INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE MINAS GERAIS – CAMPUS FORMIGA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Lucas Guimarães Bernardes

# DESENVOLVIMENTO DE SISTEMA WEB PARA GESTÃO DE ATIVOS MÓVEIS EMPRESARIAIS

Formiga – MG
2025

LUCAS GUIMARÃES BERNARDES

# DESENVOLVIMENTO DE SISTEMA WEB PARA GESTÃO DE ATIVOS MÓVEIS EMPRESARIAIS

Trabalho de conclusão de curso apresentado ao Curso Bacharelado em Administração do Instituto Federal de Minas Gerais - *Campus Formiga* para obtenção do grau de bacharel em Administração.

Orientador: Roger Santos Ferreira

Formiga – MG

2025

Bernardes, Lucas Guimarães
B005d Desenvolvimento de sistema web para gestão de ativos moveis empresariais. / Lucas Guimarães Bernardes – Formiga : IFMG, 2025.
75 p. :il. color.

Orientador: Prof. Me. Roger Santos Ferreira
Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Gestão de ativos móveis. 2. Sistema web. 3. Laravel. 4. PostgreSQL. 5. MVC I.
Ferreira, Roger Santos. II. Título.

CDD 005

*Ficha catalográfica elaborada pela Bibliotecária Lívia Renata Santos- CRB/6-2561*

**MINISTÉRIO DA EDUCAÇÃO**
**SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA**
**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS**
**Campus Formiga**
**Diretoria de Ensino**
**Docência Área Acadêmica de Computação**
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

**LUCAS GUIMARÃES BERNARDES**

**DESENVOLVIMENTO DE SISTEMA WEB PARA GESTÃO DE ATIVOS MÓVEIS EMPRESARIAIS**

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

**APROVADO** em 4 de dezembro de 2025.

BANCA EXAMINADORA

Prof. Roger Santos Ferreira (Orientador)

Prof. Manoel Pereira Junior (IFMG)

Prof. Fernando Paim Lima (IFMG)

Documento assinado eletronicamente por **Manoel Pereira Junior**, **Professor**, em 04/12/2025, às 16:48, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Roger Santos Ferreira**, **Professor**, em 04/12/2025, às 17:04, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Paim Lima**, **Professor**, em 04/12/2025, às 20:29, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **2547404** e o código CRC **CCBC8437**.

| | |
|---|---|
| 23211.001583/2024-91 | 2547404v1 |

# Resumo

Este trabalho descreve o desenvolvimento de um sistema web para gestão de ativos móveis empresariais, realizado em parceria com uma empresa da cidade de Formiga, MG que buscava substituir o uso de planilhas manuais por uma solução automatizada. O sistema, construído com o *framework* Laravel e o banco de dados PostgreSQL, centraliza o controle de aluguéis, valores financeiros e orçamentos, atendendo às demandas específicas da empresa parceira, que enfrentava desafios como a dispersão de dados, riscos de inconsistências e dificuldades no monitoramento em tempo real. A metodologia incluiu o levantamento de requisitos junto aos gestores da empresa, identificando necessidades como rastreamento de ativos, gestão de contratos, cálculo automático de custos e geração de relatórios integrados. A arquitetura MVC do Laravel permitiu a criação de uma estrutura modular e escalável, enquanto o PostgreSQL garantiu segurança e eficiência no armazenamento de dados transacionais e históricos. O sistema oferece interfaces intuitivas, desenvolvidas com HTML5 e CSS3, além de funcionalidades como autenticação de usuários, controle de permissões e *dashboard* analítico. Espera-se que a implementação da ferramenta resulte na redução de erros operacionais, na padronização de processos e na agilidade na recuperação de informações a longo prazo, eliminando a dependência de planilhas fragmentadas. O trabalho também destaca planos futuros, como a análise do impacto operacional gerado por essa solução, a construção de um aplicativo móvel para o auxílio operacional do sistema e o incremento de mecanismos auxiliares de monitoramento do sistema.

**Palavras-chave:** Gestão de ativos móveis, Sistema Web, Laravel, PostgreSQL, MVC.

# Abstract

This paper describes the development of a web-based system for managing corporate mobile assets, carried out in partnership with a company in the city of Formiga, MG, which sought to replace the use of manual spreadsheets with an automated solution. The system, built with the Laravel framework and PostgreSQL database, centralizes the control of rentals, financial values, and budgets, meeting the specific demands of the partner company, which faced challenges such as data dispersion, risks of inconsistencies, and difficulties in real-time monitoring. The methodology included gathering requirements from company managers, identifying needs such as asset tracking, contract management, automatic cost calculation, and integrated report generation. Laravel's MVC architecture allowed for the creation of a modular and scalable structure, while PostgreSQL ensured security and efficiency in the storage of transactional and historical data. The system offers intuitive interfaces developed with HTML5 and CSS3, as well as features such as user authentication, permission control, and an analytical dashboard. The implementation of the tool is expected to result in a reduction in operational errors, standardization of processes, and agility in long-term information retrieval, eliminating the dependence on fragmented spreadsheets. The work also highlights future plans, such as analyzing the operational impact generated by this solution, building a mobile application to assist with system operations, and increasing auxiliary system monitoring mechanisms.

**Keywords:** Asset management, Web development, Laravel, PostgreSQL, MVC.

# LISTA DE FIGURAS

Figura 1: Camadas da Engenharia de Software....13
Figura 2: Modelo incremental de desenvolvimento....15
Figura 3: Estrutura do processo iterativo e incremental....17
Figura 4: Estrutura típica de um documento HTML....19
Figura 5: Estrutura de uma regra em CSS....20
Figura 6: Representação visual do modelo MVC....23
Figura 7: Caso de uso da proposta inicial do projeto....25
Figura 8: Tabelas do banco para controle de acesso....37
Figura 9: Tabelas do banco para controle de patrimônio....38
Figura 10: Tabelas do banco para controle de aluguel....39
Figura 11: Tabelas do banco para controle de manutenção....40
Figura 12: Tela de autenticação do sistema....53
Figura 13: Tela principal....54
Figura 14: Continuação da tela principal....55
Figura 15: Tela de consulta de aluguéis....56
Figura 16: Tela de cadastro de aluguel....57
Figura 17: Tela de listagem de manutenções....58
Figura 18: Tela de cadastro de manutenção....59
Figura 19: Tela de checklist de cadastro de manutenção....59
Figura 20: Acesso para revisão de manutenção....60
Figura 21: Tela de revisão de manutenção....61
Figura 22: Continuação da tela de revisão de manutenção....61
Figura 23: Tela de listagem de unidades de negócio....62
Figura 24: Tela de cadastro de unidade de negócio....63
Figura 25: Tela de listagem de orçamentos....64
Figura 26: Tela de cadastro de orçamento....64
Figura 27: Menu do sistema....65
Figura 28: Menu do sistema continuado....65

# LISTA DE SIGLAS E ABREVIATURAS

MVC – *Model View Controller*

HTTP – *Hypertext Transfer Protocol*

HTML – *Hypertext Markup Language*

IDE – *Integrated Development Environment*

PHP – *PHP: Hypertext Preprocessor*

SGML – *Standard Generalized Markup Language*

URL – *Uniform Resource Locator*

# SUMÁRIO

1 Introdução....11
1.1 Objetivo geral....11
1.2 Objetivos específicos....12
2 Referencial teórico....13
2.1 Automatização de processos....13
2.2 Desenvolvimento de *software*....14
2.2.1 Desenvolvimento iterativo e incremental....16
2.3 Desenvolvimento Web....19
2.3.1 O lado *frontend*....20
2.3.2 O lado *backend*....23
3 Metodologia....25
3.1 Levantamento de requisitos....26
3.2 Ambiente de trabalho....29
3.3 Estudo de tecnologias....29
3.4 Ferramentas utilizadas....33
3.5 Etapas de desenvolvimento....37
3.5.1 Modelagem e Estruturação do Banco de Dados....38
3.5.2 Primeira entrega: A Base Operacional....45
3.5.3 Segunda entrega: Manutenção e Inteligência de Negócio....46
3.5.4 Terceira entrega: Refinamento de Acessos e Comunicação....47
3.5.5 Quarta entrega: Reformulação e Ajustes Estruturais....47
4 Desenvolvimento do projeto....50
4.1 Regras de negócio....50
4.1.1 Gestão de Inventário e Unidades de Negócio....51
4.1.2 Processo de Aluguel de Equipamentos....51
4.1.3 Sistema de Manutenções Preventivas e Bonificações....52
4.1.4 Sistema de *Checklist*....52
4.1.5 Processo de Revisão e Aprovação Administrativa....53
4.1.6 Cálculos Financeiros....54
4.2 O sistema....54
4.2.1 Tela de autenticação....54

4.2.2 Tela principal.....55
4.2.3 Tela de aluguéis.....57
4.2.4 Tela de manutenções.....59
4.2.5 Tela de unidades de negócio.....65
4.2.6 Tela de orçamentos.....66
4.2.7 Menu do sistema.....67
4.2.8 Métricas do código-fonte.....68
4.3 Desafios no desenvolvimento.....69
5 Considerações finais.....71
5.1 Trabalhos futuros.....72
REFERÊNCIAS.....74

# 1 Introdução

No ambiente empresarial contemporâneo, a automação de processos tornou-se crucial para otimizar a eficiência operacional e reduzir custos. No entanto, ainda existem empresas que dependem de métodos manuais para gerenciar atividades críticas, o que frequentemente leva a falhas decorrentes de erros humanos, resultando em atrasos e custos adicionais (EMB, 2024). Empresas que atuam no setor de manutenção de vias férreas e aluguel de equipamentos para o mesmo fim são exemplos que padecem de tais desafios ao lidar com a complexidade de suas operações diárias.

