabalhos futuros foram elevados para o escopo atual do TCC 2, por serem considerados essenciais para o valor mínimo do produto após reavaliação estratégica. Sendo demarcadas pela tag amarela do quadro esquerdo da Figura.

Esta reestruturação permitiu entregar um produto mais alinhado com as necessidades reais dos usuários, contendo 16 funcionalidades consolidadas e mantendo o foco nos itens de maior valor para o negócio dentro do prazo disponível.

Figura 16 – Brainstorm de Funcionalidades - Escopo Final



Fonte: Autoria própria.

4.2.3.3 Revisão das Funcionalidades

A sessão de Revisão das Funcionalidades tem como objetivo estimular a equipe a analisar detalhadamente a implementação de cada funcionalidade, considerando não apenas aspectos técnicos, mas também o valor de negócio e a experiência do usuário.

Nessa atividade, cada membro do time avalia seu nível de confiança em relação ao entendimento do "o que fazer"(escopo) e do "como fazer"(implementação) para cada funcionalidade. Em seguida, a equipe busca um consenso para classificar a atividade em uma das seguintes categorias: Verde (fácil); Amarelo (médio); Vermelho (difícil).

Além disso, cada funcionalidade é classificada segundo três critérios: Esforço técnico (E); Valor de negócio ($); Experiência do usuário (3).

Essa análise permitiu que a equipe tivesse uma visão clara das funcionalidades a serem priorizadas e dos desafios envolvidos no escopo inicial apresentado (Figura 17)

Figura 17 – Revisão das funcionalidades.



Fonte: Autoria própria.

4.2.3.4 Revisão Final das Funcionalidades

Com a evolução do projeto, a revisão de funcionalidades foi atualizada para refletir as mudanças no escopo e o maior entendimento técnico adquirido durante o desenvolvimento. A análise revisada (Figura 18) demonstra significativa maturação do projeto:

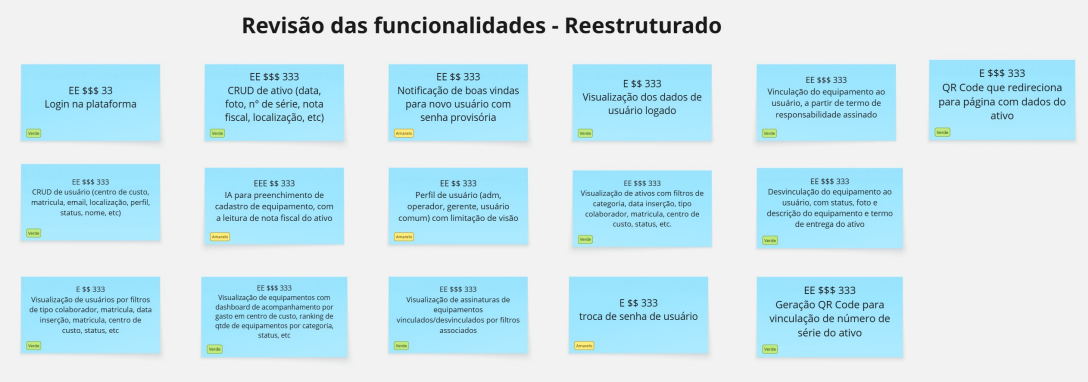
Evolução das Classificações: Funcionalidades anteriormente categorizadas como Vermelho (difíceis) foram requalificadas para Amarelo ou Verde, indicando maior domínio técnico da equipe e refinamento dos requisitos.

Reavaliação de Prioridades: O valor de negócio e a experiência do usuário foram reavaliados com base nos feedbacks coletados durante a implementação, resultando em ajustes nas prioridades de desenvolvimento.

Redução de Riscos: O esforço técnico foi recalibrado considerando a arquitetura implementada e as lições aprendidas, permitindo uma estimativa mais precisa do trabalho restante.

Esta revisão atualizada forneceu uma base sólida para o planejamento das sprints finais, garantindo o foco nas funcionalidades de maior impacto dentro do cronograma do trabalho.

Figura 18 – Revisão das Funcionalidades - Escopo Final



Fonte: Autoria própria.

4.2.4 4ª etapa: Sequenciador e Canva MVP

Na quarta e última etapa da Lean Inception, ocorre a priorização das funcionalidades ao longo do tempo e uma apresentação resumida do MVP (Produto Mínimo Viável). Para isso, são utilizadas duas ferramentas principais:

- **O Sequenciador**: Define a ordem de implementação das funcionalidades, considerando dependências, valor de negócio e esforço técnico.
- **O Canvas MVP**: Resume visualmente as principais características do MVP, alinhando expectativas entre equipe e stakeholders.

Essa etapa consolida o planejamento do produto, garantindo que o MVP seja viável, entregue valor incremental e esteja alinhado com os objetivos estratégicos.

4.2.4.1 Sequenciador

O Sequenciador é uma ferramenta estratégica que auxilia na definição da ordem de implementação das funcionalidades do produto, estabelecendo quais devem ser desenvolvidas primeiro e quais serão entregues posteriormente. Além disso, ele ajuda a delimitar o escopo do MVP (Produto Mínimo Viável) e identifica quais funcionalidades serão tratadas como incrementos futuros. Ele adota como estrutura básica:

- **Ondas**: Divisões temporais
- **Cartões**: Funcionalidades classificadas por:
  - Vermelho = Alta complexidade
  - Amarelo = Média complexidade

– Verde = Baixa complexidade

Para assegurar um cronograma realista e bem-estruturado, o sequenciador estrutura 6 regras bem definidas:

- **Regra 1**: Cada onda pode conter no máximo três funcionalidades.
- **Regra 2**: Não pode haver mais de uma funcionalidade vermelha (difícil) por onda.
- **Regra 3**: Uma onda não pode ter três cartões amarelos e vermelhos combinados.
- **Regra 4**: A soma do esforço técnico (E) dos cartões não pode passar de 5 E's por onda.
- **Regra 5**: A soma do valor de negócio ($) e da experiência do usuário (3) deve ser pelo menos 4 em cada categoria por onda.
- **Regra 6**: Se um cartão depende de outro, este deve estar em uma onda anterior.

No contexto deste projeto, o Sequenciador foi dividido em 6 ondas, cada uma representando um período de tempo específico. Por meio da colaboração entre os participantes, as principais funcionalidades foram definidas para serem implementadas nas primeiras 3 fases, garantindo que o aplicativo tenha uma base sólida para incorporar as demais funcionalidades (Figuras 21 e 22)

Figura 19 – Sequenciador. Ondas 1, 2 e 3.



Fonte: Autoria própria.

Figura 20 – Sequenciador. Ondas 4, 5 e 6.



Fonte: Autoria própria.

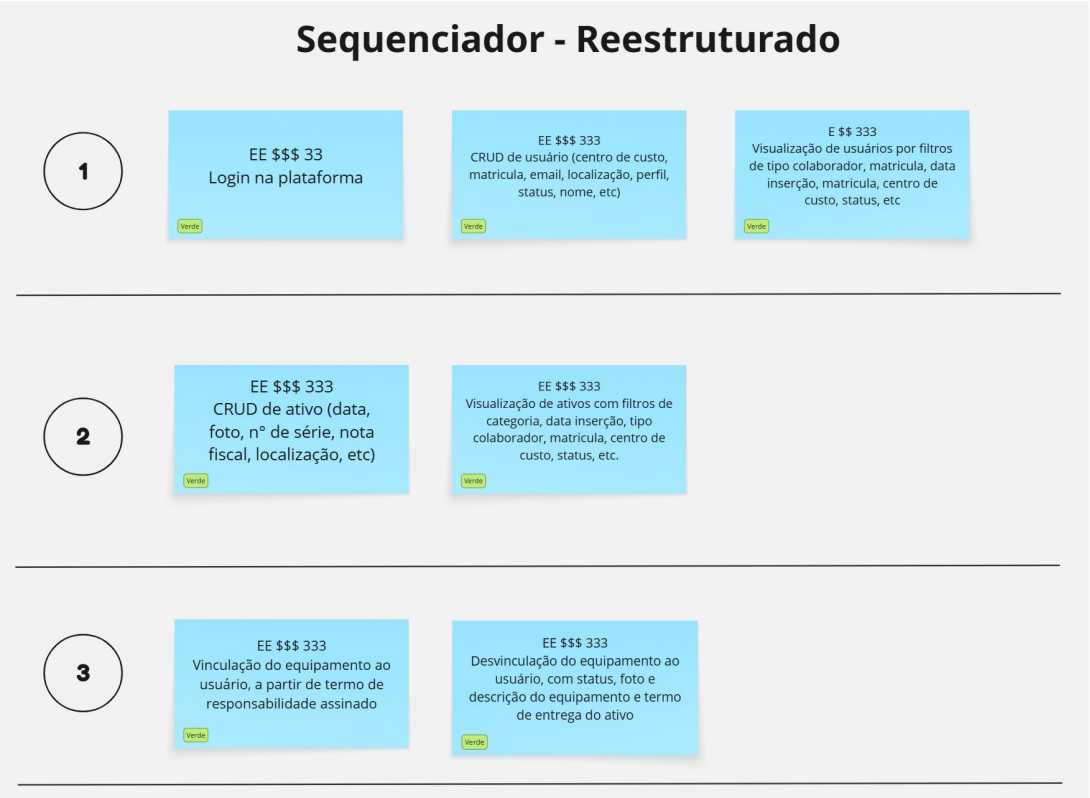
4.2.4.1.1 Sequenciador - Readequação das Ondas de Desenvolvimento

O Sequenciador manteve sua estrutura original de 6 ondas de desenvolvimento, porém com significativa readequação na distribuição das funcionalidades entre as fases. Esta reestruturação foi necessária para refletir as novas prioridades identificadas durante o desenvolvimento do projeto e otimizar o fluxo de implementação. A ferramenta continuou utilizando a mesma metodologia de classificação por cores para complexidade: vermelho (alta complexidade), amarelo (média complexidade) e verde (baixa complexidade), seguindo as seis regras estabelecidas inicialmente.

A principal alteração no sequenciamento consistiu na redistribuição estratégica das funcionalidades entre as ondas, visando uma otimização técnica e reavaliação do valor de negócio. Para a otimização técnica, foram antecipadas funcionalidades críticas para a arquitetura do sistema, postergadas features com dependências externas complexas e agrupadas funcionalidades relacionadas para ganho de eficiência. Na reavaliação de valor de negócio, priorizaram-se funcionalidades com maior impacto imediato para os usuários, reclassificando-as com base nos feedbacks coletados durante o desenvolvimento e ajustando as métricas de valor de negócio e experiência do usuário.

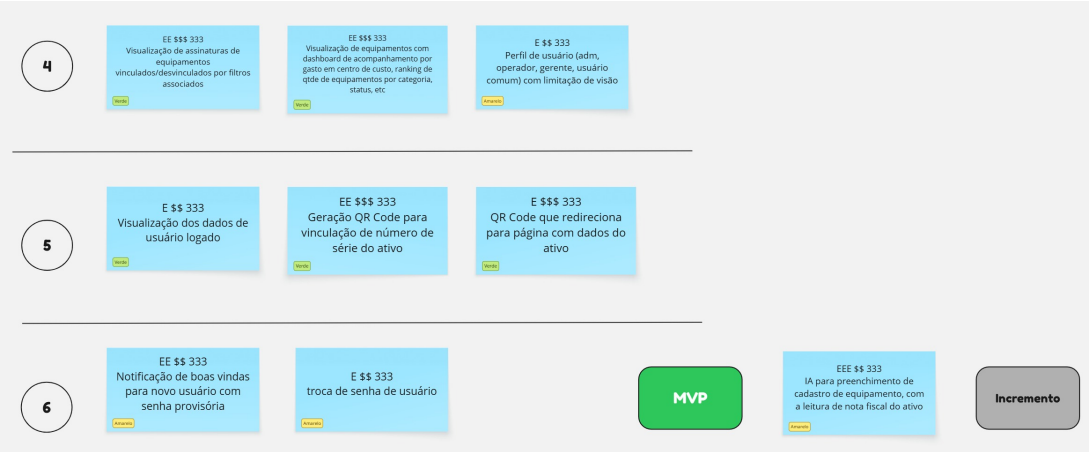
As Figuras 21 e 22 ilustram o sequenciamento definitivo adotado no trabalho, demonstrando visualmente a reordenação realizada. Esta readequação permitiu entregar um produto com funcionalidades mais alinhadas às necessidades reais dos usuários, mantendo a viabilidade técnica dentro do cronograma. A flexibilidade demonstrada no replanejamento do sequenciador reflete a maturidade do processo de desenvolvimento e a capacidade de adaptação baseada nas lições aprendidas durante a implementação prática da aplicação web.

Figura 21 – Sequenciador - Ondas 1, 2 e 3 (Versão reestruturada)



Fonte: Autoria própria.

Figura 22 – Sequenciador - Ondas 4, 5 e 6 (Versão reestruturada)



Fonte: Autoria própria.

4.2.4.2 Canva MVP

O Canvas MVP representa o artefato final no processo da Lean Inception, servindo como ferramenta estratégica para consolidar visualmente a estrutura do Produto Mínimo Viável. Este canvas sintetiza em um único quadro todos os elementos essenciais que com-

põem a versão inicial do produto, permitindo validar rapidamente as premissas e alinhar as expectativas entre equipe e stakeholders. (Figuras 24)

O principal objetivo do Canvas MVP é criar uma visão compartilhada e objetiva sobre o que será desenvolvido na primeira versão do produto. Ele atua como um guia visual que: define claramente os limites do MVP, stabelece as funcionalidades essenciais, identifica as métricas de sucesso e documenta as principais hipóteses a serem validadas.

Ele é organizado em seções lógicas que abrangem tanto aspectos de negócio quanto técnicos. A estrutura típica inclui:

- **Proposta de Valor**: A razão fundamental pela qual o MVP existe
- **Personas Alvo**: Os principais usuários que o MVP pretende atender
- **Funcionalidades Críticas**: As features indispensáveis para a versão inicial
- **Métricas de Sucesso**: Como será medido o desempenho do MVP
- **Restrições e Limitações**: Fatores técnicos ou de negócio que impactam o desenvolvimento

Figura 23 – Canva MVP.



Fonte: Autoria própria.

4.2.4.2.1 Canvas MVP - Evolução e Definição Final

O Canvas MVP constitui o artefato final no processo de Lean Inception, servindo como ferramenta estratégica para consolidar visualmente a estrutura do Produto Mínimo Viável. No contexto do trabalho, este canvas passou por significativas atualizações em

relação à versão inicial de projeto, refletindo o amadurecimento do entendimento sobre o produto e as reais necessidades do negócio. Esta evolução permitiu validar rapidamente as premissas estabelecidas e realinhar as expectativas entre equipe e *stakeholders* com base nas experiências adquiridas durante o desenvolvimento.

O principal objetivo do Canvas MVP revisado foi criar uma visão compartilhada e objetiva sobre o que seria efetivamente desenvolvido na versão final do produto. Este guia visual atuou como documento norteador que define claramente os limites do MVP, estabelece as funcionalidades essenciais após as reavaliações de escopo, identifica as métricas de sucesso ajustadas à realidade do projeto e documenta as hipóteses validadas durante o processo de desenvolvimento.

A estrutura organizacional do Canvas MVP mantém-se em seções lógicas que abrangem aspectos de negócio e técnicos, porém com conteúdo atualizado. As funcionalidades críticas sofreram readequação considerando a viabilidade técnica e o valor de negócio demonstrado durante a implementação, mantendo os outros componentes do canva associado.

A Figura 24 apresenta o Canvas MVP definitivo que orientou o desenvolvimento do projeto, incorporando todas as alterações realizadas em relação ao planejamento inicial. Dentre as principais modificações destacam-se a inclusão de funcionalidades identificadas como críticas durante o processo de desenvolvimento, a exclusão de features que se mostraram de menor prioridade após análise mais aprofundada. Esta versão final do canvas representa fielmente o produto entregue e serve como documento de referência para futuras evoluções do produto.

Figura 24 – Canvas MVP - Versão Final



Fonte: Autoria própria.

4.3 Desenvolvimento do produto

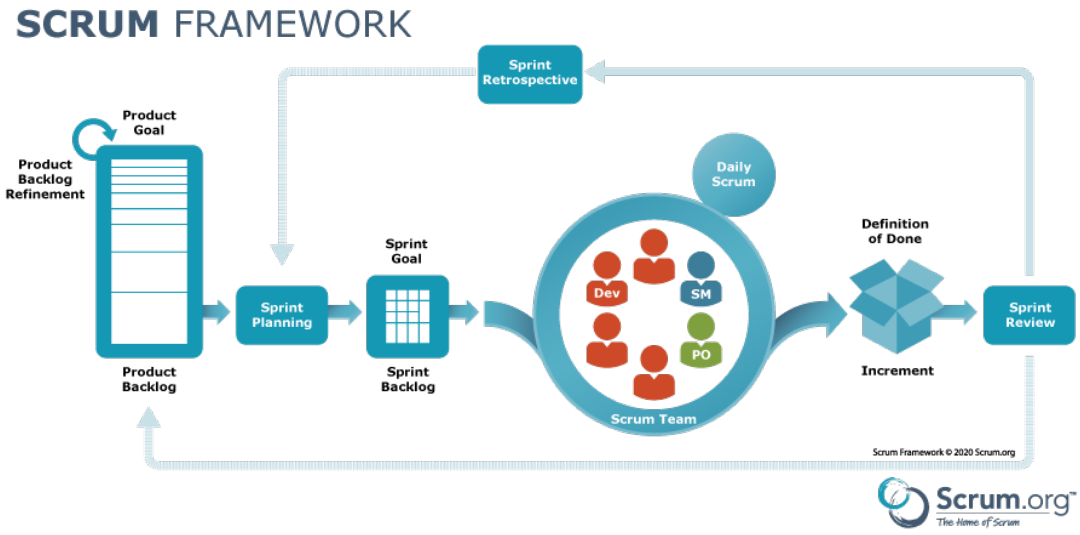
Após o levantamento de requisitos, é essencial adotar uma abordagem de desenvolvimento ágil, organizada e eficiente. Isso permite que a equipe responda rapidamente a mudanças e entregue valor contínuo ao cliente. Para este fim, este trabalho adotará o Scrum e o Kanban como pilares do desenvolvimento.

4.3.1 Scrum

O Scrum é um *framework* ágil utilizado para gerenciar o desenvolvimento de produtos complexos, enfatizando a entrega iterativa e incremental de valor. Ele é baseado em princípios de transparência, inspeção e adaptação, permitindo que as equipes respondam a mudanças de maneira eficiente e entreguem resultados alinhados às necessidades do cliente (SCHWABER; SUTHERLAND, 2020).

Sua estrutura se destaca pela alta adaptabilidade e pela promoção de ciclos curtos de trabalho, conhecidos como Sprints, os quais garantem a entrega frequente de funcionalidades e possibilitam a melhoria contínua do produto, característica representada schematicamente na Figura 25.

Figura 25 – Framework Scrum.



Fonte: SCHWABER e SUTHERLAND (2020)

No desenvolvimento do aplicativo web GuardItens, a adoção do Scrum mostra-se particularmente vantajosa devido à sua estrutura baseada em ciclos curtos e entregas incrementais. Essa abordagem permite fragmentar o projeto em partes menores e funcionais, facilitando não apenas a organização e o gerenciamento das atividades, mas também possibilitando uma validação contínua por parte dos stakeholders.

A cada Sprint, funcionalidades são disponibilizadas para revisão, o que viabiliza *feedbacks* constantes e ajustes ágeis no escopo, alinhando progressivamente o produto às reais necessidades do cliente. Dessa forma, o Scrum é definido por três papéis essenciais, cada um com responsabilidades específicas:

- ***Product Owner* (PO)**: É o responsável por maximizar o valor do produto resultante do trabalho do Time de Desenvolvimento. Suas principais atribuições incluem gerenciar e priorizar o Backlog do Produto, assegurando que ele esteja visível, claro e que todos compreendam os itens de trabalho a serem realizados.
- ***Scrum Master* (SM)**: Atua como um facilitador, garantindo que o Time siga aos princípios e práticas do Scrum. O *Scrum Master* remove impedimentos que possam atrapalhar o progresso da equipe e promove um ambiente de trabalho colaborativo e produtivo.
- **Time de Desenvolvimento (*Development Team*)**: É um grupo multifuncional e auto-gerenciável responsável por converter os itens do Backlog do Produto em incrementos de produto prontos a cada Sprint.

Além disso, o scrum cria ritmo e oportunidades regulares para inspeção e adaptação a partir de eventos específicos. Sendo descritos por:

- **Sprint**: O coração do Scrum. É um período de tempo fixo (*time-box*), normalmente de 1 a 4 semanas, no qual um incremento de produto "Pronto"(*Done*) é criado. As Sprints são consecutivas e não têm hiatus entre elas.
- **Planejamento da Sprint (Sprint Planning)**: Evento que marca o início de uma Sprint. Nele, o Time Scrum todo colabora para definir o que pode ser entregue na Sprint que se inicia e como esse trabalho será realizado.
- ***Daily Scrum* (Reunião Diária)**: Uma reunião curta (limitada a 15 minutos) realizada idealmente todos os dias no mesmo horário e local. Nela, cada membro do Time de Desenvolvimento comunica o que fez no dia anterior, o que fará hoje e se identifica algum impedimento. Seu foco é sincronizar atividades e planejar o trabalho das próximas 24 horas.
- **Revisão da Sprint (*Sprint Review*)**: Realizada ao final da Sprint, é uma sessão informal na qual o Time Scrum apresenta o incremento desenvolvido aos stakeholders. O objetivo é coletar feedback e promover discussões colaborativas para definir os próximos passos, adaptando o Backlog do Produto se necessário.
- **Retrospectiva da Sprint (Sprint Retrospective)**: É a oportunidade para o Time Scrum inspecionar a si próprio e criar um plano de melhorias para a próxima Sprint. A equipe discute o que funcionou bem, o que não funcionou e como pode se tornar mais eficaz, focando em processos e ferramentas.

Por fim, ele define os artefatos por trabalho ou valor e são projetados para promover transparência. Ilustrados por:

- **Backlog do Produto (Product Backlog)**: Uma lista dinâmica e priorizada de tudo que é necessário no produto, mantida e gerenciada pelo Product Owner. É uma ferramenta de planejamento que evolui constantemente conforme o produto e o mercado mudam.
- **Backlog da Sprint (Sprint Backlog)**: O conjunto de itens do Backlog do Produto selecionados para a Sprint, com um plano para entregar o incremento e atingir a Meta da Sprint.
- **Incremento**: A soma de todos os itens do Backlog do Produto concluídos durante uma Sprint e todas as Sprints anteriores. Cada incremento deve estar em um estado "Pronto"(de acordo com a Definition of Done acordada pela equipe), significando que é utilizável e potencialmente liberável.

O funcionamento cíclico do Scrum, com suas Sprints iterativas, promove um ritmo sustentável de trabalho, melhoria contínua e entrega frequente de valor, tornando-o uma escolha robusta para o desenvolvimento deste projeto.

Segundo SCHWABER e SUTHERLAND (2020) o Scrum é um framework adaptável por natureza, uma vez que se propõe a oferecer um conjunto de princípios e boas práticas, servindo como guia para o desenvolvimento de produtos. Considerando essa flexibilidade, o Scrum será adaptado no contexto deste projeto, que conta com um único cliente (atuando como Product Owner) e o autor do trabalho (assumindo simultaneamente os papéis de Scrum Master e desenvolvedor).

O ciclo de vida do produto, neste trabalho, inicia-se com a transposição do escopo para o Backlog do Produto, que servirá como referência central para o desenvolvimento. A partir desse artefato, priorizam-se as funcionalidades e realiza-se o Planejamento da Sprint, no qual são selecionadas as tarefas mais relevantes para compor o Backlog da Sprint. Ao final de cada ciclo (Sprint), conduz-se uma Revisão da Sprint com o cliente, a fim de validar as funcionalidades implementadas e coletar feedbacks para ajustes. Em seguida, um novo planejamento é realizado, dando início a um novo ciclo de desenvolvimento e garantindo a melhoria contínua e o alinhamento progressivo com as expectativas do cliente.

4.4 Kanban: Princípios, Funcionamento e Integração com Scrum

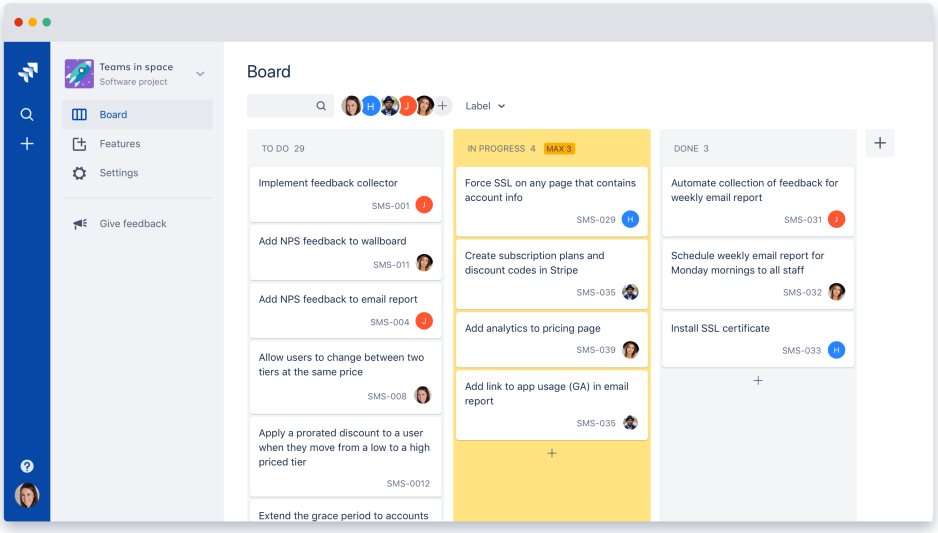
O Kanban é uma abordagem de gestão visual do trabalho originária do sistema de produção da Toyota, adaptada para contextos de desenvolvimento de software com foco na melhoria contínua e na eficiência do fluxo de trabalho. Diferente de frameworks prescritivos, o Kanban atua como um método evolutivo que permite às equipes gerenciar e melhorar gradualmente seus processos por meio de visualização clara, limitação do trabalho em andamento (Work In Progress - WIP) e análise de métricas de fluxo (Atlassian, 2023b).

Seu funcionamento baseia-se no uso de um quadro visual (Kanban Board), composto por colunas que representam os estágios do fluxo de trabalho – como "A Fazer", "Em Progresso"e "Concluído", como pode ser visualizado pela Figura 26. Cada tarefa ou história de usuário é representada por um cartão que se move entre essas colunas, proporcionando transparência imediata sobre o status do trabalho e possíveis gargalos. Um dos pilares centrais do Kanban é a limitação explícita do número de tarefas permitidas em cada coluna (WIP), evitando a sobrecarga da equipe e facilitando a identificação de impedimentos. Além disso, o Kanban enfatiza a melhoria contínua por meio da análise de métricas como lead time, cycle time e throughput, que permitem à equipe ajustar processos e aumentar progressivamente sua eficiência.

A integração entre Scrum e Kanban, denomiada de Scrumban, combina a estrutura iterativa e os papéis definidos do Scrum com a flexibilidade e a gestão visual do Kanban. Nesse modelo, o Scrum fornece a base para planejamento por sprints, reuniões de revisão e retrospectivas, enquanto o Kanban complementa com a visualização em tempo real do fluxo de trabalho, a limitação de WIP e a otimização contínua do processo. Essa combinação é especialmente vantajosa em contextos nos quais a equipe precisa responder a mudanças frequentes de prioridade ou lidar com demandas contínuas de manutenção e suporte, permitindo maior adaptabilidade sem abrir mão da rotina e da previsibilidade oferecidas pelo Scrum.

No contexto do desenvolvimento do aplicativo GuardItens, a adoção do Scrumban permitirá que o time, composto por uma única cliente (no papel de Product Owner) e o autor (atuando como Scrum Master e desenvolvedor), mantenha uma visão clara e atualizada do progresso, limite a quantidade de tarefas em execução simultânea para garantir foco e qualidade, e utilize métricas para avaliar e melhorar seu desempenho ao longo do projeto. Dessa forma, mesmo em um cenário com recursos limitados, a equipe poderá assegurar entregas constantes de valor, priorizando continuamente as demandas mais críticas e ajustando-se dinamicamente às necessidades da cliente.

Figura 26 – Método Kanban.



Fonte: Atlassian (2023b)

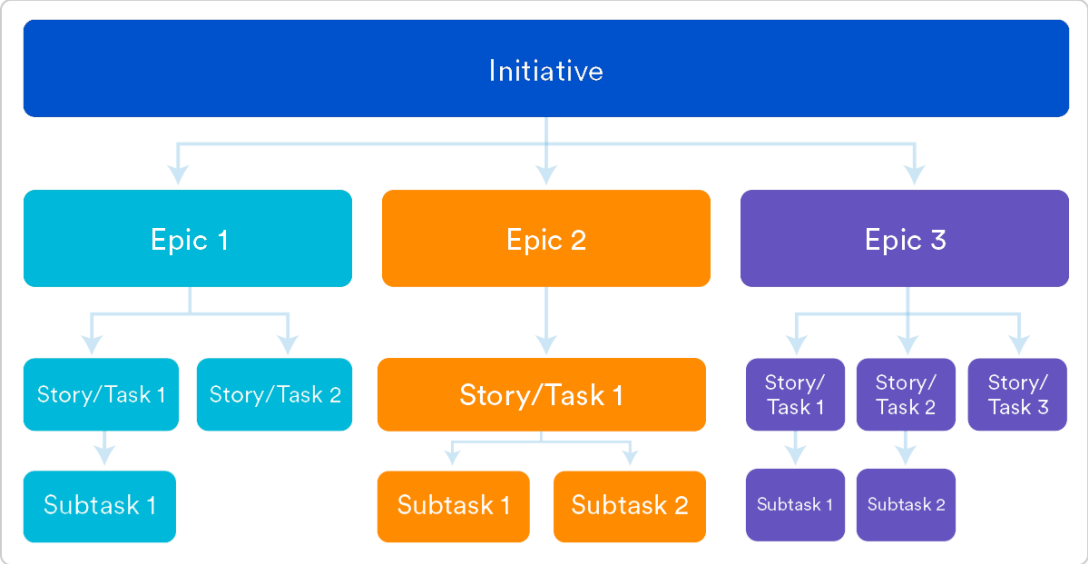
4.5 Iniciativas, Épicos e Histórias de Usuário

No gerenciamento de produtos ágeis, o trabalho é frequentemente organizado em camadas hierárquicas que facilitam o alinhamento entre a visão estratégica e a execução prática. De acordo com a Atlassian (2023c), essas camadas são comumente representadas por iniciativas, épicos e histórias de usuário, que juntos compõem uma estrutura escalável para quebrar objetivos complexos em partes executáveis, como pode ser ilustrado pela Figura 27.

- **Iniciativa**: Uma iniciativa é um esforço de alto nível que agrupa múltiplos épicos. Ela está diretamente vinculada aos objetivos estratégicos da organização e pode abranger vários times ou longos períodos de desenvolvimento.
- **Épico**: Um épico representa um grande corpo de trabalho que pode ser dividido em histórias de usuário menores. Ele geralmente engloba uma funcionalidade completa ou um conjunto de requisitos relacionados, mantendo-se gerenciável dentro de um roadmap de produto.
- **História de Usuário**: Como a unidade mais granular da estrutura ágil, a história de usuário descreve uma necessidade específica do usuário final, expressa em linguagem simples e focada no valor entregue. As histórias são a base do planejamento iterativo e da execução nas sprints.

Esse ecossistema permite que equipes mantenham o foco no valor de negócio, desde o nível macro (iniciativas) até o micro (histórias de usuário), assegurando que cada tarefa realizada contribua para os objetivos organizacionais mais amplos. A definição clara dessas camadas também favorece a rastreabilidade e a priorização do trabalho.

Figura 27 – Estrutura de épicos, histórias e iniciativas.



Fonte: Atlassian (2023c)

4.5.1 Histórias de usuário

Segundo a abordagem da Atlassian (2023c), uma história de usuário consiste na menor unidade de trabalho dentro de um contexto ágil. Ela representa um objetivo final, formulada a partir da perspectiva do usuário do software. Trata-se de uma descrição concisa e informal de uma funcionalidade, escrita com foco no valor entregue ao cliente, que pode ser tanto um usuário externo quanto um interno (como outro setor ou equipe da organização).

O propósito central de uma história de usuário é expressar de que maneira uma tarefa específica pode gerar valor definido para o cliente. Essas histórias são geralmente redigidas em linguagem simples, resumindo o resultado desejado sem detalhamento excessivo. Requisitos mais elaborados são incorporados posteriormente, após a discussão e o alinhamento da equipe.

Além disso, histórias de usuário integram-se de modo coerente a frameworks ágeis como Scrum e Kanban. No Scrum, são alocadas em sprints e progressivamente realizadas ao longo do ciclo. No Kanban, são gerenciadas em backlogs e avançam por meio de fluxos de trabalho contínuos. Essa prática contribui para a melhoria da estimativa e do planejamento no Scrum, e para o gerenciamento do trabalho em progresso (WIP) e o refinamento de fluxos no Kanban.

Por fim, as histórias de usuário funcionam como elementos fundamentais para estruturas ágeis de maior escala, como épicos e iniciativas. Épicos consistem em conjuntos de histórias relacionadas, enquanto iniciativas abrangem múltiplos épicos, assegurando que o trabalho de desenvolvimento esteja alinhado com os objetivos estratégicos da organização.

4.5.1.1 Trabalhando com Histórias de Usuário

Após a escrita de uma história de usuário, é essencial integrá-la adequadamente ao fluxo de desenvolvimento. Em geral, a história é redigida pelo Product Owner, gerente de produto ou programa, e subsequentemente submetida para revisão pela equipe.

Durante uma reunião de planejamento de sprint, a equipe seleciona as histórias que serão trabalhadas no ciclo iterativo. Nesse momento, discutem-se os requisitos e a funcionalidade necessários para implementar cada história, permitindo que a equipe explore abordagens técnicas e criativas para sua realização. Uma vez acordados, esses requisitos são formalmente incorporados à história.

Um passo frequente nessa etapa é a estimativa de esforço envolvido em cada história, com base em sua complexidade ou tempo previsto de conclusão. Técnicas comuns de estimativa incluem o uso de tamanhos de camisetas (P, M, G), a sequência de Fibonacci ou a metodologia de Planning Poker. Como uma história deve ser concluída dentro de uma sprint, aquelas que excedem esse limite são decompostas em unidades menores.