Nesse contexto, o avanço da tecnologia apresenta uma solução promissora. Com o desenvolvimento dos computadores e da computação como um todo, processos comerciais anteriormente realizados de forma manual encontraram na tecnologia uma maneira eficaz de minimizar erros e otimizar a comunicação entre os setores da empresa (NONATO, 2024). A implementação de sistemas automatizados não só reduz a incidência de falhas, como também permite que os usuários finais realizem processos de forma autônoma, diminuindo a necessidade de intervenção manual e, consequentemente, aumentando a eficiência operacional.

No entanto, o processo de desenvolvimento de uma solução computacional eficiente e capaz de atender às necessidades é uma tarefa complexa e composta de múltiplas partes. Grande parte do projeto é dedicada à engenharia de software, que atua como ferramenta de planejamento e definição da arquitetura da aplicação. No trabalho de desenvolvimento de um software são envolvidas muitas etapas além da simples programação, também é necessária a construção de manuais de uso, documentação da arquitetura e do planejamento de recursos configuráveis para permitir que o mesmo se adapte à necessidade do usuário (SOMMERVILLE, 2011).

## 1.1 Objetivo geral

O objetivo deste projeto é desenvolver um protótipo de sistema web para gestão de ativos móveis empresariais que substitua o uso de planilhas manuais, proporcionando maior automação, centralização e confiabilidade no controle das informações. A solução busca

atender às demandas da empresa parceira ao oferecer funcionalidades como rastreamento de ativos, gestão de contratos, cálculo automático de custos e geração de relatórios, promovendo padronização de processos, redução de erros e maior agilidade na tomada de decisões.

## 1.2 Objetivos específicos

1. Controle de inventário: desenvolver um módulo capaz de registrar e monitorar todos os equipamentos sob responsabilidade da empresa, garantindo que cada ativo possua informações completas sobre sua identificação, estado atual, valor de aquisição, valor de aluguel e histórico de movimentações. Esse controle deve permitir a atualização contínua dos dados, assegurando precisão nas consultas e maior confiabilidade no processo de tomada de decisões relacionadas à alocação e utilização dos ativos.
2. Controle de manutenção: implementar funcionalidades que possibilitem o registro detalhado de manutenções preventivas e corretivas realizadas em cada equipamento. O sistema deve permitir anexar imagens como evidência das atividades executadas, de forma a garantir a rastreabilidade e transparência dos processos. Além disso, deve ser possível que o time administrativo realize a revisão desses registros, aprovando ou recusando os cadastros, assegurando que somente informações validadas componham o histórico do ativo.
3. Controle de aluguéis: disponibilizar recursos que permitam que os responsáveis pelas obras clientes possam realizar orçamentos de novos equipamentos diretamente pelo sistema, eliminando a necessidade de processos manuais. O módulo deve também gerenciar os itens atualmente alugados, possibilitando acompanhar a quantidade, prazos, custos associados e status dos contratos em vigor.
4. Visualização de dados: oferecer um mecanismo de consulta e visualização integrado que apresente, de forma clara e acessível, informações consolidadas sobre a situação do inventário, os custos e prazos de aluguéis, o andamento das unidades de negócio e os valores financeiros em operação. O sistema de visualização deve possibilitar filtros e segmentações, de modo a permitir diferentes perspectivas de análise, como a avaliação da saúde financeira do negócio, a verificação da disponibilidade de ativos para novas demandas e o acompanhamento do status operacional em tempo real.

# 2 Referencial teórico

Nesse capítulo são apresentados os conceitos fundamentais que embasam o desenvolvimento desse trabalho. Inicialmente, é abordado o tema da automação de processos empresariais, explorando como a tecnologia pode ser aplicada para otimizar operações e aumentar a eficiência organizacional. Em seguida, são discutidos os princípios da Engenharia de Software, apresentando as metodologias e práticas que orientaram a construção desse projeto. Por fim, o capítulo aborda o desenvolvimento de aplicações web, enfatizando as tecnologias e ferramentas utilizadas no projeto. Esses fundamentos teóricos fornecem a base necessária para compreender as decisões técnicas e metodológicas adotadas no desenvolvimento do sistema proposto.

## 2.1 Automatização de processos

A automação de processos empresariais é uma área amplamente discutida na literatura, destacando-se como uma estratégia essencial para aumentar a eficiência operacional e reduzir custos. Segundo estudos, a substituição de processos manuais por sistemas automatizados não apenas minimiza erros humanos, mas também melhora a precisão das informações e a agilidade dos fluxos de trabalho (EMB, 2024). A automatização é particularmente relevante em empresas como a empresa em questão, onde a complexidade das operações, como o controle de inventário e gestão de alugueis de equipamentos, exige uma abordagem sistemática e precisa para evitar inconsistências e otimizar recursos.

Erros introduzidos no sistema como consequência de erros humanos como falhas em cadastros tendem a causar consequências financeiras significativas. Um valor inserido incorretamente pode significar o encargo de custos muito maiores que os inicialmente previstos para uma determinada operação da empresa. Fatores que podem acarretar ainda mais em problemas são o cansaço de trabalhadores, muitas vezes derivado da sobrecarga de tarefas, mas também os déficits no treinamento e capacitação de um funcionário (VIEIRA, 2023).

O processo de automação consiste na implementação de sistemas tecnológicos que substituem atividades tradicionalmente executadas de forma manual nas organizações. Trata-se de uma estratégia que utiliza diferentes recursos tecnológicos e ferramentas computacionais

com o propósito de tornar as operações mais eficientes, uniformes, rápidas e seguras. A implementação de sistemas automatizados pode variar significativamente em complexidade. Enquanto algumas aplicações envolvem operações relativamente simples, como o disparo automático de mensagens eletrônicas de confirmação, outras englobam processos sofisticados relacionados ao gerenciamento de projetos e à coordenação da produção industrial (NONATO, 2024).

## 2.2 Desenvolvimento de *software*

O desenvolvimento bem-sucedido de um produto de *software* exige, primeiramente, um esforço fundamental para compreender o problema antes de sequer começar a desenvolver uma solução. Nesse sentido, a Engenharia de Software atua como um mecanismo de apoio essencial, pois é a disciplina que impõe os processos necessários em um trabalho que, de outra forma, pode se tornar caótico (PRESSMAN; MAXIM, 2021).

A Engenharia de Software engloba diferentes elementos que trabalham em conjunto: processos que definem quais atividades devem ser realizadas e quais produtos serão gerados, métodos que orientam a execução dessas atividades, e ferramentas que auxiliam na realização do trabalho. Além disso, ela estabelece critérios de qualidade tanto para os processos quanto para os produtos desenvolvidos (HIRAMA, 2011). Dessa forma, ao aplicar esses princípios, é possível obter produtos com maior qualidade e alcançar melhor produtividade no desenvolvimento.

Segundo Sommervile (2011, p. 18), apesar de existirem múltiplos processos distintos para o desenvolvimento de um *software*, todos incluem quatro atividades fundamentais em comum relacionadas à engenharia de *software*, essas são:

1. A especificação do *software*, que detalha as funcionalidades e restrições de operação
2. O projeto e a implementação do *software*, onde se é realmente construído o produto
3. A validação, onde são realizadas análises para se confirmar que o produto implementado corresponde às expectativas iniciais dos clientes e cumpre com os objetivos.
4. A evolução do *software*, que trata da capacidade do mesmo de se adaptar às eventuais mudanças de utilização, bem como as novas demandas do cliente para atender a novos desafios e objetivos.