4.5.1.2 Boas Práticas para a Elaboração de Histórias de Usuário

Algumas considerações são fundamentais durante a redação de histórias de usuário:

- **Definição de Pronto**: É imprescindível estabelecer critérios claros que indiquem quando a história está concluída, geralmente quando o usuário final pode executar a tarefa proposta.
- **Subtarefas e Responsabilidades**: Identificar as etapas necessárias para implementação e atribuir responsáveis a cada uma delas.
- **Personas**: Considerar para quem a funcionalidade está sendo desenvolvida. Caso existam múltiplos usuários finais, pode ser necessário criar histórias específicas para cada persona.
- **Processos Complexos**: Para fluxos mais elaborados, recomenda-se escrever uma história para cada etapa do processo.
- **Feedback Contínuo**: Buscar actively feedback dos usuários reais, evitando suposições sobre suas reais necessidades.
- **Escopo Temporal**: Como histórias devem ser realizáveis dentro de uma sprint, aquelas com escopo muito amplo – que demandariam semanas ou meses – devem ser quebradas em histórias menores ou tratadas como épicos.

Uma vez definidas com clareza, as histórias de usuário devem ser tornadas visíveis e acessíveis a todos os membros da equipe, assegurando transparência e alinhamento contínuo

5 Proposta de solução

Este capítulo apresenta a proposta de solução desenvolvida para abordar a problemática central deste trabalho: a gestão ineficiente e descentralizada dos ativos de TI. A solução, denominada GuardItens, consiste na implementação de uma plataforma unificada de gerenciamento de ativos de TI. Seu objetivo principal é eliminar a perda de tempo em processos manuais de identificação, auditoria e controle do hardware e software, concedendo aos gestores e à equipe de TI clareza, controle e autonomia para focar em atividades estratégicas. Ao otimizar de forma inteligente a lifecycle dos ativos, a solução busca ampliar significativamente a visibilidade do patrimônio tecnológico, elevando simultaneamente a segurança, a conformidade e a eficiência operacional do departamento de TI.

A metodologia para o desenvolvimento da solução fundamenta-se na construção de um Backlog do Produto, estruturado em Épicos, Funcionalidades e Histórias de Usuário (User Stories). Adicionalmente, este capítulo exibe o Roadmap estratégico do produto e os protótipos de baixa fidelidade das interfaces da plataforma, os quais foram previamente validados pela cliente.

5.1 Backlog

De acordo com o Guia do Scrum, o Product Backlog consiste em uma lista dinâmica e prioritária de itens necessários para a evolução e o aprimoramento do produto (SCHWABER; SUTHERLAND, 2020). Ele funciona como um roteiro para a equipe Scrum, assegurando que as necessidades e expectativas da cliente sejam contempladas de maneira eficiente. Além disso, o Backlog passa por revisões e atualizações contínuas ao longo do ciclo de desenvolvimento, permitindo que as prioridades sejam reavaliadas com base no feedback da cliente e das partes interessadas (stakeholders), bem como nas mudanças nas demandas do negócio.

O Backlog do presente trabalho será detalhado por meio de Épicos e Histórias de Usuário (User Stories), derivadas das funcionalidades revisadas e definidas durante o workshop de Lean Inception, conforme ilustrado na Figura 17.

5.1.1 Épicos

O Backlog do Produto desenvolvido para este projeto é composto por 15 Histórias de Usuário (User Stories - US), organizadas em 4 Épicos distintos, representados na Tabela

1. Esse conjunto de itens definirá as funcionalidades do Produto Mínimo Viável (MVP) a ser implementado.

Tabela 1 – Épicos do Backlog

Número	**Épico**
1	Gerenciamento de Usuários e Acessos
2	Gestão do Ciclo de Vida dos Ativos
3	Operações de Vinculação e Desvinculação
4	Relatórios e Visualizações Analíticas

Fonte: Fonte: Autoria própria

1. **Gerenciamento de Usuários e Acessos:** Engloba funcionalidades de login, CRUD de usuários, visualização com filtros, troca de senha e notificações de boas-vindas.
2. **Gestão do Ciclo de Vida dos Ativos:** Inclui o cadastro de ativos, visualização com filtros e preenchimento automatizado via IA.
3. **Operações de Vinculação e Desvinculação:** Abrange os processos de associação e desassociação de equipamentos com usuários, incluindo termos de responsabilidade e QR Code.
4. **Relatórios e Visualizações Analíticas:** Consolida as funcionalidades de visualização de dados, filtros e dashboards.

5.1.2 Histórias de Usuário

O Backlog do Produto foi detalhado em Histórias de Usuário (User Stories - US), organizadas por funcionalidades prioritárias para a composição do MVP. Cada história segue o formato padrão, descrevendo o papel do usuário, a funcionalidade desejada e o benefício esperado. A Tabela 2 apresenta o mapeamento completo das histórias por épico.

Tabela 2 – Histórias de Usuário por Épico

Épico	**História de Usuário**
1	US01 - Eu, como usuário, desejo realizar login na plataforma com matrícula e senha, a fim de acessar os módulos da aplicação web conforme meu papel associado.
1	US02 - Eu, como usuário com papel associado ao módulo de gerenciamento de usuários, desejo cadastrar e editar usuários com matrícula, email, centro de custo, status e nome, incluindo a definição de papéis associados, a fim de gerenciar o acesso à plataforma conforme permissões de cada módulo.

Épico	**História de Usuário**
1	US03 - Eu, como usuário com papel associado ao módulo de gerenciamento de usuários, desejo visualizar a lista de usuários com filtros por tipo de colaborador, matrícula, centro de custo, perfil e status, a fim de facilitar a administração conforme minhas permissões de acesso.
1	US04 - Eu, como novo usuário, desejo receber uma notificação de boas-vindas com link para troca de senha, a fim de iniciar meu uso da plataforma.
1	US05 - Eu, como usuário, desejo trocar minha senha, a fim de manter minha conta segura.
1	US06 - Eu, como usuário, desejo visualizar meus próprios dados cadastrados, a fim de acompanhar minhas informações na plataforma, tendo equipamentos associados ao meu perfil e meus acessos.
2	US07 - Eu, como usuário com papel de administrador ou operador, desejo cadastrar ativos com data, foto, número de série, nota fiscal, a fim de manter o inventário atualizado.
2	US08 - Eu, como aplicação, desejo utilizar IA para preencher cadastros de ativos a partir da leitura da nota fiscal, a fim de agilizar o processo.
2	US09 - Eu, como usuário com papel de administrador ou operador, desejo visualizar ativos com filtros por tipo de ativo, data de inserção, tipo de colaborador, matrícula, centro de custo e status, a fim de facilitar a gestão.
3	US10 - Eu, como usuário com papel de administrador ou operador, desejo vincular um equipamento a um usuário mediante termo de responsabilidade assinado, a fim de registrar a custódia.
3	US11 - Eu, como usuário com papel de administrador ou operador, desejo desvincular um equipamento de um usuário, registrando status, foto e descrição, a fim de finalizar a custódia.
3	US12 - Eu, como usuário com papel de administrador ou operador, desejo visualizar assinaturas de equipamentos vinculados ou desvinculados com filtros associados, a fim de acompanhar os termos.
3	US13 - Eu, como usuário com papel de administrador ou operador, desejo verificar o histórico de vinculações em um ativo em específico.
3	US14 - Eu, como usuário, desejo escanear um QR Code que redirecione para visualização do ativo, a fim de consultar suas informações.

Fonte: Autoria própria

5.2 Roadmap

Para o planejamento estratégico do projeto, optou-se pela elaboração de um *Roadmap Ágil*, que consiste em uma representação visual de alto nível que comunica a direção do produto ao longo do tempo, alinhando-se aos objetivos estratégicos e às necessidades dos stakeholders. Diferente de cronogramas tradicionais com datas rígidas, o Roadmap Ágil é focado em resultados e mantido como um artefato vivo, sujeito a ajustes contínuos conforme o aprendizado e o feedback são incorporados ao ciclo de desenvolvimento. (MIRO, 2024)

Ele serve como um guia estratégico que conecta a visão do produto ao trabalho executado nas *Sprints* do Scrum. Ele está organizado em temas épicos (grandes iniciativas), que são decompostos em funcionalidades entregáveis, sem se prender a prazos fixos, mas sim a metas de curto, médio e longo prazo. A sua estrutura contempla:

- **Visão do Produto**: Definição clara do objetivo final do produto e do valor que se pretende entregar.
- **Objetivos Estratégicos**: Metas mensuráveis que orientam as decisões de priorização.
- **Épicos**: Grandes blocos de funcionalidades agrupadas por temática.
- **Linhas do Tempo**: Estimativas de alto nível para entrega dos épicos, organizadas em trimestres.

5.2.1 Elaboração do Roadmap

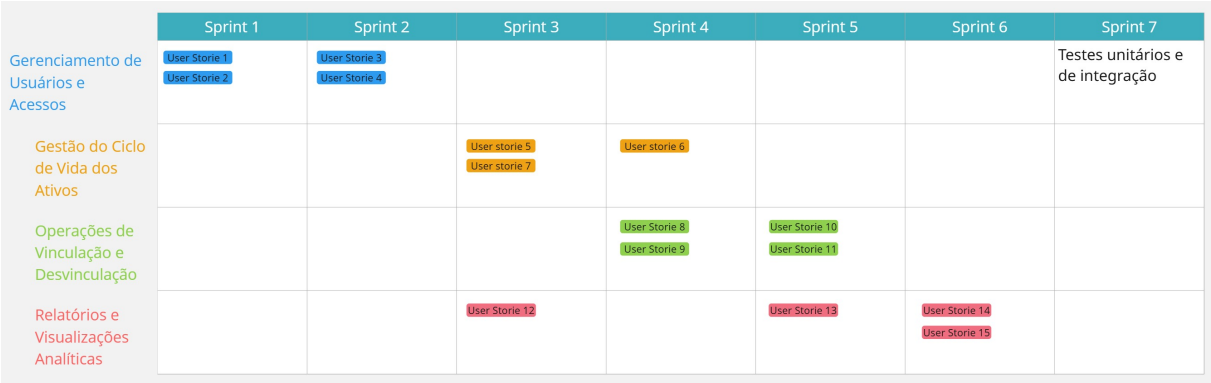
O Roadmap foi elaborado em colaboração com o cliente e demais stakeholders, utilizando a ferramenta Miro para permitir o alinhamento visual e a priorização coletiva. Sua manutenção ocorrerá a cada revisão do *Product Backlog*, incorporando insights das *Sprint Reviews* e eventuais mudanças de escopo ou prioridades.

Cada épico do Roadmap é decomposto em Histórias de Usuário menores e mais granularizadas, que passam a compor o Backlog e são priorizadas para as Sprints. Dessa forma, o Roadmap garante que o trabalho tático das Sprints permaneça alinhado com a estratégia global do produto.

No cenário deste trabalho, o roadmap foi modelado em uma visualização híbrida e prática, utilizando a ferramenta Miro. A estrutura escolhida organiza os Épicos no eixo vertical (lado esquerdo) e as Sprints no eixo horizontal (parte superior), formando uma matriz. No centro, na interseção de cada épico e sprint, são alocadas as Histórias de Usuário (US) previstas para serem trabalhadas naquela ciclo, conforme ilustrado na Figura 28.

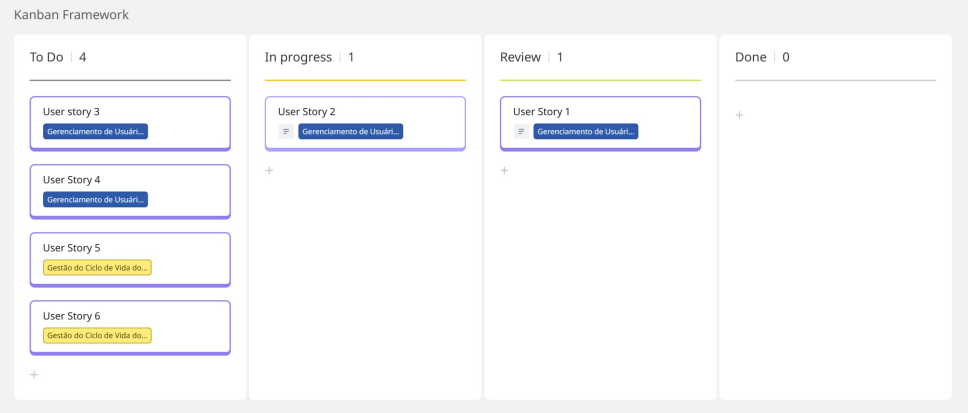
Para complementar esta visão macro e gerenciar a execução, um Quadro Kanban operacional é utilizado em paralelo (Figura 29). Enquanto a matriz de roadmap retrata o que será feito e em qual sprint, o Kanban ilustra o status de cada tarefa. As US da matriz são decompostas em tarefas técnicas no backlog do produto e, uma vez priorizadas e selecionadas para uma sprint, são movidas para o quadro Kanban, que segue o fluxo padrão (*To Do*, *In Progress*, *Review* e *Done*). Esta dupla de artefatos proporciona transparência completa sobre o plano e o progresso, facilitando a identificação de gargalos e o controle do Work In Progress (WIP).

Figura 28 – Matriz de Roadmap.



Fonte: Autoria própria.

Figura 29 – Quadro Kanban para gestão do fluxo de trabalho.



Fonte: Autoria própria.

5.3 Protótipos

O protótipo de alta fidelidade representa a evolução final do design de interface e experiência do usuário, incorporando elementos visuais refinados, interações realistas e fluxos navegacionais completos, conforme definido pela biblioteca de recursos do Figma

(Figma Inc., 2023). Esta versão permite validações precisas de usabilidade, testes de funcionalidades e o refinamento de detalhes críticos antes da etapa de desenvolvimento, assegurando que o produto atenda aos requisitos de negócio e às expectativas dos usuários.

5.3.1 Diretrizes de design e identidade visual

Antes da apresentação das telas, é importante destacar as diretrizes de design adotadas para a aplicação web GuardItens. A identidade visual foi desenvolvida com base nos princípios de usabilidade, contraste e acessibilidade, utilizando uma paleta de cores predominantemente clara com tons de cinza (#ECEFF1, #B4B4B5, #A0A0A0) que proporcionam conforto visual durante o uso prolongado, complementada pelo verde (#38E07B) para indicar ações positivas, status ativos e elementos de destaque. Os tons de branco (#FFFFFF) são utilizados estrategicamente em diferentes opacidades para fundos e elementos de interface, garantindo hierarquia visual e legibilidade adequadas. A tipografia prioriza legibilidade com a fonte Manrope sobre o fundo claro.

A Figura 30 apresenta a logo da aplicação web GuardItens, concebida para transmitir os conceitos de precisão, controle e inteligência na gestão de ativos de tecnologia. O elemento visual remete à organização e rastreabilidade, alinhadas à proposta de valor do produto.

Figura 30 – Logotipo do GuardItens.

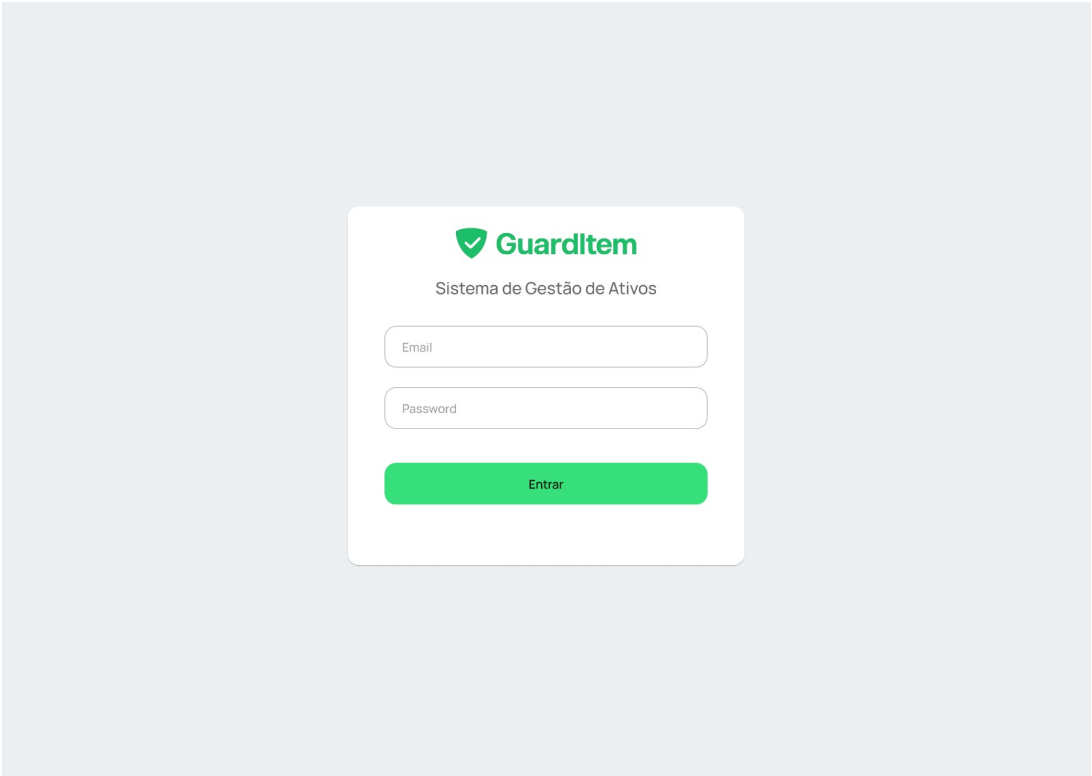


Fonte: Autoria própria.

5.3.2 *Login* e tela inicial

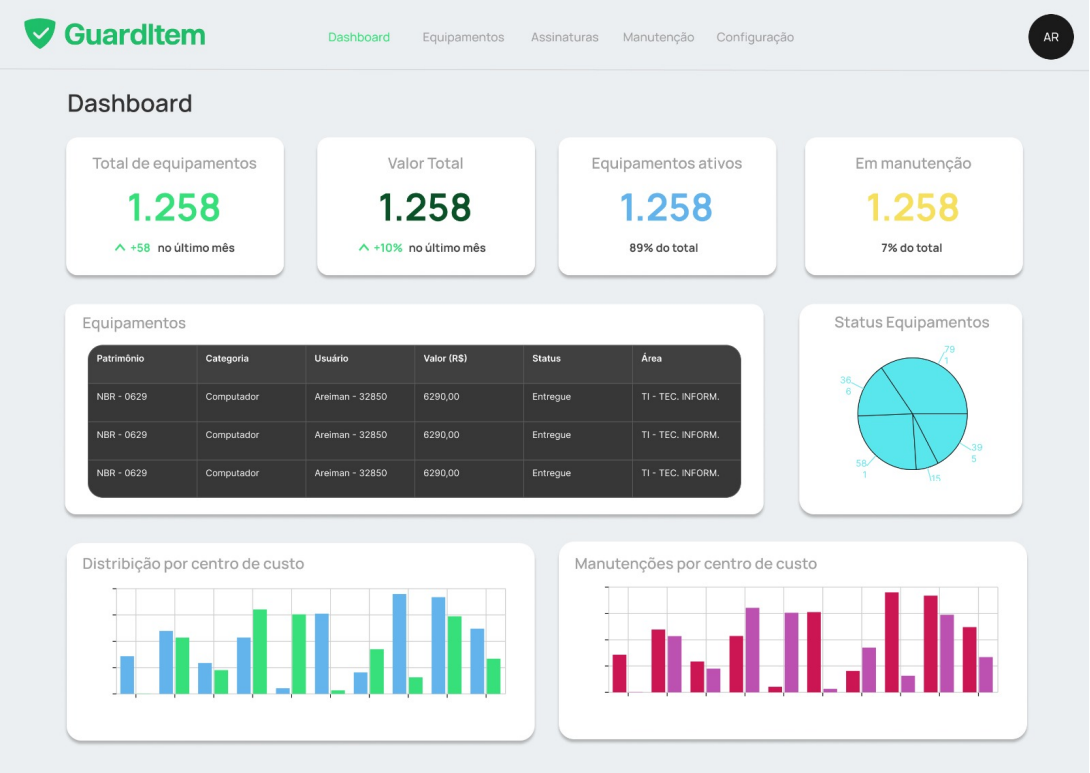
As telas apresentadas na Figura 31 demonstram o fluxo de autenticação do produto, começando pela página de *login* onde os usuários inserem suas credenciais. Já as telas da Figura 32 correspondem ao dashboard principal, que fornece uma visão consolidada dos ativos de TI e métricas relevantes para gestão.

Figura 31 – Protótipo da tela de login.



Fonte: Autoria própria.

Figura 32 – Protótipo da tela de *dashboard*.



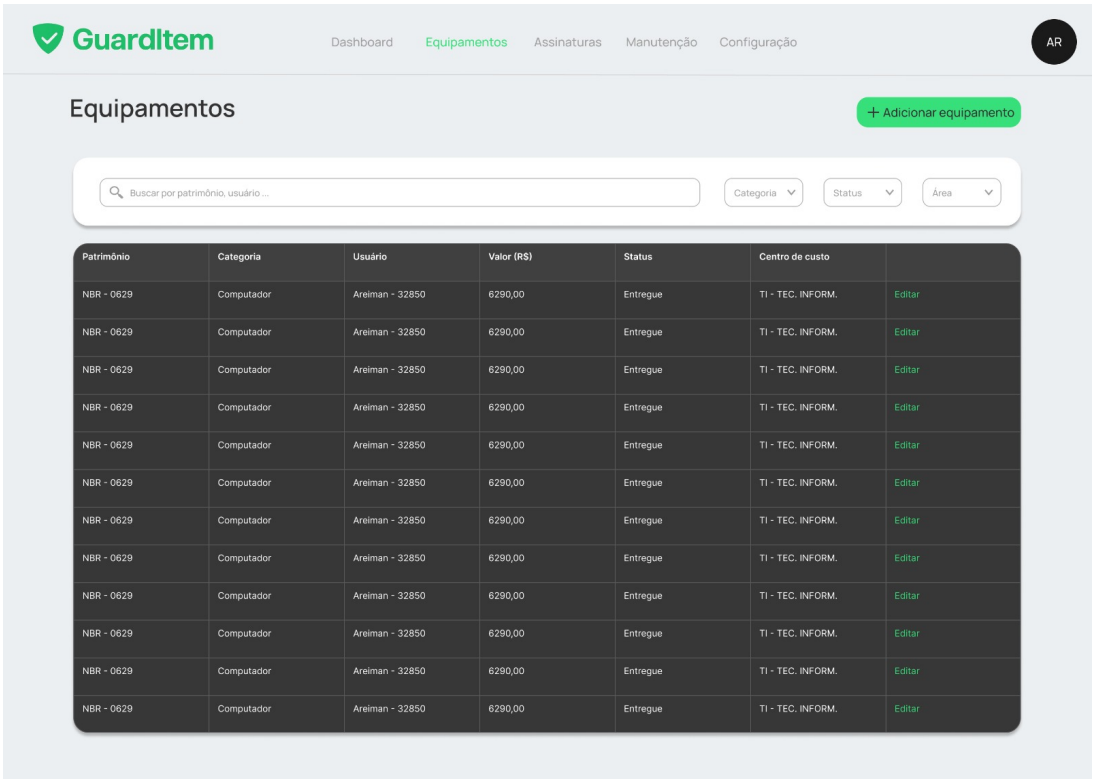
Fonte: Autoria própria.

5.3.3 Gerenciamento de equipamentos

As telas apresentadas na Figura 33 demonstram a interface de gerenciamento de equipamentos do produto GuardItens, que centraliza o controle do inventário de ativos de TI. A tela principal apresenta uma tabela com todos os equipamentos cadastrados, contendo colunas para identificação, categoria, status, localização e data de aquisição, além de opções de filtro e busca para facilitar a localização de itens específicos.

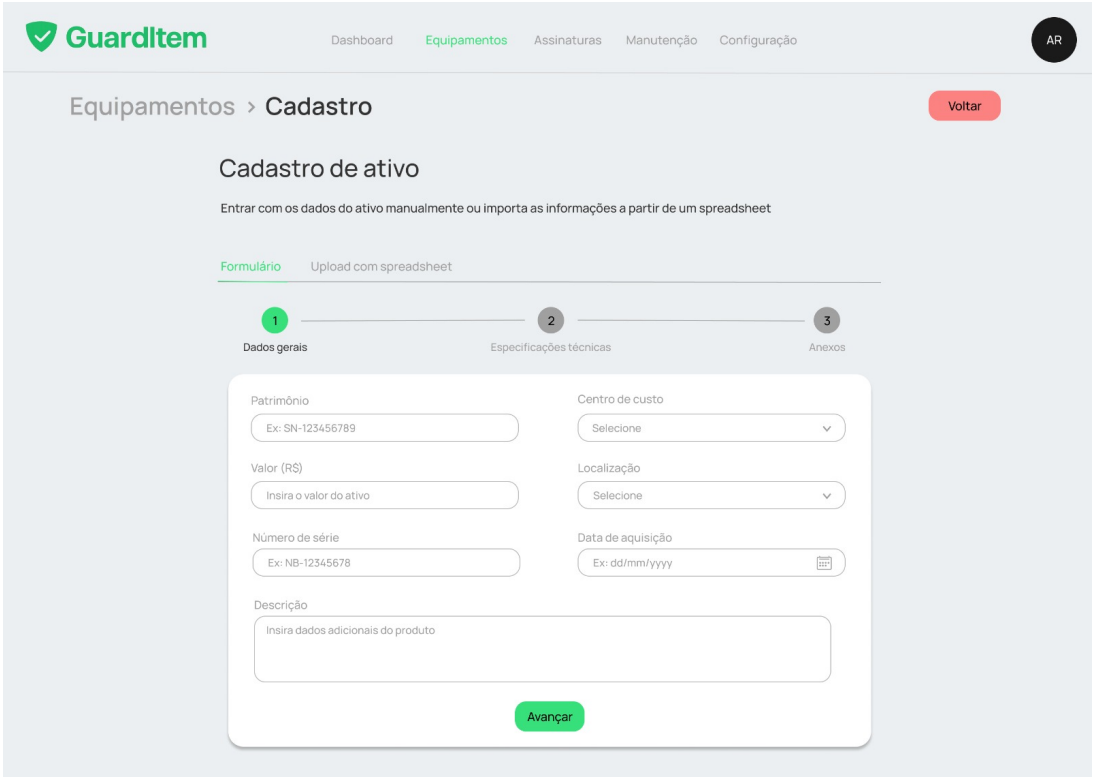
As Figuras 34 (Formulário de cadastro parte 1), 35 (Formulário de cadastro parte 2) e 36 (Formulário de cadastro parte 3) complementam o fluxo ao exibir a tela de cadastro de equipamentos, que oferece duas abordagens distintas: entrada individual através de formulário detalhado com campos como número de série, modelo, fabricante, especificações técnicas e data de aquisição. Permitindo a associação de anexos, com notas fiscais, criando um histórico documental completo para cada ativo.

Figura 33 – Protótipo da tela de equipamentos.



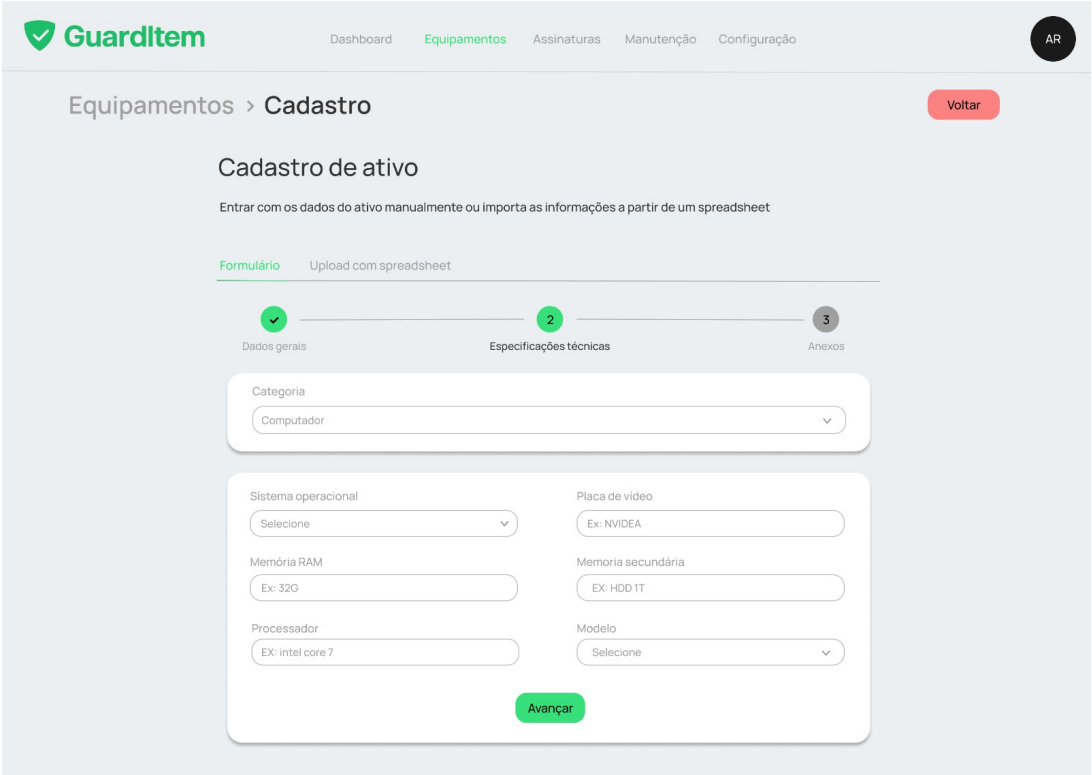
Fonte: Autoria própria.

Figura 34 – Protótipo da tela de cadastro de equipamentos.



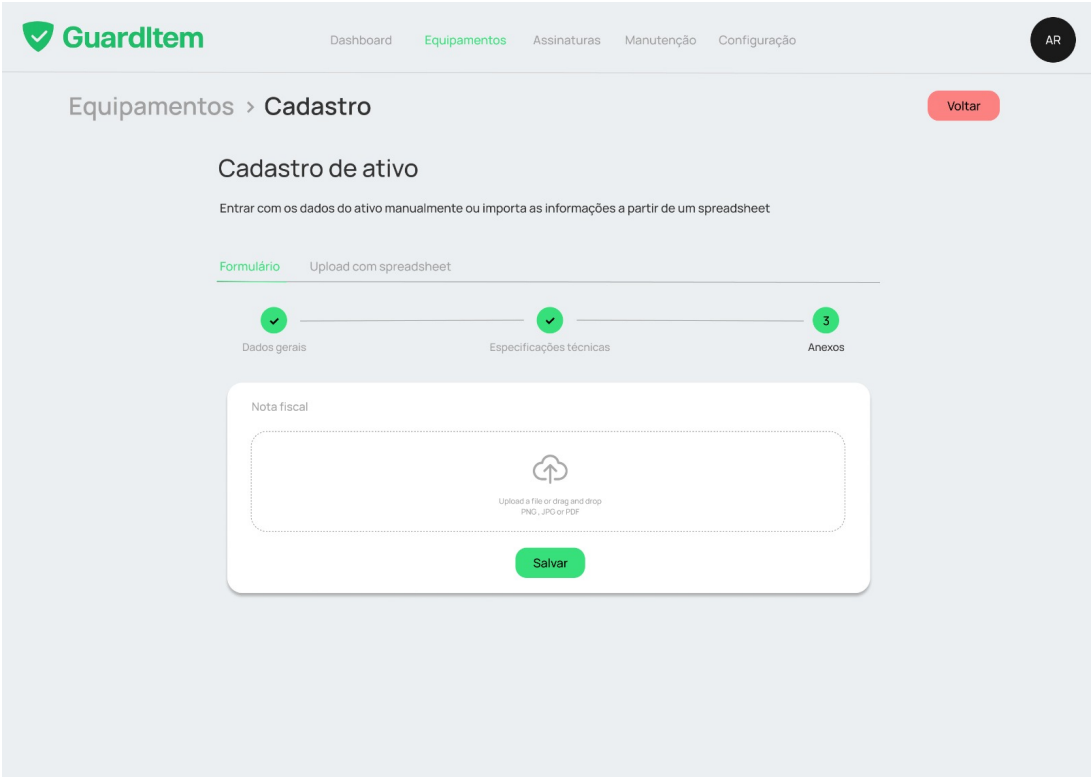
Fonte: Autoria própria.

Figura 35 – Protótipo da tela de cadastro de equipamentos parte 2.



Fonte: Autoria própria.

Figura 36 – Protótipo da tela de cadastro de equipamentos parte 3.



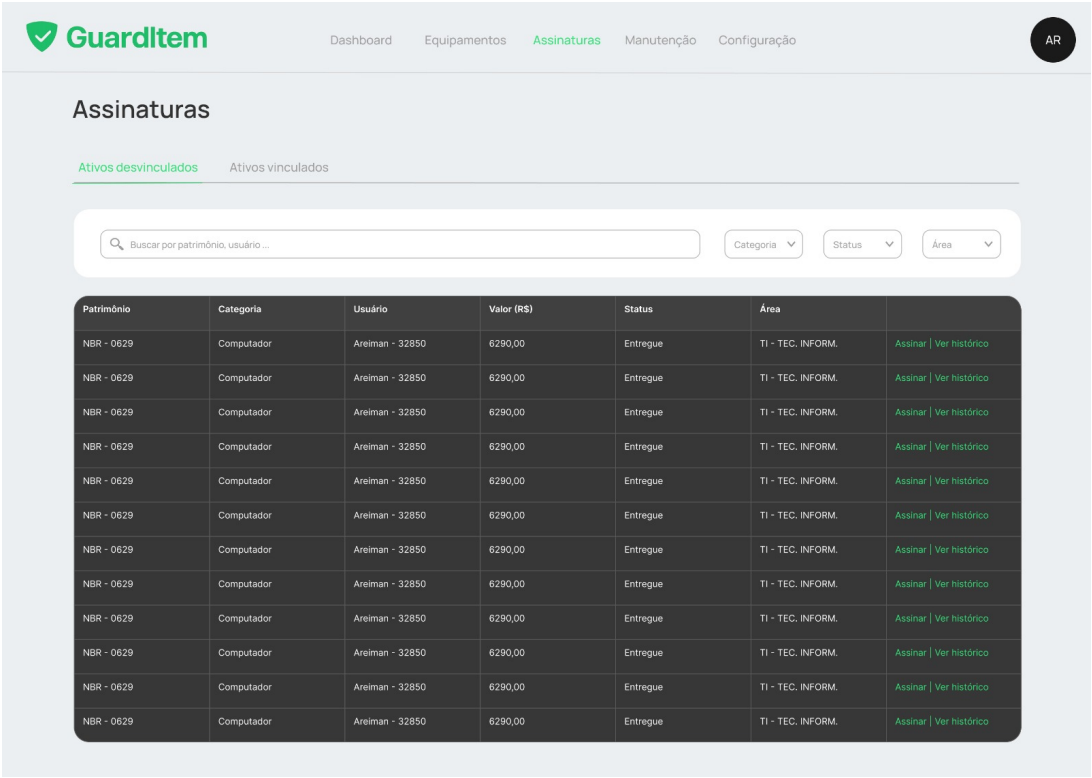
Fonte: Autoria própria.