A Engenharia de *Software* é um processo que, inicialmente proposto por Friedrich Ludwig na década de 1960, evoluiu ao longo dos anos impulsionada pela necessidade crítica de aplicar uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento de *software*, já que com o aumento da complexidade dos sistemas também aumentavam-se as falhas, que poderiam ter consequências catastróficas (PRESSMAN; MAXIM, 2021).

Para PRESSMAN e MAXIM (2011), a Engenharia de *Software* tem evoluído de certa forma a representar uma estrutura matemática que pode ser bem entendida como uma tecnologia em níveis de diferentes responsabilidades. Pode-se visualizar na Figura 8 uma representação visual desses níveis.

*Figura 1: Camadas da Engenharia de Software*


*Fonte: HIRAMA, 2011*

A base da figura, formada pela camada “Foco em Qualidade” apresenta a ênfase à preocupação com a excelência do produto final. Já na camada de processo existe o mecanismo para a integração das camadas de métodos e ferramentas para que se possa desenvolver um produto nos prazos e requisitos estabelecidos. A camada de métodos aborda as atividades necessárias para a construção de um software, abrangendo um conjunto amplo de atividades como a análise de requisitos, projeto, implementação, testes e manutenção. Por fim a camada de ferramentas estabelece o apoio de ferramentas automatizadas ou semi-automatizadas para o auxílio nesse processo (HIRAMA, 2011).

Para o desenvolvimento deste projeto, foi adotada uma abordagem baseada na metodologia Iterativa e Incremental. Essa escolha se justifica pela proposta de que o sistema seja futuramente integrado ao conjunto de ferramentas utilizadas pela empresa parceira, o que exige um acompanhamento próximo e constante do processo de desenvolvimento. Essa metodologia é detalhada no tópico 2.2.1.

A metodologia possibilitou que a empresa parceira acompanhasse continuamente o andamento do trabalho, participasse ativamente das revisões e solicitasse ajustes necessários em cada fase. Dessa forma, foi possível garantir que a ferramenta estivesse sempre alinhada com as necessidades reais da operação e que eventuais desvios ou melhorias pudessem ser incorporados antes da conclusão do projeto.

### 2.2.1 Desenvolvimento iterativo e incremental

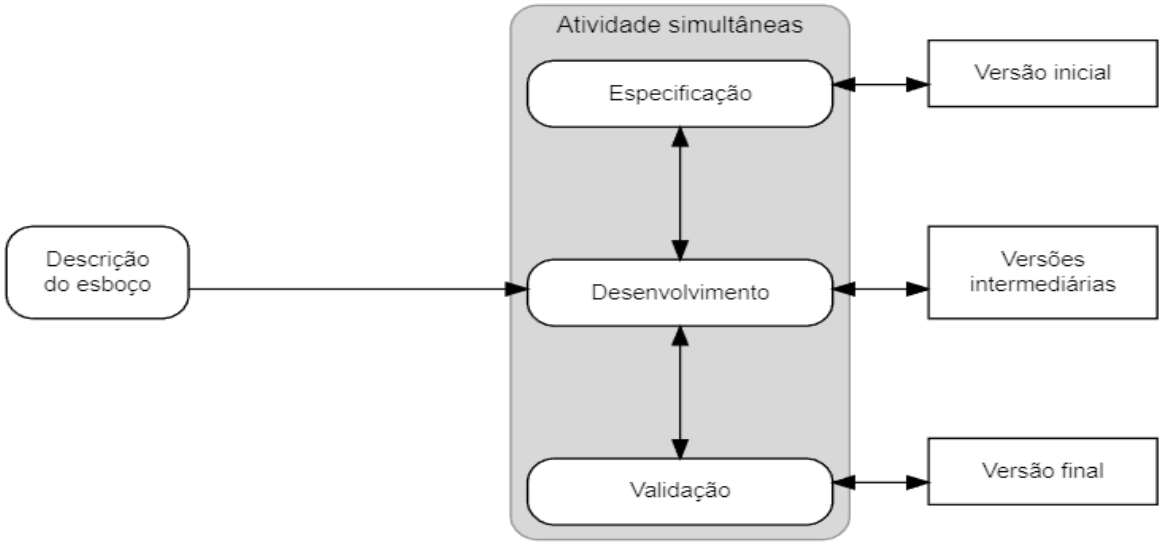
O modelo incremental é um dos modelos clássicos de engenharia de *software* criado com o propósito de ser uma evolução do modelo em cascata, construído com base na ideia de que o desenvolvimento acontece com certas etapas que se repetem a cada evolução do *software*. Nesse modelo inicialmente é realizado um levantamento de requisitos para o projeto como um todo, com um planejamento inicial das etapas de desenvolvimento e as funcionalidades a serem implementadas em cada uma, logo então seguido de um processo em repetição de implementação e validação (NASCIMENTO, 2015). Na Figura 2 é possível visualizar uma representação visual desse processo.

O desenvolvimento incremental reflete de maneira bastante fiel a forma como a maioria dos problemas complexos são resolvidos no mundo real. Por essa razão, essa abordagem se adapta especialmente bem a sistemas voltados para o ambiente corporativo, como sistemas de negócios, plataformas de *e-commerce* e aplicações personalizadas para contextos específicos. Na prática, é bastante raro que uma solução completa e definitiva para um problema seja construída logo na primeira tentativa. O mais comum é que a solução seja desenvolvida gradualmente, através de múltiplas etapas de análise, implementação e validação. Durante esse processo, é natural que surjam percalços, descobertas de novos requisitos ou mudanças nas necessidades do cliente, o que acaba exigindo ajustes, correções e até mesmo reestruturações em partes do sistema já desenvolvidas (SOMMERVILLE, 2011).

É justamente nesse cenário que o modelo incremental demonstra sua maior vantagem. Ao dividir o desenvolvimento em entregas menores e sucessivas, problemas e necessidades de mudança são identificados muito mais cedo no processo, quando o software ainda está em fase de construção e é mais flexível para ajustes. Isso torna o processo significativamente mais econômico, pois corrigir ou modificar funcionalidades durante o desenvolvimento é muito menos custoso do que ter que refazer grandes partes de um sistema já finalizado. Além disso, o cliente consegue visualizar e validar o produto aos poucos, reduzindo o risco de que o resultado final não atenda às expectativas ou necessidades reais da operação (BEZERRA, 2007).

É comum que nesse modelo as funcionalidades mais importantes e urgentes sejam as primeiras a serem desenvolvidas. Dessa forma, é possível que o cliente visualize ainda cedo no processo do desenvolvimento se o produto de *software* está atendendo às suas demandas e, para o caso da necessidade de alterações, não ocorre uma quantidade significativa de retrabalho da aplicação já que a mesma ainda se encontra em estágios iniciais (SOMMERVILLE, 2011).

*Figura 2: Modelo incremental de desenvolvimento*




*Fonte: Adaptado de SOMMERVILLE, 2011*

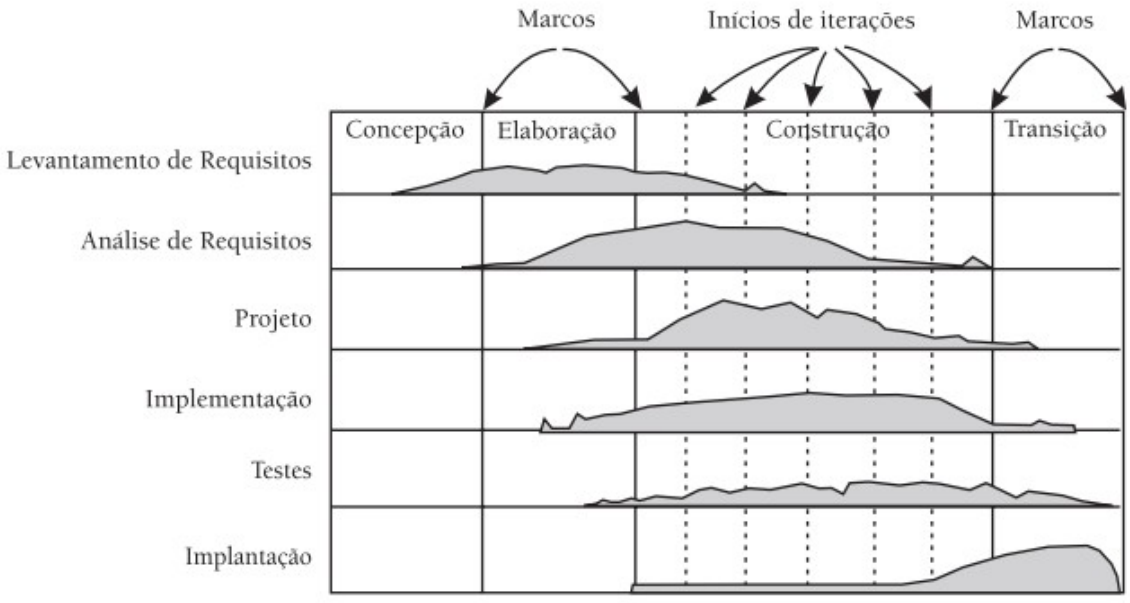
O ciclo de vida do processo iterativo e incremental pode ser compreendido através de duas dimensões fundamentais: a dimensão temporal e a dimensão dos fluxos de trabalho,

conforme apresentado por Bezerra (2007). A Figura 3 ilustra visualmente como essas duas dimensões se inter-relacionam e formam a estrutura completa do processo de desenvolvimento.