5.3.4 Gestão de assinaturas

5.3.4.1 Equipamentos desvinculados

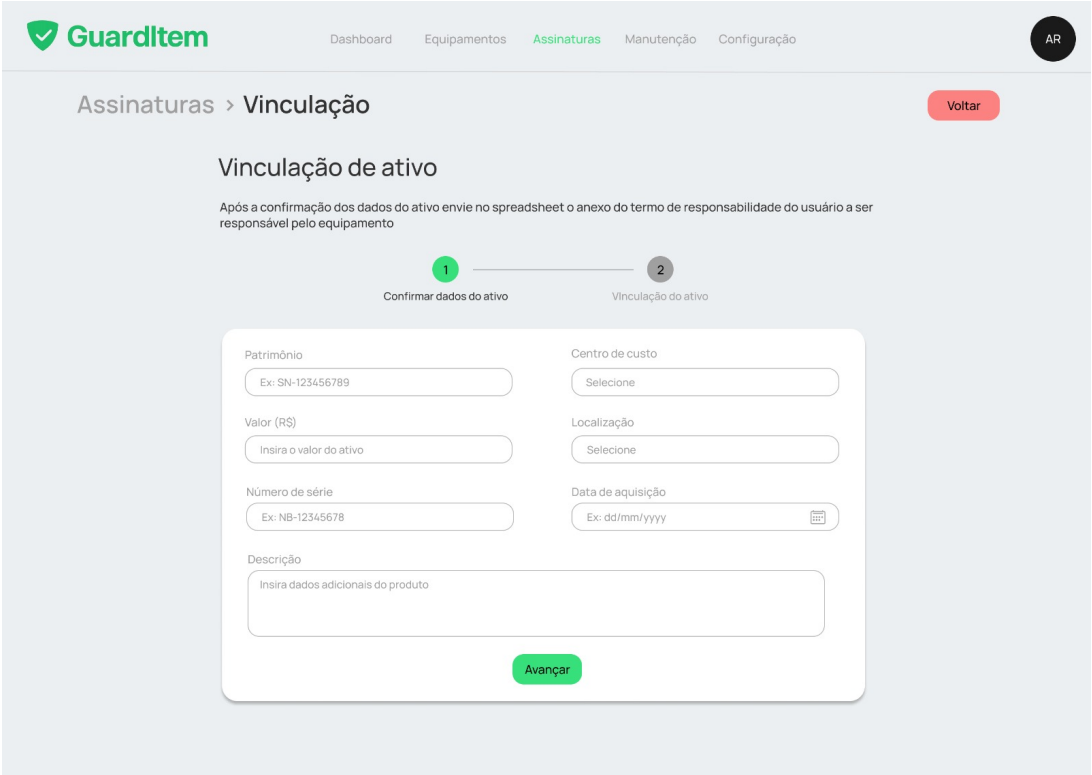
As Figuras 37, 38 e 39 demonstram o fluxo para equipamentos desvinculados. A primeira tela apresenta o inventário de ativos disponíveis para associação, com filtros por categoria, área e status. A segunda tela detalha o processo de vinculação, onde é possível associar o equipamento a um colaborário através do preenchimento de um formulário específico e upload do termo de responsabilidade digitalizado, criando um registro completo com data de assinatura e informações das partes envolvidas.

Figura 37 – Protótipo da tela de assinaturas com ativos desvinculados.



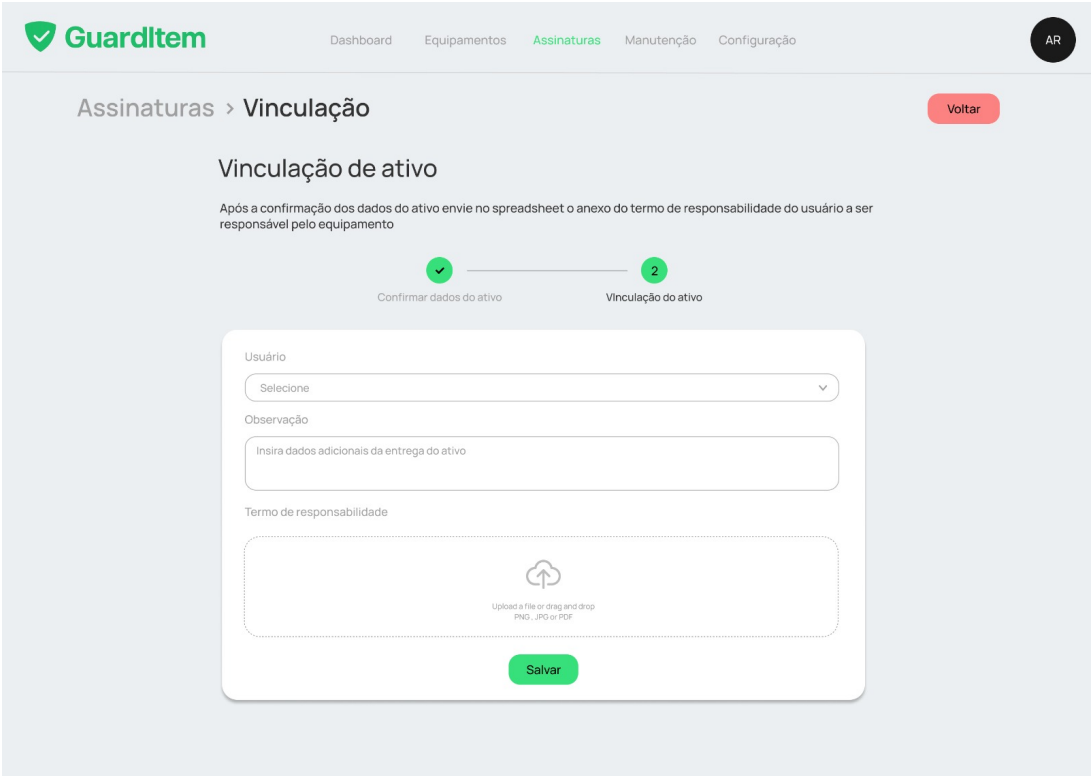
Fonte: Autoria própria.

Figura 38 – Protótipo da tela de cadastro de vinculação parte 1.



Fonte: Autoria própria.

Figura 39 – Protótipo da tela de cadastro de vinculação parte 2.

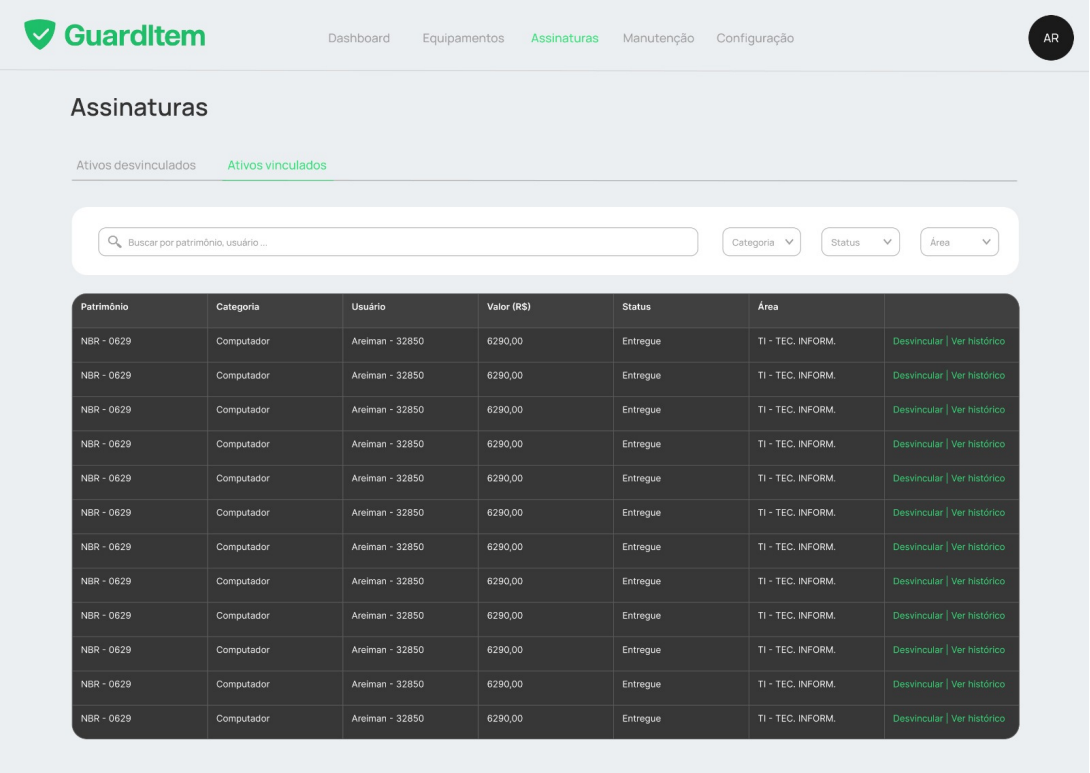


Fonte: Autoria própria.

5.3.4.2 Equipamentos vinculados

A Figura 40 representam a gestão de equipamentos vinculados. A tela principal exibe todos os ativos associados a colaboradores, com informações do responsável atual, data de vinculação e status operacional. A tela de desvinculação, similar a de vinculação, permite remover a associação através do registro de um termo de baixa, que documenta as circunstâncias da devolução e atualiza o status do equipamento no inventário.

Figura 40 – Protótipo da tela de assinaturas com ativos vinculados.

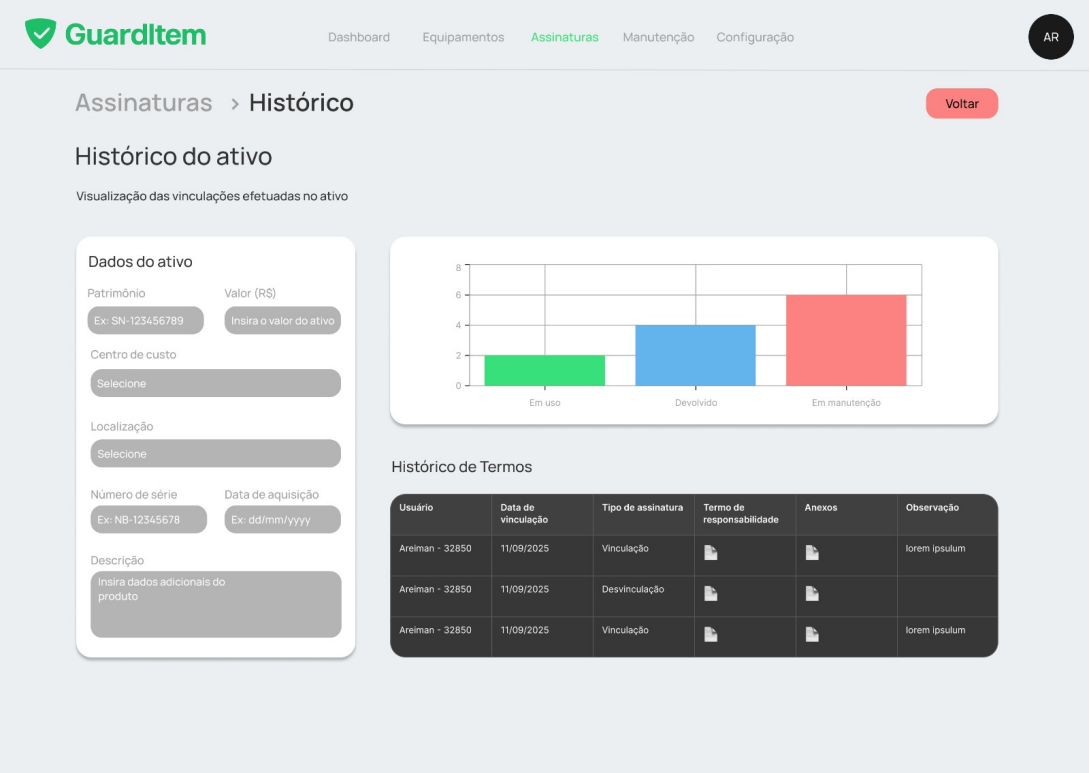


Fonte: Autoria própria.

5.3.5 Histórico e status

A Figura 41 complementa os módulos de equipamentos e assinaturas ao apresentar a tela de histórico completo dos termos associados a cada equipamento além dos status associados por operador. Esta interface permite visualizar toda a linha do tempo de vinculações e desvinculações, incluindo os documentos digitais de cada operação, datas de assinatura, colaboradores envolvidos e status históricos do ativo, proporcionando transparência total na gestão de responsabilidades.

Figura 41 – Protótipo da tela de histórico do ativo.



Fonte: Autoria própria.

6 Tecnologias e Ferramentas de Desenvolvimento

O presente capítulo dedica-se à exposição do conjunto de tecnologias e ferramentas essenciais que fundamentam o desenvolvimento do aplicativo em questão. São elucidados os instrumentos principais adotados para a gestão e o acompanhamento do projeto, bem como as tecnologias de front-end e back-end empregadas, as quais possibilitaram a construção eficaz da solução. Por fim, destaca-se a aplicação específica de cada item selecionado no contexto deste trabalho.

6.1 Ferramentas de Apoio ao Desenvolvimento

As ferramentas descritas nesta seção constituem a base operacional que sustenta a execução do projeto. Elas desempenham funções críticas, abrangendo desde o gerenciamento de tarefas e versionamento de código até a coleta de métricas diretamente dos usuários. Esses dados são vitais para orientar o processo de desenvolvimento, permitindo decisões embasadas e contínuos aprimoramentos no aplicativo.

6.1.1 Github

O GitHub consiste em uma plataforma de hospedagem baseada em nuvem destinada ao versionamento e armazenamento de código-fonte de projetos de software. Conforme documentação oficial (GitHub, 2025), a ferramenta possibilita o armazenamento de repositórios e a divisão do código em diferentes branches, além de permitir a revisão de alterações por meio de Pull Requests, entre outras funcionalidades.

No presente trabalho, o GitHub foi empregado para armazenar e rastrear o código-fonte do aplicativo, assegurando um controle de versão eficiente durante todo o processo de desenvolvimento.

6.1.2 Miro

O Miro consiste em uma plataforma online que oferece diversos modelos de quadros virtuais, focados especialmente em atividades colaborativas. Conforme documentação oficial (Miro, 2025), a ferramenta possibilita a criação e o compartilhamento de quadros, permitindo que múltiplos usuários colaborem em tempo real durante processos de ideação, design ou planejamento de projetos.

No contexto deste trabalho, o Miro foi empregado para facilitar a realização da *Lean Inception* – metodologia estruturada para definição de visão de produto e elicitação de requisitos –, utilizando-se um *template* específico da plataforma desenvolvido para a execução remota dessa técnica.

6.1.3 Figma

O Figma caracteriza-se como uma ferramenta de design colaborativo voltada à prototipagem de interfaces para websites, aplicativos móveis e outros produtos digitais. De acordo com sua documentação (Figma, 2025), a plataforma permite a criação de protótipos interativos de alta fidelidade, além de oferecer funcionalidades avançadas para colaboração síncrona entre designers e desenvolvedores.

Para este projeto, o Figma foi utilizado no desenvolvimento da prototipagem das interfaces do aplicativo, servindo como referência visual fundamental durante toda a fase de implementação das funcionalidades.

6.2 Tecnologias de Desenvolvimento

O desenvolvimento da aplicação web proposta neste trabalho fundamenta-se na adoção de um conjunto de tecnologias consolidadas no mercado, selecionadas conforme sua adequação aos requisitos do projeto, maturidade e suporte comunitário. Esta seção detalha as tecnologias empregadas no front-end e back-end da solução, bem como as plataformas que fornecem a infraestrutura necessária para o desenvolvimento do projeto.

6.2.1 Front-end: React

No desenvolvimento de software, denomina-se front-end a camada de apresentação e interação direta com o usuário final. Para aplicações web, o React consolida-se como uma das principais bibliotecas JavaScript para construção de interfaces modernas e responsivas. Desenvolvida e mantida pelo Facebook, o React introduziu o conceito de componentes reutilizáveis, que revolucionou a forma de desenvolver interfaces web e desktop.

Conforme documentação oficial (React, 2024), esta tecnologia oferece uma arquitetura baseada em componentes que promove a reutilização de código e a separação de concerns. O Virtual DOM, uma das inovações mais significativas do React, permite atualizações eficientes da interface através de um diff algorithm que identifica e aplica apenas as mudanças necessárias no DOM real.

O ecossistema React conta com ferramentas complementares como React Router para gerenciamento de rotas, Context API para gerenciamento de estado global, e diversos hooks que facilitam o gerenciamento de estado e efeitos colaterais em componentes

funcionais. A comunidade ativa e o vasto conjunto de bibliotecas disponíveis tornam o React uma escolha robusta para projetos de grande escala.

Neste trabalho, o React foi fundamental para criar uma interface de usuário intuitiva e responsiva, com componentes modulares que podem ser facilmente testados e mantidos. A capacidade de integrar-se com TypeScript acrescenta type safety ao desenvolvimento front-end, reduzindo erros em tempo de execução.

6.2.1.1 Tecnologias Complementares para o Ecossistema React

Para complementar o desenvolvimento com React e TypeScript, foram integradas tecnologias adicionais que ampliam as capacidades da aplicação e melhoram a experiência de desenvolvimento. A seleção dessas ferramentas foi pautada pela maturidade, suporte da comunidade e adequação aos requisitos do projeto.

O Material-UI (MUI) foi adotado como biblioteca de componentes React que implementa os princípios do Material Design, sistema de design desenvolvido pelo Google. Conforme sua documentação (Material-UI, 2025), esta biblioteca oferece um conjunto abrangente de componentes pré-construídos, responsivos e acessíveis, que aceleram significativamente o processo de desenvolvimento de interfaces. A adoção do Material-UI neste trabalho permitiu a criação de uma interface coesa e profissional, com atenção aos aspectos de usabilidade e experiência do usuário, mantendo a consistência visual em toda a aplicação.

Para as requisições HTTP e integração com APIs RESTful, foi selecionado o Axios, cliente HTTP baseado em Promises. De acordo com a documentação oficial (Axios, 2025), esta biblioteca oferece vantagens significativas em relação ao fetch nativo. A integração do Axios com TypeScript proporcionou type safety completo nas comunicações com a API, desde a definição dos contratos de dados até o tratamento de erros, reduzindo significativamente erros em tempo de execução.

Para o processo de build e desenvolvimento, optou-se pelo Vite como ferramenta de build. Conforme documentação (Vite, 2025), o Vite oferece Hot Module Replacement (HMR) extremamente rápido e build otimizado para produção. A combinação dessas tecnologias com o core React e TypeScript resultou em um ecossistema de desenvolvimento robusto e alinhado com as melhores práticas modernas de desenvolvimento front-end.

6.2.2 Back-end: Node.js

Para a camada server-side da aplicação, optou-se pela utilização do Node.js, ambiente de execução JavaScript construído sobre o motor V8 do Chrome. O Node.js representa uma mudança de paradigma no desenvolvimento back-end ao utilizar um modelo

de I/O não-bloqueante e orientado a eventos, o que o torna particularmente eficiente para aplicações que exigem alta concorrência e operações intensivas de I/O.

Conforme documentação oficial (Node.js, 2024), a arquitetura do Node.js baseia-se em um event loop single-thread que gerencia múltiplas conexões simultaneamente de forma não-bloqueante. Esta característica torna-o ideal para aplicações em tempo real, APIs RESTful e microserviços.

O ecossistema npm (Node Package Manager) oferece o maior repositório de pacotes de código aberto do mundo, com mais de um milhão de pacotes disponíveis. Este vasto ecossistema permite acelerar significativamente o desenvolvimento através da reutilização de soluções bem estabelecidas para problemas comuns.

O Node.js oferece excelente performance para operações de rede e é amplamente utilizado por empresas de grande porte como Netflix, PayPal e Uber. Sua capacidade de usar JavaScript tanto no front-end quanto no back-end permite full-stack development com uma única linguagem de programação.

6.2.2.1 Tecnologias Complementares para o Ecossistema Node.js

Para complementar o desenvolvimento com Node.js e TypeScript no back-end, foram integradas tecnologias adicionais que potencializam a performance, segurança e eficiência da aplicação. A seleção dessas ferramentas considerou critérios de performance, facilidade de uso e adequação aos requisitos da aplicação.

O Fastify foi escolhido como framework web para construção da API RESTful, destacando-se por sua performance superior e baixa sobrecarga computacional. Conforme documentação oficial (Fastify, 2025), este framework oferece suporte nativo a TypeScript, validação de schemas com JSON Schema, injeção de dependências e sistema de hooks extensível. Sua arquitetura modular e ecossistema de plugins permitem uma customização avançada das funcionalidades da API.

Para a camada de persistência de dados, optou-se pelo Prisma como ORM (Object-Relational Mapping). De acordo com a documentação (Prisma, 2025), esta ferramenta oferece uma camada de abstração de banco de dados type-safe, com geração automática de tipos TypeScript baseados no schema da base de dados. O Prisma simplifica operações complexas de banco de dados, oferece queries preventivas contra SQL injection e possui ferramentas de migração e seeding integradas.

A documentação da API foi implementada utilizando Swagger/OpenAPI, seguindo as especificações padrão do industry para APIs RESTful. Conforme documentação (Swagger, 2025), esta solução permite a geração automática de documentação interativa, com exemplos de requests e responses, além de possibilitar testes diretamente através da interface web. A integração com TypeScript garante que a documentação esteja sempre

sincronizada com a implementação real dos endpoints.

Para autenticação e autorização de usuários, foi implementado JWT (JSON Web Tokens) como mecanismo de segurança. Baseado na especificação RFC 7519 (JWT, 2025), este padrão permite a criação de tokens seguros e auto-contidos, que armazenam informações de claims do usuário de forma criptografada. A implementação inclui estratégias de refresh token, validação de assinatura e controle de expiração para garantir a segurança das sessões de usuário.

A combinação dessas tecnologias com o ambiente Node.js e TypeScript resultou em uma arquitetura de back-end robusta, segura e altamente escalável, capaz de atender aos requisitos de performance e confiabilidade exigidos pela aplicação.

6.2.3 Linguagem de Programação: TypeScript

TypeScript, desenvolvido pela Microsoft, é uma linguagem de programação que estende o JavaScript adicionando tipos estáticos e outros recursos avançados. Conforme especificação oficial (TypeScript, 2024), o TypeScript foi projetado para o desenvolvimento de aplicações grandes e complexas, addressing muitas das limitações do JavaScript em projetos enterprise.

O sistema de tipos estático do TypeScript permite detectar erros durante a compilação, antes da execução do código. Esta característica é particularmente valiosa em grandes codebases, onde erros de tipo podem ser difíceis de detectar e debuggar. Os tipos também servem como documentação viva do código, tornando-o mais legível e maintainable.

Recursos avançados como generics, interfaces, decorators e namespaces permitem patterns de programação mais sofisticados e expressivos. O TypeScript suporta os mais recentes features do ECMAScript enquanto oferece backwards compatibility com JavaScript existente.

A integração com editores de código modernos como Visual Studio Code proporciona uma experiência de desenvolvimento excepcional, com autocompletion inteligente, refactoring tools avançados e real-time error checking. A adoção do TypeScript across the stack front-end e back-end garante consistência e qualidade de código em toda a aplicação.

6.2.4 Validação de Dados com Zod

Para garantir a robustez e confiabilidade no processamento de dados, foi implementado o Zod como biblioteca principal para validação de schemas. Conforme documentação oficial (Zod, 2025), esta ferramenta oferece um sistema de validação type-safe com inferência automática de tipos TypeScript, proporcionando uma camada adicional de segurança em tempo de compilação e execução.

O Zod foi integrado tanto no front-end quanto no back-end, assegurando consistência na validação de dados em todas as camadas da aplicação. No lado do servidor, atua em conjunto com o Fastify para validar payloads de requisições HTTP, parâmetros de URL e queries, enquanto no cliente valida dados de formulários e respostas da API antes do processamento.

Entre as principais vantagens observadas na utilização do Zod destacam-se a capacidade de definir schemas complexos com validações customizadas, mensagens de erro descritivas e a geração automática de tipos TypeScript a partir dos schemas definidos. Esta abordagem elimina a duplicação de código entre a definição de tipos e a validação runtime, mantendo uma única fonte de verdade para a estrutura de dados da aplicação.

A implementação do Zod contribuiu significativamente para a redução de bugs relacionados à manipulação de dados, oferecer feedback claro aos usuários sobre erros de entrada e garantir a integridade das informações processadas.

6.2.5 Testes de Software com Vitest

Para garantir a qualidade e confiabilidade do código nas camadas da aplicação, foi implementado o Vitest como framework de testes principal. Conforme documentação oficial (Vitest, 2025), esta ferramenta oferece um ambiente de testing completo e otimizado para projetos modernos com TypeScript e Vite, proporcionando execução rápida de testes.

O Vitest foi configurado para abranger diferentes níveis de teste, incluindo testes unitários, de integração e end-to-end. Sua integração nativa com o ecossistema Vite permite aproveitar a mesma configuração de build do projeto, com suporte a hot module replacement durante o desenvolvimento e geração de reports de cobertura detalhados.

A implementação do Vitest foi efetuada no back-end para garantir a qualidade de código nos endpoints da API, serviços de negócio, validações com Zod e integrações com o Prisma através de mocking de banco de dados.

A adoção desta ferramenta contribuiu significativamente para a manutenibilidade do código, detecção de regressões e confiança nas implementações, seguindo as práticas de Test-Driven Development (TDD) e Behavior-Driven Development (BDD) quando aplicáveis.

6.2.6 Containerização com Docker e Banco de Dados PostgreSQL

Para garantir consistência entre ambientes de desenvolvimento, staging e produção, foi implementada a containerização completa da aplicação utilizando Docker. Conforme documentação oficial (Docker, 2025), esta plataforma permite empacotar aplicações e suas dependências em containers isolados, garantindo portabilidade e reproducibilidade em diferentes infraestruturas.

A arquitetura de containers foi organizada em serviços interconectados, incluindo containers para o serviço de filas com bullmq no redis, serviço de email com mailcathcer e banco de dados PostgreSQL. A utilização do Docker Compose permite orquestrar todos os serviços com definição de networks, volumes e variáveis de ambiente específicas para cada container.

O PostgreSQL foi selecionado como sistema gerenciador de banco de dados relacional principal, conforme documentação (PostgreSQL, 2025), devido à sua robustez, conformidade ACID, suporte a JSONB para dados semi-estruturados e recursos avançados como full-text search. A containerização do PostgreSQL inclui configurações otimizadas para desempenho, políticas de backup automatizado e persistência de dados através de volumes Docker.

A implementação do Dockerfile para o bullmq foi operada via ambiente redis para armazenar os dados temporários de fila. Ele faz uso de alpine como base, com layers separadas para dependências e código da aplicação, aproveitando o cache do Docker para acelerar builds consecutivos.

A infraestrutura como código foi implementada através de docker-compose.yml versionado, permitindo a reprodução exata do ambiente com um único comando.

A integração do Prisma ORM com o PostgreSQL containerizado inclui migrações automatizadas, seeds para dados iniciais e ferramentas de introspection do schema. A conexão entre serviços é gerenciada através de Docker networks, garantindo isolamento e segurança na comunicação entre containers.

7 Arquitetura de software

Para o desenvolvimento da aplicação web de gestão de ativos de TI, optou-se por uma abordagem arquitetural híbrida que combina os princípios da Clean Architecture para o back-end com o padrão *Feature-Based Domain Architecture* para o front-end. Esta escolha estratégica visa garantir a separação de domínios, facilitar a manutenibilidade e permitir a evolução independente das diferentes camadas do projeto.

7.1 Arquitetura do Back-end: *Clean Architecture*

Neste trabalho, será utilizada a *Clean Architecture* como base para a estruturação do projeto Node.js. Essa arquitetura, proposta por Martin (2017), tem como objetivo principal separar as responsabilidades do projeto em diferentes camadas, promovendo baixo acoplamento e alta coesão entre os componentes.

A *Clean Architecture* organiza o produto em camadas concêntricas, onde cada camada tem uma responsabilidade bem definida e depende apenas das camadas mais internas. As principais camadas são:

- *Entities*: Camada mais interna contendo as entidades de negócio e regras de negócio. Inclui modelos de domínio puro e interfaces de repositório, completamente independentes de frameworks externos.
- *Use Cases*: Responsável pela orquestração dos fluxos de negócio. Contém os casos de uso específicos da aplicação, como criação de filas, gestão de usuários e processamento de atendimentos, implementando a lógica de aplicação sem dependências externas.
- *Interface Adapters*: Atua como ponte entre as camadas internas e externas. Inclui controladores HTTP, serializadores, validadores e implementações concretas dos repositórios definidos no domínio.
- *Infrastructure*: Camada mais externa que implementa detalhes técnicos como acesso a banco de dados, sistemas de mensageria, filas e integrações com bibliotecas externas. Esta camada implementa os contratos definidos nas camadas internas.
- *Utils*: Componentes transversais que suportam múltiplas camadas, incluindo sistemas de armazenamento de arquivos, *loggers* específicos por domínio, processamento de *uploads* e utilitários de apoio ao funcionamento da aplicação.

7.1.1 Escolhas de Implementação para Tarefas Assíncronas

Dentro da camada de *Infrastructure*, para o tratamento de operações assíncronas e computacionalmente custosas, em particular o envio de emails transacionais através da biblioteca Nodemailer, optou-se pela implementação de um sistema de filas baseado no BullMQ (2024) com Ltd. (2024) como *message broker* e armazenamento em memória.

Esta decisão arquitetural traz benefícios significativos para o projeto:

- Desacoplamento e Resiliência: O processo principal de resposta às requisições HTTP é desacoplado do envio de emails. Falhas temporárias no serviço de email não impactam a experiência do usuário final, pois as *jobs* são mantidas na fila e reprocessadas automaticamente.
- Escalabilidade: O projeto pode escalar horizontalmente os *workers* que consomem a fila de forma independente da API *web*, atendendo a picos de demanda de forma eficiente.
- Confiabilidade: O BullMQ oferece mecanismos robustos para retentativas de falhas, jobs agendadas e persistência, garantindo que nenhum email seja perdido mesmo em cenários de reinicialização do servidor.
- Monitoramento: A integração com Redis permite visualizar métricas em tempo real e o estado das filas, facilitando a operação e o *troubleshooting*.

Esta solução se alinha perfeitamente com os princípios da *Clean Architecture*, pois o detalhe técnico da implementação das filas está contido na camada mais externa (*Infrastructure*), enquanto as camadas internas de *Use Cases* definem apenas o contrato abstrato de que um email deve ser enviado, sem conhecimento da implementação específica.

7.2 Arquitetura do Front-end: *Feature-Based Domain Architecture*

Para a construção do frontend deste trabalho, será adotada a *Feature-Based Domain Architecture* como abordagem arquitetural principal. Esta metodologia Martin (2012), Vernon (2013) organiza o código-fonte com base nos domínios de negócio e funcionalidades do produto, promovendo alta coesão e baixo acoplamento entre os módulos.

A arquitetura por domínios divide a aplicação em unidades funcionais autônomas, onde cada feature representa um subdomínio específico do negócio. Esta abordagem segue os princípios da *Clean Architecture* e *Domain-Driven Design* (DDD), facilitando a manutenibilidade, testabilidade e evolução do produto. A organização do projeto frontend segue a seguinte estrutura hierárquica:

- Domínios (*Features*): Unidades funcionais que representam subdomínios do negócio, como usuario, equipmento, assinaturas. Cada domínio contém toda a lógica específica para aquela funcionalidade.
- Componentes por Domínio: Componentes de UI específicos de cada domínio, agrupados conforme sua responsabilidade funcional.
- *Hooks* Customizados: Lógica de estado e efeitos reutilizáveis encapsulados em *hooks* específicos por domínio.
- Esquemas de Validação: Definições de estrutura e validação de dados organizadas por domínio.
- Componentes Compartilhados: Componentes de UI genéricos e reutilizáveis across domains.
- Contextos Globais: Gerenciamento de estado compartilhado entre múltiplos domínios.

7.3 Comunicação e Integração

A comunicação entre front-end e back-end será realizada através de APIs RESTful, utilizando JSON como formato de dados padrão. A camada de apresentação consumirá os endpoints implementados no back-end através do cliente HTTP Axios, com tratamento de erros padronizado e gestão de estado.

A separação em repositórios independentes para front-end e back-end permitirá a evolução independente de cada camada, com contratos de API bem definidos através de OpenAPI/Swagger, validações dos dados de entrada com Zod e validação da saída com serializadores que garantem a compatibilidade entre os sistemas.

7.4 Vantagens da Abordagem Proposta

A arquitetura híbrida proposta, que combina *Clean Architecture* no back-end e *Feature-Based Domain Architecture* no front-end, oferece vantagens significativas quando contrastada com abordagens tradicionais como o padrão MVC (*Model-View-Controller*). Enquanto o MVC frequentemente leva ao acoplamento prematuro e à concentração de lógica de negócio nos controladores, como criticado por Martin (2017), a abordagem por domínios adotada neste trabalho promove benefícios arquiteturais substanciais em termos de coesão, acoplamento e manutenibilidade.

A organização por funcionalidades (*features*) proporciona alta coesão e baixo acoplamento, seguindo o Princípio da Responsabilidade Única proposto por Martin (2017).

Cada domínio encapsula integralmente sua própria lógica, componentes e estados, reduzindo significativamente as dependências inadvertidas entre módulos não relacionados. Esta característica é particularmente valiosa em sistemas de média e grande complexidade, onde a manutenibilidade torna-se um fator crítico para a sustentabilidade do projeto.