A dimensão temporal organiza o desenvolvimento em fases bem delimitadas. Dentro de cada fase podem ocorrer uma ou mais iterações, e ao término de cada iteração é produzido um incremento do sistema, ou seja, uma versão funcional que adiciona novas capacidades ao produto. Esses incrementos podem ser disponibilizados aos usuários finais para validação ou mantidos como versões internas para testes e refinamentos. Cada fase é concluída através de um marco, que representa um ponto de controle importante onde decisões estratégicas sobre o projeto são tomadas, objetivos são avaliados e o progresso é medido (BEZERRA, 2007).

Já a dimensão dos fluxos de trabalho engloba o conjunto de atividades técnicas realizadas durante cada iteração, independentemente da fase em que o projeto se encontra. Essas atividades incluem o levantamento de requisitos, a análise e especificação desses requisitos, o projeto da solução, a implementação do código, os testes de verificação e validação, e finalmente a implantação. É importante destacar que, embora essas atividades estejam presentes em todas as fases, a proporção e a intensidade com que cada uma delas é executada varia conforme o momento do desenvolvimento. Por exemplo, nas fases iniciais há maior concentração em levantamento e análise de requisitos, enquanto nas fases intermediárias predominam as atividades de projeto e implementação, e nas fases finais ganham destaque os testes e a implantação do sistema (BEZERRA, 2007).

*Figura 3: Estrutura do processo iterativo e incremental*




*Fonte: BEZERRA,* 2007

## 2.3 Desenvolvimento Web

Inventada em 1992 por Tim Berners-Lee, a *web* foi um conjunto de tecnologias construídas por ele para que possibilitasse aos cientistas do mundo inteiro a criação de textos e pesquisas, de forma eletrônica, possibilitando o fácil compartilhamento e interligação de conteúdos. O núcleo dessa primeira versão da *web* era composto do protocolo *Hypertext Transfer Protocol* (HTTP) e da linguagem de marcação de conteúdo *Hypertext Markup Language* (HTML) (BONIATI e SILVA, 2013).

As aplicações na *web* operam no modelo cliente/servidor, onde uma aplicação de retaguarda – o servidor – recebe requisições via rede por clientes para que forneça o conteúdo de um determinado documento. Nesse contexto os servidores são comumente nomeados de *webservers*, enquanto a aplicação cliente recebe o nome de Navegador. Os navegadores são as aplicações responsáveis por receber o conteúdo em HTML e criar uma representação visual amigável para o usuário, permitindo a interpretação do conteúdo do documento. No contexto de uma aplicação, o código que executa no lado cliente é denominado de *frontend*, enquanto o código que executar no servidor é *backend*. (BONIATI; SILVA, 2013).

### 2.3.1 O lado *frontend*

O HTML é uma linguagem de marcação padrão utilizada na construção de páginas e aplicações *web*, sua função é estritamente de estruturação de conteúdo. Um documento HTML é composto por marcadores (*tags*) que adicionam significado ao texto, e que permitem a inclusão de ligações entre múltiplos textos. Em sua construção mais básica, é uma linguagem que usa elementos para descrever uma estrutura de uma página, como posicionamento de texto, tamanho e fonte (DUCKETT, 2011). Foi utilizada como base para a sua construção a especificação SGML, um método reconhecido internacionalmente com regras gerais para a criação de linguagens de marcação (SILVA, 2019). Na Figura 4 é apresentada uma estrutura típica de um documento HTML, onde são apresentadas algumas *tags* delimitadoras dos elementos da página *web*.

No momento da construção do presente projeto, o HTML encontra-se na sua versão 5, essa que acumula uma série de melhorias e adições ao HTML ocorridos durante o desenvolvimento de suas versões. O HTML 5 passa a representar muito mais do que apenas uma nova versão da linguagem, mas todo um conjunto de tecnologias auxiliares, essas que permitem a construção de recursos muito mais avançados do que aqueles que eram encontrados nos primeiros projetos *web* (SILVA, 2019).

A sintaxe do HTML é bastante flexível, algo que é derivado da sua intenção de manter a retrocompatibilidade com especificações e navegadores mais antigos. No HTML5 por exemplo, é possível utilizar regras de construção do HTML4, como *tags* **p** e **li** sem uma tag correspondente de fechamento, ou até mesmo a utilização de atributos de *tags* sem aspas (SILVA, 2019).

*Figura 4: Estrutura típica de um documento HTML*


*Fonte: DUCKETT, 2011*

A estrutura básica de uma página HTML segue alguns padrões. As *tags* **<html>**, **<head>** e **<body>** serão as mais encontradas em documentos. Essas compõem a estrutura básica da definição de uma página. O **<html>** é aberto logo no início do documento e fechado no final de tudo, e indica que o conteúdo é um texto em formato HTML, enquanto **<head>** e **<body>** indicam respectivamente, o cabeçalho e corpo do documento. No cabeçalho são inseridas definições que não necessariamente gera resultados visuais para o usuário, porém é essencial para a indicação de dados da página como autor, título, idioma, entre outros. Já o **<body>** indica o conteúdo da página, aquilo que é eventualmente é processado e exibido ao usuário (BONIATI; SILVA, 2013).

Em versões modernas do HTML, existe uma separação entre a definição do conteúdo da página e a estilização desse conteúdo. A *Cascading Style Sheet* (CSS) é uma linguagem construída especificamente para esse propósito e trabalha lado a lado com o HTML. No CSS o conteúdo do HTML é dividido em blocos e, para cada um dos blocos, são expostos parâmetros que permitem a edição de fontes, cores e tamanhos do conteúdo desses blocos. Na Figura 5 é possível visualizar como uma regra de estilização de um elemento do tipo **p** (ou parágrafo) é construída no CSS de forma a alterar a fonte do conteúdo textual (DUCKETT, 2011).

*Figura 5: Estrutura de uma regra em CSS*


*Fonte: DUCKETT, 2011*

A estrutura de regras no CSS possui duas partes: o seletor e a declaração. O seletor indica qual elemento do conteúdo de uma página que será afetado pelas regras definidas dentro da declaração. As regras podem conter múltiplos seletores, auxiliando no refinamento e precisão na seleção de um elemento específico em um documento HTML, permitindo uma estilização bastante flexível. Os seletores também podem ser construídos a partir de classes, em vez de tipos de elementos específicos, permitindo que múltiplos elementos diferentes compartilhem um mesmo conjunto de atributos (DUCKETT, 2011).

Por fim, acompanhado do HTML e do CSS, existe a linguagem JavaScript. O JavaScript é a linguagem de programação da Web. A grande maioria dos websites a utilizam e praticamente todos os navegadores modernos possuem suporte. A linguagem foi criada pela

Netscape em 1995, inicialmente com o nome de Mocha. A intenção era uma tecnologia que permitisse o processamento de dados diretamente no lado cliente da estrutura *web (GRILLO; FORTES, 2008)*.

A função primordial do JavaScript é a capacidade de criar programas que estão diretamente integrados no código HTML, possibilitando o processamento de dados, validação de formulários e alteração de valores e elementos HTML dinamicamente. Esse processamento local é fundamental pois evita a constante troca de informações com o servidor, otimizando o tempo de resposta e a latência da aplicação *(GRILLO; FORTES, 2008)*.

### 2.3.2 O lado *backend*

As tecnologias HTML, CSS e JavaScript trabalham no lado cliente da web, porém aplicações mais modernas que requerem interações dinâmicas com o usuário utilizam tecnologias *backend* para gerar conteúdos HTML, CSS e JavaScript dinamicamente. A principal tecnologia *backend* utilizada no projeto foi a linguagem PHP, acompanhada do *framework* Laravel.

A linguagem PHP é uma linguagem de *script* de propósito geral voltada para o desenvolvimento web, criada por Rasmus Lerdorf em 1993 e lançada em 1995 (PHP, [s.d.]). O PHP é executado no servidor e o resultado é retornado ao navegador como HTML puro, permitindo a criação de páginas web dinâmicas e interativas. A linguagem PHP foi adotada como requisito da empresa e permanece um dos pilares mais amplamente utilizados no desenvolvimento web.

O PHP é amplamente utilizado no desenvolvimento web, sendo que mais de 75% dos desenvolvedores preferem PHP para tarefas *server-side* devido ao seu processo de desenvolvimento rápido. A linguagem suporta programação orientada a objetos e possui vasta documentação e uma comunidade global ativa, o que garante amplo suporte técnico (STAUFFER, 2019).

O Laravel é uma *framework* PHP de desenvolvimento de aplicações focado no desenvolvimento rápido, isto é, ele toma como prioridades coisas como a redução da curva de aprendizado e do tempo entre iniciar uma aplicação e publicá-la. É um *framework* que prioriza convenção sobre configuração, isso é, o código e configurações utilizam estruturas

padrões e fixas, em vez de deixar a cargo do desenvolvedor criar suas próprias estruturas (STAUFFER, 2019).

O Laravel é voltado ao padrão arquitetônico MVC e fornece como parte do seu pacote diversas funcionalidades comuns para a construção de *websites*, dentre elas:

- Roteamento de páginas intuitivo com agrupamento de rotas, injeção de *middleware* e *route-model binding* que simplifica a consulta de entidades no banco a partir de dados obtidos através da URL
- Eloquent ORM, uma abstração orientada a objetos para acesso a bancos relacionais, com suporte a relacionamentos, query scopes e fácil manutenção das migrações de esquema do banco
- Blade, motor de templating com sintaxe clara, segurança XSS automática e capacidades como layouts, componentes e seções dinâmicas
- Artisan, uma ferramenta de linha de comando que automatiza tarefas comuns, como migrações, preenchimento de dados padrões, testes e criação de componentes, acelerando o fluxo de trabalho
- Autenticação e segurança integradas, incluindo validação de entrada, hashing de senhas e proteção contra CSRF/XSS

Sua comunidade extensa e ativa, juntamente do seu ecossistema maduro garantem uma vasta gama de materiais e ferramentas disponíveis para o auxílio do processo de desenvolvimento, reduzindo trabalho e acelerando a entrega (Gołofit, 2023).

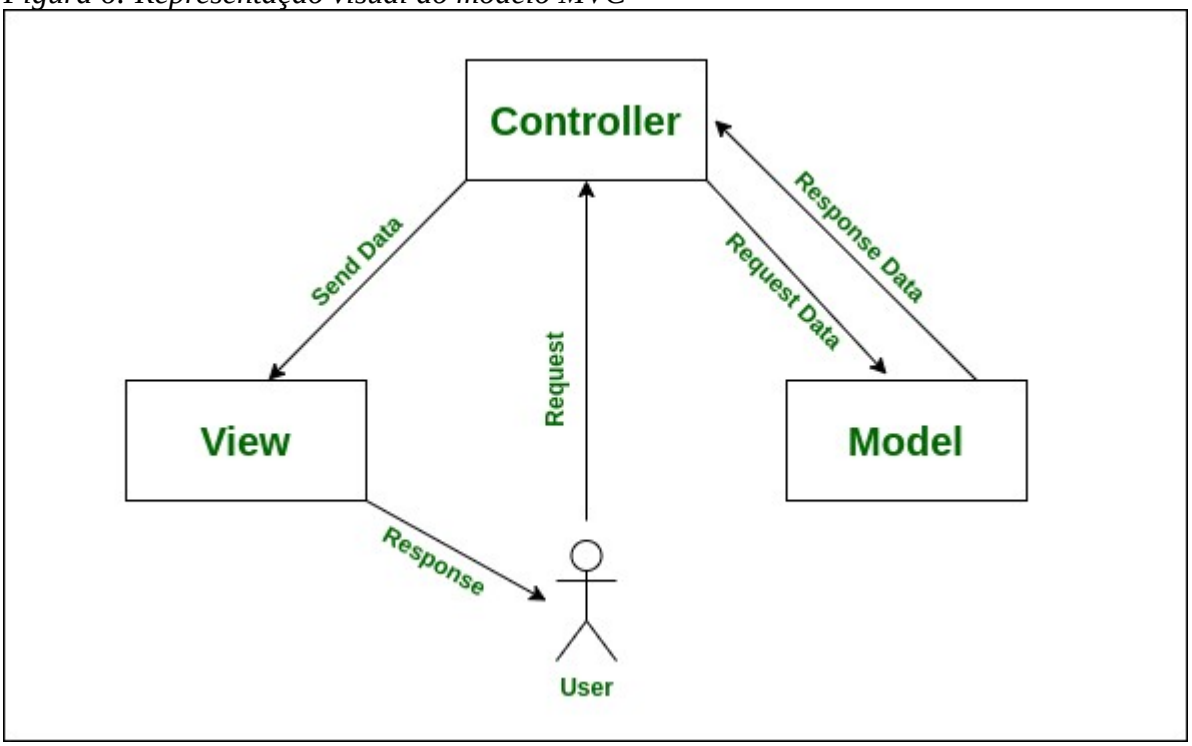
Múltiplos padrões arquitetônicos existem para a organização do código e responsabilidades um projeto de *software*, porém um ponto em comum entre todos esses formatos é a busca por um padrão que facilite a manutenção e implementação desses sistemas. O modelo Model-View-Controller (MVC) é um que propõe a divisão de responsabilidades entre 3 camadas principais de uma aplicação. Assim como diz o nome, as 3 camadas principais são: *Model* (Modelo), *View* (Visão) e *Controller* (Controle) (CORREA; VALLEJO, 2022).

1. Controle é a camada da lógica de negócio e interações do sistema. É nessa camada que é construído o núcleo operacional do sistema e que são interligadas todas as partes da estrutura.

2. Visão representa a camada visual do projeto, isto é, interfaces do usuário para acesso ao sistema. Essas interfaces podem tomar diversos formatos como interfaces gráficas e de linha de comando.
3. Modelo representa a camada de dados da aplicação. É nessa camada que é gerenciado todo o estado e armazenamento de uma aplicação, juntamente do seu acesso a sistemas de bancos de dados e outros sistemas de armazenamento.

A figura 6 apresenta visualmente como essas 3 camadas se entrelaçam, onde o usuário interage diretamente com as camadas de visão e controle, enviando e recebendo requisições ao sistema. Enquanto isso, a camada modelo é acessada como uma camada de fundo pelos sistemas internos do *software*.

*Figura 6: Representação visual do modelo MVC*




*Fonte: GeeksforGeeks, [s.d.]*

# 3 Metodologia

Nessa seção é detalhado o processo de desenvolvimento desse trabalho, como foi realizado o levantamento de requisitos, o planejamento e organização dos recursos

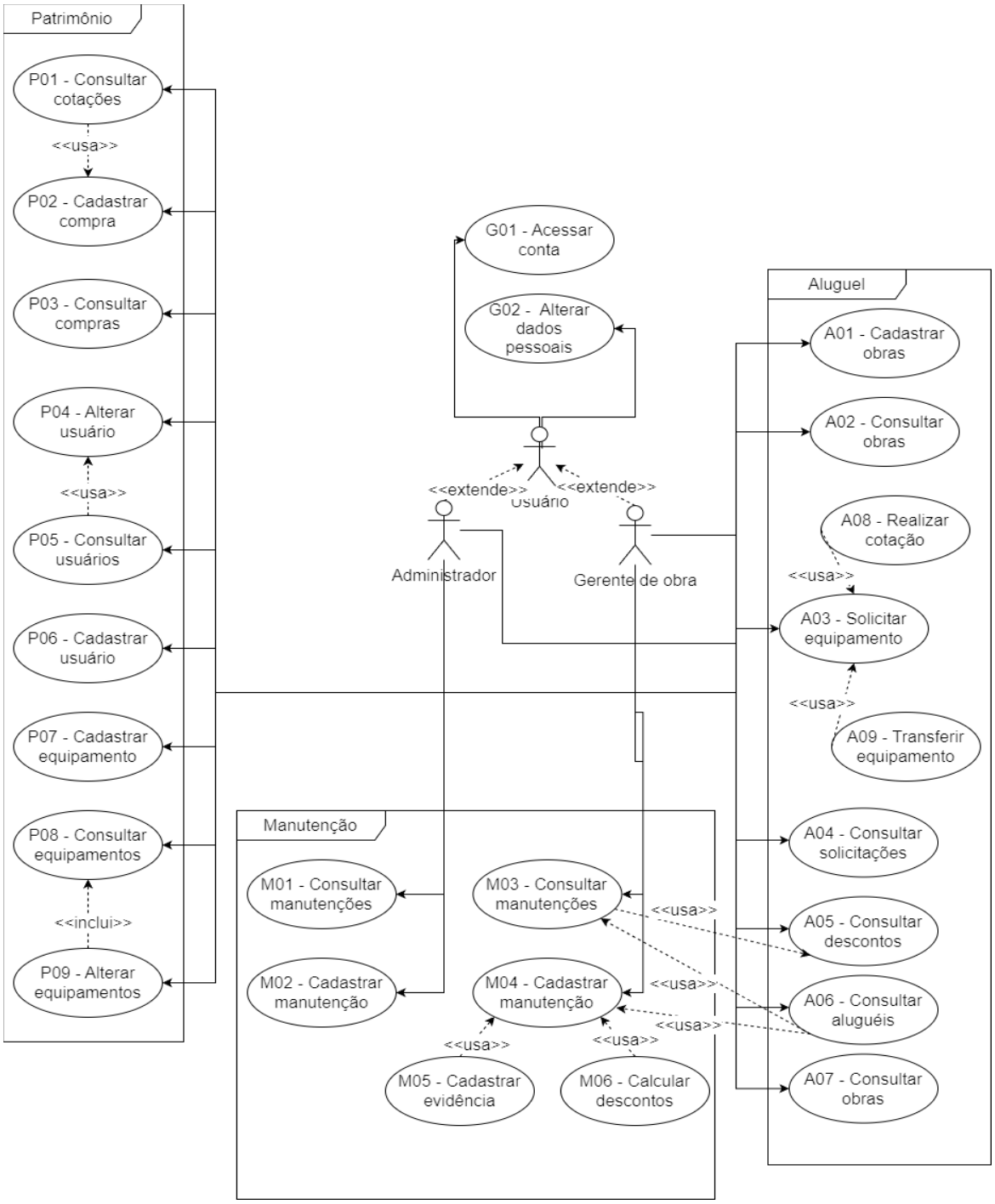
necessários para o sistema e o estudo das tecnologias necessárias para o trabalho e o processo de implementação.