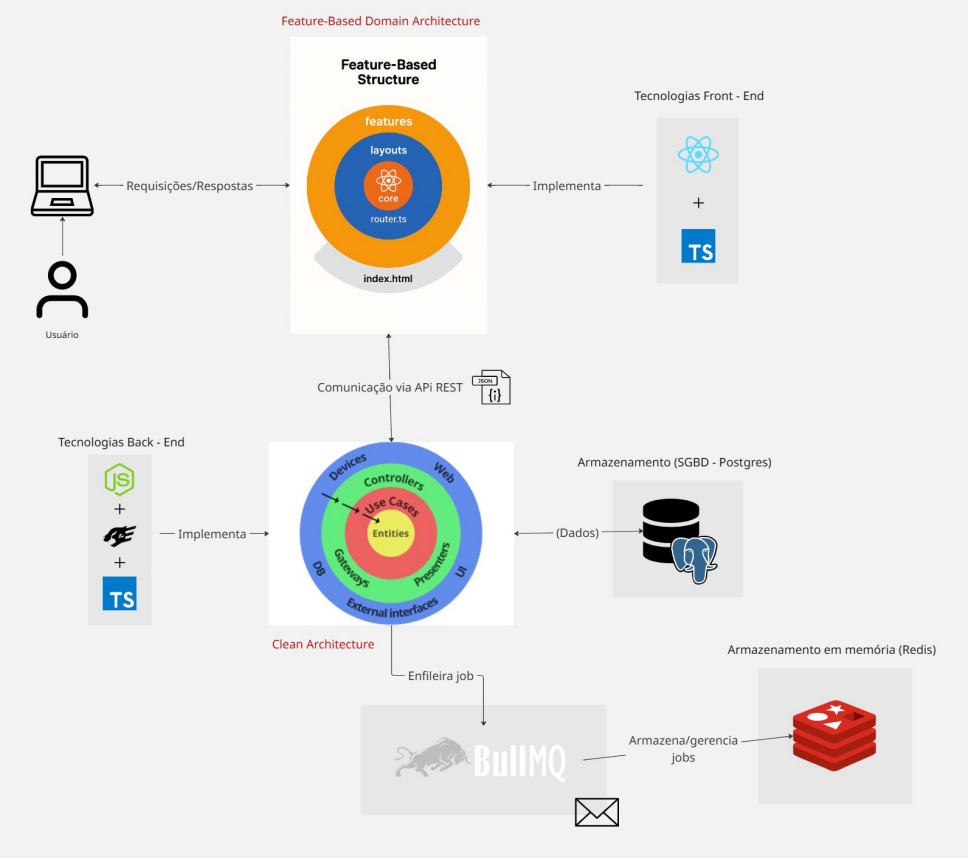
No que concerne à manutenibilidade, a arquitetura baseada em domínios facilita substancialmente a localização e modificação de funcionalidades específicas sem impactar outras áreas do sistema, conforme destacado por Vernon (2013). Esta qualidade contrasta favoravelmente com a abordagem MVC, onde mudanças em modelos frequentemente desencadeiam efeitos em cascata que exigem ajustes em múltiplas camadas. A evolução independente dos domínios, permitindo que diferentes partes do *software* avancem em ritmos distintos, constitui outra vantagem significativa apontada por Evans (2003) em sua abordagem de *Domain-Driven Design*.

No back-end, a *Clean Architecture* complementa estas vantagens através do isolamento rigoroso das regras de negócio em camadas concêntricas, protegendo o domínio central de mudanças em frameworks e bibliotecas externas, conforme os princípios estabelecidos por Martin (2017). Simultaneamente, no front-end, a componentização baseada em domínios maximiza a reutilização dentro de cada contexto funcional específico. Esta estrutura combinada garante que o *software* seja robusto, escalável e capaz de evoluir para atender demandas futuras, mantendo alta qualidade de código e reduzindo significativamente os custos de manutenção quando comparado com abordagens MVC tradicionais.

7.5 Diagrama de arquitetura

A Figura 42 ilustra a arquitetura híbrida proposta para a aplicação web, integrando a *Clean Architecture* no back-end com a *Feature-Based Domain Architecture* no front-end. O diagrama demonstra visualmente as camadas concêntricas do back-end, onde as dependências fluem em direção ao núcleo de domínio, e a organização modular do front-end em *features* independentes. Esta representação evidencia a separação de responsabilidades, o fluxo de dados entre as camadas e a comunicação via APIs RESTful, destacando como as decisões arquiteturais se articulam para formar um sistema coeso e escalável.

Figura 42 – Arquitetura de software.



Fonte: Fonte: Autor

8 Containerização e Orquestração de Serviços

A containerização desempenhou papel fundamental na arquitetura da aplicação web GuardItens, permitindo o empacotamento consistente de todas as dependências e serviços necessários para o funcionamento da aplicação. A abordagem baseada em containers proporcionou isolamento entre serviços, facilidade de *deploy* e consistência entre ambientes de desenvolvimento e produção. O projeto foi estruturado em containers separados para *backend* e *frontend*, cada um com sua própria configuração de Docker Compose, refletindo a separação física dos diretórios do projeto.

8.1 Arquitetura de Containerização do *Backend*

O *backend* do GuardItens adota uma arquitetura multi-container onde cada serviço de infraestrutura é executado em containers isolados, orquestrados através do Docker Compose. Esta abordagem permite que cada componente, seja banco de dados, cache Redis e serviço de email, seja executado de forma independente, comunicando-se através de redes virtuais internas. A separação em containers distintos garante que cada serviço possa ser escalado, atualizado e mantido independentemente.

8.1.1 Docker Compose do *Backend*

O arquivo `docker-compose.yml` define a orquestração de todos os serviços de infraestrutura necessários para a execução do backend. Esta configuração estabelece três serviços essenciais: PostgreSQL para persistência de dados, Redis para cache e filas do bullmq, e MailCatcher para interceptação de emails em ambiente de desenvolvimento.

Listing 8.1 – Arquivo Docker Compose - Backend

```
services:
  db:
    container_name: GuardItens-ts-postgres
    image: postgres
    restart: unless-stopped
    ports:
      - 5432:5432
    environment:
```

```
    POSTGRES_PASSWORD: postgres

redis:
  container_name: GuardItens-redis
  image: redis:alpine
  restart: unless-stopped
  ports:
    - 6379:6379
  command: redis-server --appendonly yes
  volumes:
    - redis-data:/data

mailcatcher:
  container_name: GuardItens-mailcatcher
  image: schickling/mailcatcher
  restart: unless-stopped
  ports:
    - 1080:1080  # Web interface
    - 1025:1025  # SMTP server

app:
  container_name: GuardItens-node-app
  image: node:alpine
  working_dir: /app
  volumes:
    - .:/app
    - /app/node_modules
  ports:
    - 3000:3000
  command: sh -c "npm install && npm run dev"
  environment:
    - NODE_ENV=development
    - DATABASE_HOST=db
    - REDIS_HOST=redis
  depends_on:
    - db
    - redis

worker:
```

```
    container_name: GuardItens-worker
    image: node:alpine
    working_dir: /app
    volumes:
      - .:/app
      - /app/node_modules
    command: sh -c "npm install && npm run dev:worker"
    environment:
      - NODE_ENV=development
      - DATABASE_HOST=db
      - REDIS_HOST=redis
    depends_on:
      - db
      - redis

volumes:
  redis-data:
```

8.1.2 Configuração dos Serviços de *Backend*

O serviço de banco de dados PostgreSQL é configurado utilizando a imagem oficial mais recente do PostgreSQL, com a porta 5432 exposta para acesso externo. As variáveis de ambiente definem a senha de acesso e o nome do banco de dados, enquanto um volume nomeado garante a persistência dos dados mesmo após a reinicialização do container. A política de reinício `unless-stopped` assegura alta disponibilidade do serviço de banco de dados.

O serviço de cache Redis utiliza a imagem Alpine, conhecida por sua segurança e tamanho reduzido. Configurado com a opção `appendonly yes`, o Redis mantém a durabilidade dos dados em cache, com persistência garantida através de volume específico. A porta 6379 é exposta para permitir conexões da aplicação backend com o sistema de cache e filas gerenciada pelo bullmq.

Para ambiente de desenvolvimento, o serviço MailCatcher fornece funcionalidades essenciais de interceptação e visualização de emails. Através da interface web acessível na porta 1080, os desenvolvedores podem visualizar todos os emails enviados pela aplicação durante os testes. O servidor SMTP *fake* na porta 1025 permite que a aplicação envie emails sem a necessidade de configurações complexas de servidor de email real.

8.2 Arquitetura de Containerização do *Frontend*

O frontend do GuardItens segue igualmente uma abordagem baseada em *containers*, garantindo consistência entre ambientes de desenvolvimento e produção. A aplicação React é containerizada de forma independente, refletindo a separação arquitetural entre *frontend* e *backend*. Esta separação permite que as duas aplicações evoluam de forma independente, com ciclos de desenvolvimento e entregas distintas.

8.2.1 Docker Compose do *Frontend*

O arquivo `docker-compose.yml` do *frontend* define a configuração específica para a aplicação React, focando nas necessidades de desenvolvimento e *build* da interface do usuário. Diferente do *backend*, o *frontend* não requer serviços de infraestrutura adicionais, dependendo apenas da API *backend* para seu funcionamento completo.

Listing 8.2 – Arquivo Docker Compose - Frontend

```
version: '3.8'

services:
  frontend:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "5173:5173" # Map Vite's default port
    volumes:
      - .:/app # Mount the project directory
      - /app/node_modules # Prevent overwriting node_modules
    environment:
      - NODE_ENV=development
```

8.2.2 Dockerfile do *Frontend*

O arquivo `Dockerfile` do *frontend* define a configuração específica para a aplicação React, focando nas necessidades de desenvolvimento e *build* da interface do usuário. Diferente do *backend*, o *frontend* não requer serviços de infraestrutura adicionais, dependendo apenas da API *backend* para seu funcionamento completo.

A estrutura do Dockerfile foi cuidadosamente elaborada para otimizar tanto o ambiente de desenvolvimento quanto o de produção. Utilizando a imagem base `node:18-alpine`, garante-se compatibilidade com todas as dependências do React enquanto mantém um

tamanho reduzido da imagem final. A escolha da versão 18 do Node.js assegura suporte às funcionalidades mais recentes do JavaScript e estabilidade de longo prazo, sendo uma versão LTS (Long Term Support).

O processo de construção da imagem segue um fluxo multi-estágio que maximiza a eficiência. Inicialmente, o diretório de trabalho é definido como `/app` e os arquivos de dependências (`package.json` e `package-lock.json`) são copiados para o container.

Listing 8.3 – Dockerfile do Frontend

```
FROM node:18-alpine

WORKDIR /app

COPY package.json ./

RUN npm install

COPY . .

EXPOSE 5173

CMD ["npm", "run", "dev"]
```

Esta abordagem de containerização para o frontend proporciona significativos ganhos de produtividade no desenvolvimento, uma vez que todos os desenvolvedores trabalham em ambientes idênticos, eliminando problemas de inconsistência entre diferentes setups locais. Adicionalmente, a containerização facilita o processo de deploy, uma vez que a mesma imagem testada em desenvolvimento pode ser utilizada em produção, garantindo comportamento consistente em todos os ambientes.

8.2.3 Configuração do Serviço de *Frontend*

O container do *frontend* é construído a partir de um Dockerfile específico localizado no diretório da aplicação React. A configuração expõe a porta 3000 para acesso à interface web e define a variável de ambiente `REACT_APP_API_URL` para configurar a URL da API *backend*. A utilização de volumes permite o desenvolvimento com hot-reload, onde alterações no código fonte são refletidas imediatamente na aplicação em execução, agilizando o processo de desenvolvimento.

A imagem do *frontend* é baseada no Node.js 18 Alpine, garantindo compatibilidade com as dependências do React e otimização de tamanho. Em ambiente de desen-

volvimento, o container executa o servidor de desenvolvimento do React, enquanto em produção serviria os arquivos estáticos buildados.

8.3 Vantagens da Abordagem Adotada

A containerização do GuardItens trouxe benefícios significativos para o desenvolvimento e operação do projeto. A consistência entre ambientes é garantida através do uso das mesmas imagens e configurações em desenvolvimento, teste e produção, eliminando problemas relacionados a diferenças de ambiente. O isolamento de serviços assegura que cada componente execute em ambiente controlado, prevenindo conflitos de dependências e permitindo que cada serviço utilize versões específicas de suas dependências.

A facilidade de setup é outra vantagem marcante, onde novos desenvolvedores podem levantar todo o ambiente com comandos simplificados, sem a necessidade de instalação e configuração manual de cada serviço. A portabilidade do permite que a aplicação seja executada em qualquer *host* com Docker instalado, seja em máquina local, servidor ou ambiente de nuvem. A arquitetura adotada está preparada para escalação horizontal de serviços, podendo facilmente adicionar mais instâncias de qualquer container conforme a demanda.

8.4 Considerações sobre *GitHub Actions*

Embora o *software* esteja preparado para integração contínua, a implementação do *GitHub Actions* não foi contemplada no escopo deste trabalho, ficando como melhoria futura. A estrutura de containerização adotada, no entanto, facilita significativamente a implementação de *pipelines* CI/CD, uma vez que todos os serviços estão adequadamente containerizados e parametrizados. A separação em *Docker Compose* files distintos para *backend* e *frontend* permitiria a construção e teste independente de cada camada do projeto, otimizando os processos de integração e *deployment* contínuos.

9 Demonstração das Funcionalidades Implementadas

Este capítulo apresenta as principais funcionalidades desenvolvidas na aplicação web GuardItem, demonstrando visualmente a interface do usuário e os fluxos de trabalho implementados.

Em vez de imagens estáticas, a demonstração completa do sistema, com todas as suas telas e fluxos, pode ser conferida no vídeo disponível no YouTube através do link: <https://youtu.be/QSanM9pDMMQ>. O vídeo percorre a jornada completa do usuário, desde o acesso ao *software* até as operações complexas de gestão de ativos de TI. É importante observar que, no vídeo, a logo e o nome do produto ainda aparecem como "AssetWise", uma vez que a alteração para "GuardItens"ocorreu após a gravação.

9.1 Sistema de Autenticação e Acesso

9.1.1 Tela de Login

A tela de login constitui o ponto de entrada para o GuardItem, oferecendo uma interface intuitiva e segura para autenticação de usuários. A implementação inclui validações em tempo real para credenciais, tratamento de erros personalizado e medidas de segurança contra tentativas de acesso não autorizadas.

A interface apresenta campos para email e senha. O design minimalista foca na usabilidade, garantindo que usuários de diferentes níveis técnicos possam acessar sem dificuldades. Após a autenticação bem sucedida, o usuário é redirecionado para o dashboard principal da aplicação.

9.1.1.1 Tela de Login - Campos Inválidos

A interface de login foi desenvolvida para fornecer feedback claro e imediato ao usuário em casos de credenciais inválidas, seguindo princípios de usabilidade e experiência do usuário. Quando o usuário insere credenciais incorretas, a aplicação apresenta mensagens de erro contextualizadas que guiam a correção.

Esta abordagem garante que os usuários recebam orientação adequada para corrigir credenciais inválidas, reduzindo a frustração e melhorando a eficiência do processo de autenticação.

9.1.2 Fluxo de Primeiro Acesso

O GuardItem implementa um fluxo seguro para primeiro acesso de usuários, onde cada novo colaborador recebe uma senha temporária via e-mail. Este mecanismo garante que apenas pessoas autorizadas possam acessar o *software*, seguindo práticas recomendadas de segurança da informação.

Ao receber o e-mail de boas-vindas, o usuário utiliza a senha temporária para fazer seu primeiro login. Imediatamente após a autenticação bem-sucedida, a aplicação redireciona automaticamente para a tela de redefinição de senha, obrigando o usuário a definir uma nova senha pessoal.

9.1.3 Tela de Redefinição de Senha Obrigatória

A tela de redefinição de senha no primeiro acesso apresenta requisitos robustos de segurança, exigindo que a nova senha atenda a critérios específicos como comprimento mínimo, uso de caracteres especiais, números e letras maiúsculas. O *software* inclui validação em tempo real para orientar o usuário durante a criação da nova senha.

Após a definição da nova senha, o usuário é autenticado automaticamente com as novas credenciais e recebe uma confirmação visual do sucesso da operação. A senha temporária é invalidada imediatamente, garantindo que não possa ser reutilizada.

9.1.4 Troca de Senha

Para situações onde usuários gostariam de trocar suas credenciais, o *software* oferece um fluxo seguro de troca de senha. Através da opção "trocar senha"na tela de perfil do usuário, ele pode efetuar a redefinição.

9.1.5 Perfil do Usuário e Gestão de Permissões

Cada usuário possui um perfil personalizável onde pode visualizar suas informações básicas, como nome, e-mail, departamento centro de custo. Além de dados associados ao produto como os equipamentos vinculados e perfis de permissão. A tela de perfil também permite ao usuário alterar sua senha a qualquer momento, mediante confirmação da senha atual.

9.1.6 Segurança e Auditoria

Todas as operações de autenticação são registradas em logs de auditoria, incluindo criações de sessão de login sucedidas, redefinições de senha e alterações de perfil. Estes registros são essenciais para compliance e investigação de incidentes de segurança. Tendo

como trabalho futuro o módulo de central de logs que permita a visualização completa desses dados para auditória completa.

O *software* também implementa proteções contra ataques comuns, como rate limiting para prevenção de força bruta, sanitização de entradas do usuário. As senhas são armazenadas utilizando algoritmos de hash seguros, garantindo que mesmo em caso de violação de dados, as credenciais permaneçam protegidas.