## 3.1 Levantamento de requisitos

O primeiro passo no desenvolvimento do projeto foi a realização de uma reunião inicial com a empresa parceira para o entendimento e construção das bases do conceito do que deveria ser o sistema. O objetivo principal deste encontro foi realizar um levantamento aprofundado dos requisitos, buscando compreender não apenas a ideia central do projeto, mas também os objetivos de negócio e o valor que a empresa esperava obter com a implementação da solução final. Foi um momento crucial para estabelecer uma visão compartilhada e delimitar as necessidades funcionais e não funcionais que deveriam ser atendidas.

A partir das informações e diretrizes coletadas nesse primeiro contato, foi realizada a análise e a modelagem da solução proposta. Esse trabalho interno resultou na elaboração de um diagrama de caso de uso, que pode ser visualizado na Figura 7, destinado a ilustrar formalmente as interações previstas dos usuários com o sistema.

*Figura 7: Caso de uso da proposta inicial do projeto*




*Fonte: Elaborada pelo autor*

O diagrama de casos de uso do sistema está estruturado em torno de três pilares principais que representam as áreas funcionais centrais da aplicação: Patrimônio, Aluguel e Manutenção.

O pilar **Patrimônio** engloba o fluxo operacional relacionado ao gerenciamento do inventário de equipamentos. Este módulo contempla o registro de novos equipamentos no sistema, a realização de alterações em equipamentos existentes, a consulta de informações sobre os ativos cadastrados e o acompanhamento da situação operacional de cada item do inventário.

O pilar **Aluguel** representa o fluxo operacional voltado ao controle das locações de equipamentos. Este módulo abrange o cadastro e gerenciamento das unidades de negócio que atuam como clientes da empresa, bem como todos os processos relacionados à realização de aluguéis de equipamentos. As unidades de negócio interagem diretamente com este fluxo operacional ao solicitar e manter equipamentos sob regime de locação.

Por fim, o pilar **Manutenção** contempla toda a operação relacionada ao cadastro e revisão de manutenções preventivas executadas pelas unidades clientes. Este módulo é responsável por registrar os procedimentos de manutenção realizados nos equipamentos alugados, incluindo o preenchimento de *checklists*, o registro de serviços executados e o envio de comprovações fotográficas, além de permitir a validação administrativa dessas manutenções.

Os detalhes operacionais aprofundados de cada um desses módulos, incluindo as regras de negócio específicas e os fluxos de trabalho implementados, são apresentados de forma detalhada no tópico 4.1 deste documento.

Nesse momento de levantamento de requisitos também foi definida a metodologia que guiaria o projeto durante os próximos meses. Conforme já apresentado anteriormente, optou-se pela abordagem iterativa e incremental, essa que permitiria o desenvolvimento do sistema em partes bem definidas, com entregas parciais ao longo do processo. Essa escolha se mostrou fundamental para manter a empresa parceira constantemente envolvida no acompanhamento do progresso e para possibilitar ajustes conforme as necessidades fossem sendo refinadas.

Foi estabelecido também o formato de cada etapa do desenvolvimento. Ao término de cada fase, seria realizada uma reunião de apresentação simples e direta, onde as funcionalidades desenvolvidas seriam demonstradas em funcionamento dentro do próprio sistema. Essas apresentações tinham como objetivo mostrar concretamente o que havia sido implementado e orientar a equipe da empresa sobre como utilizar cada nova funcionalidade, garantindo que todos compreendessem o valor agregado em cada entrega.

Definido o escopo do projeto, o passo seguinte foi a seleção das ferramentas tecnológicas que seriam utilizadas no desenvolvimento. Logo nas primeiras reuniões com a empresa, foi estabelecido como requisito obrigatório que o sistema fosse desenvolvido utilizando a linguagem PHP, com o auxílio do *framework* Laravel. Essa exigência foi fundamentada pela familiaridade que os desenvolvedores internos da empresa já possuíam com essa tecnologia específica.

A escolha do PHP e do Laravel não apenas facilitaria a compreensão do código-fonte por parte da equipe interna da empresa, mas também simplificaria significativamente as manutenções futuras da aplicação. Como os profissionais já possuíam experiência e conhecimento prévios nessa linguagem e *framework*, a empresa garantiria que o projeto fosse sustentável e pudesse ser mantido de maneira eficiente a longo prazo, alinhando a solução tecnológica com os recursos e competências já disponíveis na organização.

Para o autor deste trabalho, essa definição representou um desafio considerável de aprendizado. Devido à falta de familiaridade com o ecossistema PHP e com o *framework* Laravel, foi necessário dedicar tempo significativo ao estudo dessas tecnologias. Como ocorreu o estudo é detalhado no tópico 3.3.

## 3.2 Ambiente de trabalho

Para todas as etapas de desenvolvimento do projeto foi utilizado um computador pessoal com as seguintes especificações:

- Processador Ryzen 5 3600
- 16GB de memória RAM 3200MHz
- SSD 2TB NVMe
- Sistema operacional Windows 11 – Professional de 64 bits

## 3.3 Estudo de tecnologias

O planejamento inicial do projeto com a empresa previa a utilização da linguagem de programação Java em conjunto com o *framework* Spring para o desenvolvimento da aplicação. Essa escolha inicial se baseava no conhecimento e experiência prévia do autor com

essas tecnologias, o que garantiria maior agilidade e segurança no processo de desenvolvimento. No entanto, logo nas primeiras reuniões com a empresa parceira, foi estabelecido como requisito obrigatório que o sistema fosse desenvolvido utilizando a linguagem PHP e o *framework* Laravel, conforme detalhado no tópico 3.1.

Essa mudança de direcionamento tecnológico representou o primeiro grande desafio do projeto, exigindo que todo o estudo necessário sobre a nova linguagem e *framework* fosse realizado em um período restrito de aproximadamente dois meses. Dentro desse prazo limitado, foi necessário não apenas aprender a sintaxe e os conceitos fundamentais do PHP, mas também dominar a arquitetura e os recursos do Laravel, de modo a estar apto a iniciar o desenvolvimento efetivo do sistema dentro do cronograma estabelecido.

Diante desse cenário, a utilização do PHP e do Laravel definiu a primeira grande etapa técnica do projeto: a necessidade de um estudo aprofundado e acelerado dessas tecnologias. Esse processo intensivo de aprendizagem contemplou diferentes recursos didáticos, como tutoriais em vídeo, tutoriais em formato de texto e, principalmente, a leitura detalhada da documentação oficial de cada ferramenta. O objetivo dessa etapa foi compreender não apenas os fundamentos da linguagem PHP, mas também as boas práticas de desenvolvimento, a estruturação de projetos com Laravel e os recursos mais modernos que o *framework* disponibiliza para o desenvolvimento de aplicações web robustas e escaláveis.

O estudo da linguagem PHP envolveu a compreensão de sua sintaxe, tipagem dinâmica, manipulação de *arrays* e *strings*, orientação a objetos, *namespaces* e tratamento de exceções. Foi necessário também entender conceitos específicos do PHP moderno, como padrões de codificação e o uso do Composer como gerenciador de dependências, ferramenta comumente encontrada em projetos PHP, inclusive aqueles que utilizam Laravel.

Paralelamente ao estudo da linguagem, foi necessário configurar um ambiente de desenvolvimento adequado para executar a aplicação. Esse processo incluiu a instalação e configuração de diversos componentes que formam a pilha tecnológica necessária para rodar aplicações Laravel. Para simplificar esse processo, foi utilizado o Laragon, uma ferramenta que automatiza a configuração de ambientes de desenvolvimento PHP no Windows. O Laragon facilitou significativamente a instalação e configuração do servidor web Nginx, e do interpretador PHP em sua versão mais recente. Essa ferramenta também oferece uma interface amigável para gerenciar múltiplos projetos simultaneamente, criar *hosts* virtuais automaticamente e alternar entre diferentes versões de PHP quando necessário.

Para auxiliar no processo de desenvolvimento e depuração do código, foi também instalado e configurado o Xdebug, uma extensão do PHP que funciona como depurador interativo. O Xdebug permite acompanhar a execução do código passo a passo, inspecionar valores de variáveis em tempo real, identificar gargalos de performance e rastrear a origem de erros com muito mais precisão do que seria possível apenas com comandos simples de impressão na tela. A configuração adequada dessa ferramenta, integrada ao editor de código, mostrou-se essencial para aumentar a produtividade durante o desenvolvimento e facilitar a resolução de problemas complexos.

O estudo do Laravel, por sua vez, envolveu compreender sua arquitetura baseada no padrão Model-View-Controller (MVC), que organiza a aplicação em três camadas distintas: os *Models*, responsáveis pela lógica de negócio e interação com o banco de dados; os *Controllers*, que processam as requisições do usuário e coordenam as respostas; e as *Views*, que definem a apresentação visual das informações. Essa separação de responsabilidades facilita a manutenção do código e permite que diferentes aspectos da aplicação sejam desenvolvidos de forma mais independente.

Um dos componentes mais importantes estudados foi o Blade, o motor de *templates* do Laravel. O Blade permite a criação de *views* dinâmicas através de uma sintaxe simples e intuitiva, suportando recursos como herança de *templates*, componentes reutilizáveis, diretivas condicionais e laços de repetição. Com o Blade, foi possível construir interfaces de usuário complexas mantendo o código organizado e evitando repetições desnecessárias. A compreensão de conceitos como *layouts* mestres, seções, includes e componentes foi fundamental para estruturar as páginas do sistema de forma eficiente.

Outro aspecto relevante do Laravel é o Artisan, sua interface de linha de comando. O Artisan oferece diversos comandos úteis que automatizam tarefas repetitivas do desenvolvimento, como a criação de *controllers*, *models* e *migrations*, a execução de migrações de banco de dados, a limpeza de *cache*, e a geração de código *boilerplate* para diversos componentes da aplicação. O estudo e a prática com o Artisan permitiram acelerar significativamente o processo de desenvolvimento, reduzindo o tempo gasto com tarefas manuais e padronizando a estrutura do código gerado.

Além disso, foi estudado o Eloquent ORM, o sistema de mapeamento objeto-relacional do Laravel. O Eloquent permite interagir com o banco de dados utilizando objetos PHP em vez de escrever consultas SQL manuais, tornando o código mais legível e menos

propenso a erros. Foi necessário entender como o Laravel aplica os conceitos de mapeamentos objeto-relacional em sua estrutura e quais as construções específicas são utilizadas pelo *framework*.

O sistema de autenticação e autorização do Laravel também foi objeto de estudo detalhado. O *framework* oferece recursos integrados para login, registro de usuários, recuperação de senha e controle de acesso baseado em permissões. Compreender esses mecanismos foi essencial para implementar o sistema de controle de acesso administrativo previsto no escopo do projeto.

Outros recursos importantes do Laravel estudados incluem o sistema de validação de formulários, que permite definir regras de validação de forma declarativa e clara; o sistema de rotas, que mapeia URLs para *controllers* e ações específicas; o *middleware*, que permite executar código antes ou depois do processamento de uma requisição; e o sistema de sessões, fundamental para manter o estado de autenticação dos usuários e armazenar dados temporários.

Com o estudo da linguagem e do *framework* devidamente encaminhados, foi realizada também uma revisão direcionada de materiais e documentações relacionadas ao sistema de banco de dados PostgreSQL. Esse aprofundamento foi necessário para familiarizar-se com as particularidades desse sistema gerenciador de banco de dados, sobretudo suas variações na arquitetura interna e nas implementações específicas da linguagem SQL. O PostgreSQL possui recursos avançados que diferem de outros bancos de dados, como criação de tipos de dados customizados e operações nativas em JSON. Dessa forma, buscou-se garantir um conhecimento sólido para que a integração entre a aplicação desenvolvida em Laravel e o banco de dados PostgreSQL ocorresse de maneira eficiente, confiável e alinhada às boas práticas do mercado.

Esse período de estudo, embora tenha demandado tempo significativo, foi fundamental para o sucesso do projeto. O conhecimento adquirido não apenas permitiu a implementação das funcionalidades propostas, mas também garantiu que o código produzido seguisse padrões de qualidade e boas práticas reconhecidas pela comunidade de desenvolvedores, facilitando futuras manutenções e expansões do sistema.

## 3.4 Ferramentas utilizadas

Para viabilizar o desenvolvimento do sistema de forma eficiente, foi necessário selecionar e configurar um conjunto de ferramentas apropriadas que pudessem atender às demandas técnicas do projeto. A escolha de cada ferramenta levou em consideração aspectos como robustez, facilidade de uso, compatibilidade com as tecnologias adotadas e alinhamento com as práticas modernas de desenvolvimento de software. A seguir, são apresentadas as principais ferramentas utilizadas ao longo do projeto, descrevendo suas características, funcionalidades e o papel que desempenharam no processo de desenvolvimento.

**PhpStorm – Ambiente de desenvolvimento integrado especializado para PHP**

O PhpStorm é um ambiente de desenvolvimento integrado (IDE) desenvolvido pela empresa JetBrains, especificamente projetado para atender às demandas de projetos baseados na linguagem PHP. Sua escolha para este trabalho se deu em razão de sua ampla gama de recursos voltados para a otimização do fluxo de trabalho e o aumento da produtividade do desenvolvedor, características essenciais considerando o prazo restrito disponível para o aprendizado das tecnologias e o desenvolvimento do sistema.

Entre suas principais funcionalidades destacam-se as sugestões automáticas de código, que auxiliam na escrita de trechos mais precisos e reduzem significativamente a ocorrência de erros de sintaxe e de digitação. Para um desenvolvedor que precisava aprender PHP e Laravel em tempo limitado, esse recurso representou um diferencial importante na escolha da ferramenta, pois facilitaria a familiarização com a sintaxe da linguagem e as convenções do *framework*. O editor também oferece análise estática de código em tempo real, identificando potenciais problemas antes mesmo da execução da aplicação, característica que contribuiria para melhorar a qualidade do código produzido mesmo durante a fase de aprendizado.

Outro recurso determinante para a escolha do PhpStorm foi a documentação integrada das APIs da linguagem PHP e de *frameworks* como o Laravel, que agiliza a consulta a métodos, classes e bibliotecas sem a necessidade de interromper o fluxo de trabalho para buscar informações em fontes externas. Essa funcionalidade foi considerada particularmente importante no contexto de aprendizado acelerado das tecnologias, pois permitiria consultas rápidas e contextualizadas durante o desenvolvimento, reduzindo o tempo necessário para compreender e aplicar os conceitos.

O PhpStorm também oferece integração nativa com sistemas de banco de dados, permitindo visualizar a estrutura das tabelas, executar consultas SQL diretamente do editor e manipular dados sem a necessidade de recorrer a ferramentas externas dedicadas. Essa integração foi considerada vantajosa para centralizar em um único ambiente tanto o código da aplicação quanto as operações de banco de dados. Além disso, possui suporte completo a sistemas de controle de versão, como o Git, integrando-se de forma eficiente ao processo de versionamento do projeto através de uma interface gráfica intuitiva para realizar *commits*, *merges*, visualizar histórico de alterações e resolver conflitos.

Dessa forma, o PhpStorm foi selecionado por reunir, em uma única ferramenta integrada, todos os recursos necessários para facilitar o aprendizado acelerado de PHP e Laravel, além de oferecer um ambiente profissional e completo para o desenvolvimento do sistema dentro do prazo estabelecido.