9.2 Dashboard e Visão Geral

9.2.1 Dashboard Principal

O dashboard principal oferece uma visão consolidada de todos os ativos de TI gerenciados pela aplicação web, apresentando métricas-chave e indicadores em tempo real. A interface foi projetada para proporcionar uma experiência informativa e eficiente, permitindo que gestores tomem decisões baseadas em dados atualizados.

O dashboard inclui uma tabela rankeada com os 10 ativos de maior valor, ordenados do mais caro para o mais barato, permitindo rápida identificação dos investimentos mais significativos. Complementando esta visão, um gráfico de pizza exibe a distribuição dos equipamentos por status, proporcionando uma visão geral instantânea do estado do parque tecnológico. Adicionalmente, dois gráficos associativos correlacionam equipamentos com centro de custo, destacando a área com maior gasto financeiro e a que possui o maior número de equipamentos.

9.3 Gestão de Ativos de TI

9.3.1 Cadastro de Ativos

A funcionalidade de cadastro de ativos permite o registro completo de equipamentos de TI, incluindo informações técnicas, especificações e dados de aquisição. O formulário inteligente adapta-se ao tipo de equipamento selecionado, solicitando informações específicas relevantes para cada categoria de ativo.

Campos como número de série, modelo, fabricante, data de aquisição e valor são incluídos no cadastro básico. O *software* gera automaticamente um código de identificação único para cada ativo, facilitando o rastreamento e inventário. Validações impedem a duplicação de equipamentos com mesmo número de série.

9.3.2 Histórico de Vinculações e Rastreamento

Cada ativo mantém um histórico completo de todas as vinculações e alterações realizadas ao longo de seu ciclo de vida. Esta funcionalidade permite rastrear toda a jornada do equipamento dentro da organização, desde sua aquisição até eventual baixa ou transferência.

O histórico registra as alterações significativas de vinculações e desvinculações de assinaturas. Cada entrada no histórico contém data e hora exatas e detalhes da modificação realizada.

9.3.3 Identificação por QR Code

Para facilitar a identificação física dos equipamentos, o *software* gera automaticamente um QR Code único para cada ativo cadastrado. Este código pode ser impresso e fixado no equipamento, permitindo acesso rápido às informações do ativo através de leitura com smartphone ou leitor dedicado. O QR Code contém informações específicas do equipamento, incluindo código de identificação e nome do ativo.

O *software* oferece opções de impressão e download de etiquetas, fornecendo os dados de ativo ao escanear o QR Code gerado.

9.3.4 Consulta e Pesquisa de Ativos

A interface de consulta oferece recursos avançados de pesquisa e filtragem, permitindo localizar rapidamente qualquer ativo do patrimônio. Os usuários podem filtrar por categoria, localização, status, responsável ou qualquer campo de metadados cadastrado.

Os resultados são apresentados em formato de tabela interativa, com opções de ordenação por diferentes critérios. Ações rápidas permitem editar, visualizar detalhes.

9.4 Gestão de Assinaturas de Equipamentos

9.4.1 Vinculação de Assinaturas a Ativos

Uma das funcionalidades centrais do *software* é a capacidade de vincular assinaturas de software e serviços aos equipamentos de TI. Esta feature permite o controle completo do ciclo de vida das licenças, desde a aquisição até a renovação ou cancelamento.

A interface permite associar múltiplas assinaturas a um mesmo equipamento, definindo datas de vigência e termos contratuais. O histórico de vinculações fica permanentemente registrado para cada assinatura, permitindo auditoria completa das mudanças ao longo do tempo.

9.5 Módulo de Adiministração

O módulo de administração do GuardItem fornece ferramentas completas para gestão de atividades corretivas e preventivas, apoiado por uma estrutura organizacional robusta que inclui gestão de usuários, centros de custo e perfis de acesso. Esta integração permite um controle preciso dos recursos humanos envolvidos na manutenção do patrimônio de TI.

9.5.1 CRUD de Usuários

A gestão de usuários permite o cadastro e administração completa de todos os colaboradores com acesso ao *software*. O CRUD (Create, Read, Update, Delete) de usuários oferece interface intuitiva para adicionar novos usuários, editar informações existentes, desativar contas e gerenciar permissões de acesso.

Cada usuário é cadastrado com informações completas incluindo nome, e-mail, centro de custo e papel associado. O *software* permite a busca e filtragem, facilitando a localização de usuários específicos em organizações de grande porte. A desativação de usuários preserva o histórico de ações realizadas enquanto impede novos acessos ao *software*.

9.5.2 CRUD de Centros de Custo

O módulo de centros de custo permite a estruturação financeira da organização, vinculando recursos e despesas às respectivas áreas orçamentárias. Cada centro de custo é identificado por código único, descrição, responsável e informações de alocação orçamentária.

A vinculação entre usuários e centros de custo permite o controle preciso de aprovações e responsabilidades financeiras. Esta integração é fundamental para workflows de aquisição e manutenção, onde cada solicitação é automaticamente associada ao centro de custo do usuário requisitante, agilizando processos de aprovação e contabilização.

9.5.3 CRUD de Papéis de Acesso

O *software* implementa um modelo granular de papéis de acesso, permitindo a definição precisa de permissões para diferentes funções organizacionais. Cada papel pode ser customizado com permissões específicas para módulos e funcionalidades.

A interface de gestão de perfis permite visualizar e modificar permissões de forma intuitiva, com agrupamentos lógicos por módulos do *software*.

9.6 Considerações Finais da Implementação

As funcionalidades demonstradas neste capítulo representam o core do produto GuardItem, atendendo aos requisitos essenciais para gestão moderna de ativos de TI. A interface foi desenvolvida seguindo princípios de User Experience (UX) e User Interface (UI), resultando em uma aplicação intuitiva que reduz a curva de aprendizado e aumenta a produtividade dos usuários.

A arquitetura de componentes React permitiu a criação de interfaces reutilizáveis e consistentes, enquanto a integração com o backend Node.js garantiu performance e confiabilidade nas operações. O *software* como um todo demonstra maturidade técnica e adequação aos objetivos estabelecidos para o projeto.

10 Conclusão

O desenvolvimento da aplicação web GuardItens representou uma jornada significativa de transformação de conceitos arquiteturais em uma solução prática e funcional para gestão de ativos de TI. Ao longo deste trabalho, demonstrou-se como a arquitetura híbrida, combinando *Clean Architecture* no *back-end* e *Feature-Based Domain Architecture* no *front-end*, que provou ser uma escolha acertada, proporcionando os benefícios de desacoplamento, manutenibilidade e evolução independente que justificaram sua adoção desde as etapas iniciais do projeto.

A implementação bem-sucedida do *software* permitiu atingir a grande maioria dos objetivos de produto estabelecidos durante a *Lean Inception*. O GuardItens efetivamente elimina planilhas manuais e centraliza dados confiáveis em um único repositório, oferece rastreamento completo do status e responsável por cada recurso, e gera *insights* valiosos através de seu dashboard analítico. Na dimensão de governança, o produto aumenta a mitigação de riscos e promove economia de custos através do controle preciso do ciclo de vida dos ativos. A experiência do usuário, outro pilar fundamental, foi contemplada com uma interface intuitiva que reduz significativamente o tempo despendido em processos burocráticos de inventário e auditoria.

Do ponto de vista técnico, as decisões arquiteturais mostraram-se adequadas aos desafios encontrados. O uso de BullMQ com Redis para operações assíncronas, particularmente no envio de e-mails com Nodemailer, provou ser uma escolha robusta para garantir desempenho e confiabilidade. A componentização baseada em domínios no *front-end* facilitou a manutenção e evolução das funcionalidades, enquanto a separação em camadas do *back-end* permitiu o isolamento das regras de negócio, tornando a aplicação mais resiliente a mudanças em tecnologias subjacentes.

Embora o cálculo automatizado de depreciação tenha ficado como trabalho futuro devido à sua complexidade contábil específica, as demais funcionalidades essenciais foram implementadas com sucesso, incluindo gestão completa de ativos, vinculação de assinaturas, controle de usuários e centros de custo, e geração de QR Codes para identificação física dos equipamentos. O produto entregue demonstra maturidade técnica e adequação aos requisitos do domínio de gestão patrimonial de TI, oferecendo uma base sólida para evoluções futuras.

Por fim, este trabalho não apenas resultou em uma aplicação web funcional, mas também validou a eficácia das escolhas metodológicas e arquiteturais adotadas. A combinação entre abordagens ágeis, princípios de arquitetura limpa e desenvolvimento orientado a domínios mostrou-se produtiva para a criação de software sustentável, que permite maior manutenabilidade e baixo acoplamento entre as partes, além de ser alinhado com as ne-

cessidades do negócio, constituindo um caso de estudo relevante para projetos similares em ambientes corporativos.

Referências

ABES. *Manual ABES de Gestão de Ativos de Software*. São Paulo, 2020. 31 p. Acesso em: 19 abril 2025. Disponível em: <https://abes.com.br/wp-content/uploads/2020/06/manual-ABES.pdf>. Citado 3 vezes nas páginas 18, 22 e 26.

ABES. *Dados do Setor de Software e Serviços*. [S.l.], 2022. 32 p. Acesso em: 11 junho 2025. Disponível em: <https://abes.com.br/dados-do-setor/>. Citado 2 vezes nas páginas 22 e 24.

Atlassian. *IT Asset Management Framework*. 2023. Acesso em: 22 abril 2025. Disponível em: <https://www.atlassian.com/itsm/asset-management>. Citado 3 vezes nas páginas 19, 20 e 21.

Atlassian. *Kanplan: o que é e como usar*. 2023. Acesso em: 23 agosto 2025. Disponível em: <https://www.atlassian.com/br/agile/kanban/kanplan>. Citado 2 vezes nas páginas 52 e 53.

Atlassian. *Épicos, histórias e temas*. 2023. Acesso em: 23 agosto 2025. Disponível em: <https://www.atlassian.com/br/agile/project-management/epics-stories-themes>. Citado 2 vezes nas páginas 54 e 55.

AXELOS. *ITIL Foundation: ITIL 4 Edition*. London: The Stationery Office, 2019. 385 p. Acesso em: 16 abril 2025. Citado na página 18.

Axios. *Axios Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://axios-http.com/docs/intro>. Citado na página 74.

BOERGER, J. *Rethinking Information Technology Asset Management*. Nova York: Business Expert Press, 2021. 161 p. Acesso em: 19 abril 2025. Citado 2 vezes nas páginas 19 e 20.

BONHAM, S. S. *IT Project Portfolio Management*. Norwood: Artech House, 2005. 255 p. Acesso em: 19 abril 2025. Citado na página 18.

BULLMQ. *BullMQ Documentation*. 2024. Acesso em: 19 novembro 2025. Disponível em: <https://docs.bullmq.io/>. Citado na página 80.

CAROLI, P. *Lean Inception: How to Align People and Build the Right Product*. 1. ed. [s.n.], 2024. Acesso em: ago. 2024. Disponível em: <https://caroli.org/en/lean-inception-4/>. Citado 2 vezes nas páginas 28 e 29.

DELOITTE. *IT Asset Management (ITAM) Global Survey 2021*. 2021. Acesso em: 10 abril 2025. Disponível em: <https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Risk/gx-risk-deloitte-global-itam-survey-results-2021.pdf>. Citado 3 vezes nas páginas 13, 14 e 24.

Docker. *Docker Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://docs.docker.com/>. Citado na página 77.

EVANS, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. [S.l.]: Addison-Wesley Professional, 2003. Acesso em: 19 novembro 2025. Citado na página 82.

FAGUNDES, E. M. E. Cobit um kit de ferramentas para a excelência na gestão de ti. *São Paulo. Disponível em:< http://www. efagundes. com/Artigos/COBIT. htm>. Acesso em: 08 abril 2025*, v. 22, 2006. Citado 3 vezes nas páginas 11, 13 e 14.

Fastify. *Fastify Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://www.fastify.io/docs/latest/>. Citado na página 75.

Figma. *Figma Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://help.figma.com/hc/en-us>. Citado na página 73.

Figma Inc. *High-fidelity prototyping*. 2023. Acesso em: 15 out. 2023. Disponível em: <https://www.figma.com/resource-library/high-fidelity-prototyping/>. Citado na página 63.

Flexera. *2024 State of ITAM Report*. 2024. Acesso em: 23 abril 2025. Disponível em: <https://info-flexera-com.translate.goog/ITAM-REPORT-State-of-IT-Asset-Management?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc>. Citado 2 vezes nas páginas 20 e 21.

Gartner Peer Insights. *Enterprise Asset Management Software Market Reviews*. 2023. Acesso em: 11 junho 2025. Disponível em: <https://www.gartner.com/reviews/market/enterprise-asset-management-software>. Citado 3 vezes nas páginas 12, 14 e 25.

GitHub. *GitHub Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://docs.github.com/>. Citado na página 72.

GREEN, P.; HELSTROM, J. Asset valuation methods. *Journal of IT Governance*, v. 12, p. 20–25, 2011. Acesso em: 19 abril 2025. Citado na página 18.

Institute of Asset Management. *PAS 55:2008 Asset Management*. London, 2008. 72 p. Acesso em: 19 abril 2025. Disponível em: <https://theiam.org/knowledge-library/bsi-pas-55/>. Citado na página 19.

International Data Corporation. *IDC FutureScape: Latin America IT Industry 2023 Predictions*. 2023. <https://www.idc.com>. Acesso em: 10 jun. 2025. Citado na página 26.

International Organization for Standardization. *ISO/IEC 55000:2014 - Asset management*. 2014. Citado na página 18.

JWT. *JSON Web Token RFC 7519*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://tools.ietf.org/html/rfc7519>. Citado na página 76.

Lei nº 13.709. *Lei Geral de Proteção de Dados Pessoais*. 2018. <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 10 jun. 2025. Citado 2 vezes nas páginas 22 e 25.

LTD., R. *Redis Queue Use Case*. 2024. Acesso em: 19 novembro 2025. Disponível em: <https://redis.io/docs/latest/develop/use-cases/queues/>. Citado na página 80.

MARTIN, R. C. *Clean Code: A Handbook of Agile Software Craftsmanship*. [S.l.]: Prentice Hall, 2012. Acesso em: 19 novembro 2025. Citado na página 80.

MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. [S.l.]: Prentice Hall, 2017. Acesso em: 18 setembro 2025. ISBN 978-0134494166. Citado 3 vezes nas páginas 79, 81 e 82.

Material-UI. *Material-UI Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://mui.com/material-ui/getting-started/>. Citado na página 74.

MIRO. *O que é um roadmap?* 2024. Acesso em: 18 maio 2025. Disponível em: <https://miro.com/pt/agile/o-que-e-roadmap/>. Citado na página 61.

Miro. *Miro Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://help.miro.com/hc/en-us>. Citado na página 72.

Mordor Intelligence. *Tamanho do mercado de gestão de ativos de TI e análise de ações – Tendências e previsões de crescimento (2024 – 2029)*. 2023. Acesso em: 06 maio 2025. Disponível em: <https://www.mordorintelligence.com/pt/industry-reports/it-asset-management-market>. Citado 5 vezes nas páginas 11, 12, 14, 23 e 24.

NATAL, R. Gerenciamento de ativos de tecnologia da informação – o elo perdido do itil. *Revista Meio Byte – Comunicamos Tecnologia*, 2010. Acesso em: 08 abril 2025. Disponível em: <http://www.meiobyte.net/MB007/MB007.pdf>. Citado 5 vezes nas páginas 11, 13, 14, 20 e 21.

Node.js. *Node.js Documentation*. 2024. Acesso em: 18 setembro 2025. Disponível em: <https://nodejs.org/docs/latest/api/>. Citado na página 75.

ORTOLANI, L. F. B. A tecnologia da informação na administração pública. *Revista Bate Byte*, 1995. Acesso em: 08 abril 2025. Citado 4 vezes nas páginas 11, 13, 14 e 15.

PostgreSQL. *PostgreSQL Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://www.postgresql.org/docs/>. Citado na página 78.

Prisma. *Prisma Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://www.prisma.io/docs/>. Citado na página 75.

React. *React Documentation*. 2024. Acesso em: 18 setembro 2025. Disponível em: <https://react.dev/learn>. Citado na página 73.

SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. 2020. Acesso em: 22 agosto 2025. Disponível em: <https://www.scrum.org/resources/scrum-guide>. Citado 4 vezes nas páginas 49, 50, 52 e 58.

SILVA, S. R. T. d. *Relacionamento empresa-cliente na inovação: estudo aplicado à indústria têxtil e vestuário portuguesa*. Tese (Doutorado) — Universidade Beira Interior, Covilhã, 2021. Citado na página 27.

Swagger. *Swagger Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://swagger.io/docs/>. Citado na página 75.

TypeScript. *TypeScript Documentation*. 2024. Acesso em: 18 setembro 2025. Disponível em: <https://www.typescriptlang.org/docs/>. Citado na página 76.

VERNON, V. *Implementing Domain-Driven Design*. [S.l.]: Addison-Wesley Professional, 2013. Acesso em: 19 novembro 2025. Citado 2 vezes nas páginas 80 e 82.

Vite. *Vite Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://vitejs.dev/guide/>. Citado na página 74.

Vitest. *Vitest Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://vitest.dev/>. Citado na página 77.

Zod. *Zod Documentation*. 2025. Acesso em: 18 setembro 2025. Disponível em: <https://zod.dev/>. Citado na página 76.