**Xdebug – Módulo adicional do PHP para depuração de código**

O Xdebug é uma extensão para PHP utilizada no processo de depuração e análise de código. Sua principal função é auxiliar desenvolvedores na identificação e correção de erros, permitindo a execução passo a passo do código, a inspeção detalhada de variáveis em tempo real e o acompanhamento minucioso do fluxo de execução da aplicação.

Durante o desenvolvimento deste projeto, o Xdebug mostrou-se fundamental para resolver problemas complexos que não seriam facilmente identificados apenas através da análise visual do código ou de mensagens de erro genéricas. A capacidade de pausar a execução em pontos específicos, examinar o estado completo da aplicação naquele momento e avançar linha por linha pelo código permitiu compreender com precisão o comportamento do sistema e localizar a origem de erros de forma muito mais eficiente.

Além da depuração tradicional, o Xdebug oferece recursos importantes como *stack traces* detalhados, que mostram o caminho completo percorrido pelo código até o ponto de falha, incluindo todas as chamadas de função encadeadas. Essa informação é crucial para entender o contexto em que um erro ocorreu, especialmente em aplicações que fazem uso extensivo de *frameworks* e bibliotecas externas.

Integrado ao PhpStorm, o Xdebug proporcionou uma experiência de depuração eficiente, sendo uma ferramenta essencial para melhorar a produtividade e garantir maior qualidade no desenvolvimento da aplicação.

**Laragon – Ambiente de execução local para PHP**

Aplicações desenvolvidas em PHP necessitam de um servidor web para processar as requisições HTTP, interpretar o código PHP e entregar as páginas renderizadas aos usuários. A configuração manual desse ambiente envolve a instalação e configuração individual de diversos componentes, como o servidor web (Apache ou Nginx), o interpretador PHP com suas extensões necessárias, o sistema gerenciador de banco de dados e outras ferramentas auxiliares. Esse processo pode ser trabalhoso e propenso a erros, especialmente para desenvolvedores que estão iniciando com a tecnologia.

Considerando o contexto do projeto, onde havia a necessidade de aprender PHP e Laravel em um prazo limitado de dois meses, tornou-se essencial escolher uma solução que minimizasse o tempo gasto com configurações técnicas e maximizasse o tempo dedicado ao aprendizado efetivo das tecnologias e ao desenvolvimento da aplicação. Por esse motivo, foi escolhido o Laragon, uma ferramenta que automatiza completamente a configuração de ambientes de desenvolvimento PHP no Windows.

A escolha do Laragon foi motivada por sua capacidade de realizar toda a etapa de pré-configuração de forma automatizada e integrada, instalando e configurando todos os componentes necessários com apenas alguns cliques. Essa característica foi determinante para a decisão, pois elimina a necessidade de estudar processos complexos de configuração de servidores, instalação de extensões PHP e ajustes de variáveis de ambiente, permitindo que o foco do aprendizado permanecesse nos conceitos da linguagem e do *framework*.

Outro fator que influenciou a escolha foi o conjunto de ferramentas auxiliares oferecidas pelo Laragon, como a criação automática de registros DNS locais e hosts virtuais, que permitem acessar a aplicação através de URLs amigáveis (como http://gesup.test) em vez de endereços numéricos complexos. A capacidade de alternar facilmente entre diferentes versões do PHP e gerenciar múltiplos projetos simultaneamente também foi considerada vantajosa para um ambiente de aprendizado e experimentação.

Adicionalmente, a interface gráfica simples para iniciar e parar serviços, visualizar logs de erro em tempo real e acessar ferramentas administrativas do banco de dados representou um diferencial importante na escolha, pois reduziria a complexidade operacional do ambiente de desenvolvimento. Dessa forma, o Laragon foi selecionado como a solução ideal para o ambiente de desenvolvimento deste projeto, permitindo que o autor pudesse

concentrar seus esforços no aprendizado das tecnologias e no desenvolvimento da aplicação, sem despender tempo excessivo com configurações técnicas complexas que não contribuiriam diretamente para os objetivos do trabalho.

**Git – Controle de versão**

Devido à escala e à complexidade do projeto a ser desenvolvido, tornou-se indispensável a adoção de um sistema de controle de versão para o código-fonte. Sistemas de controle de versão são fundamentais no processo de desenvolvimento de software moderno, pois permitem manter um histórico completo e detalhado de todas as alterações realizadas ao longo do tempo, incluindo informações sobre quem fez cada mudança, quando foi feita e qual foi o motivo.

Essa rastreabilidade facilita enormemente a reversão para versões anteriores em caso de introdução acidental de falhas, permitindo voltar rapidamente a um estado estável da aplicação. Além disso, o histórico de alterações serve como documentação viva do projeto, possibilitando analisar a evolução do código e entender as decisões técnicas tomadas em diferentes momentos do desenvolvimento.

Outra vantagem significativa dos sistemas de controle de versão é a possibilidade de desenvolvimento paralelo de múltiplas funcionalidades através do uso de *branchs* (ramificações). Isso evita que alterações conflitantes sejam introduzidas simultaneamente no código principal e garante maior organização no processo de trabalho, permitindo que diferentes recursos sejam desenvolvidos de forma isolada e integrados posteriormente de maneira controlada.

Nesse contexto, foi adotado o Git, uma ferramenta de controle de versão bastante utilizada no mercado de tecnologia atualmente, amplamente reconhecida por sua robustez, performance e flexibilidade em diferentes fluxos de trabalho. O Git é um sistema distribuído, o que significa que cada desenvolvedor possui uma cópia completa do histórico do projeto em sua máquina local, garantindo maior segurança e independência em relação a servidores centralizados.

No contexto deste trabalho, o Git foi utilizado não apenas para versionar o código-fonte da aplicação, mas também para organizar o desenvolvimento em *commits* lógicos e bem documentados, facilitando a compreensão das mudanças realizadas. O uso de *branchs* permitiu experimentar novas funcionalidades sem afetar o código estável, e a possibilidade de

reverter alterações problemáticas proporcionou maior confiança para realizar mudanças mais audaciosas quando necessário.

**PostgreSQL – Banco de dados relacional**

O PostgreSQL é um sistema gerenciador de banco de dados relacional (SGBD) de código aberto, com licença gratuita para uso comercial, além de ser um dos nomes mais proeminentes nos campos de desenvolvimento de aplicações de grande porte (DB-ENGINES, 2024). É uma ferramenta robusta e com um grande histórico no mercado, sendo utilizado por projetos das mais diversas escalas, desde pequenas aplicações até sistemas corporativos que gerenciam milhões de registros diariamente.

A escolha do PostgreSQL para este projeto se justifica por diversos fatores. Primeiramente, trata-se de um banco de dados extremamente versátil e repleto de recursos avançados que garantem confiabilidade e integridade dos dados. O PostgreSQL implementa completamente os conceitos de transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo que as operações no banco de dados sejam executadas de forma segura mesmo em cenários de falhas ou acessos concorrentes.

Além disso, o PostgreSQL oferece recursos que facilitam a escalabilidade da aplicação, permitindo que o sistema cresça tanto em termos de número de requisições processadas diariamente quanto no volume de dados armazenados. Funcionalidades como índices avançados, particionamento de tabelas, replicação e suporte a tipos de dados complexos (como arrays e JSON) tornam o PostgreSQL uma escolha sólida para projetos que podem precisar evoluir significativamente no futuro.

A compatibilidade nativa do Laravel com o PostgreSQL, através do Eloquent ORM e do sistema de migrations, também foi um fator importante na escolha. O framework oferece suporte completo às funcionalidades específicas do PostgreSQL, permitindo aproveitar seus recursos avançados sem precisar escrever SQL manual de forma extensiva.

## 3.5 Etapas de desenvolvimento

O processo de desenvolvimento foi organizado em quatro etapas principais, cada uma focada na construção e entrega de um conjunto específico de funcionalidades. Essa divisão

permitiu que a empresa acompanhasse o progresso do projeto de forma incremental, validando cada entrega antes do início da próxima fase.

Ao final de cada etapa, eram realizadas reuniões de apresentação das funcionalidades desenvolvidas, onde a empresa avaliava o que havia sido implementado e fornecia *feedback* sobre possíveis ajustes e melhorias. Essas observações eram então incorporadas como refinamentos para a próxima entrega, garantindo que o sistema evoluísse de acordo com as necessidades reais da operação.

### 3.5.1 Modelagem e Estruturação do Banco de Dados

Uma das primeiras atividades técnicas realizadas foi a elaboração da estrutura do banco de dados necessária para acomodar todas as entidades e fluxos de informações que seriam utilizados pelo sistema. Essa estrutura foi planejada de forma a suportar os principais módulos identificados na fase de análise, incluindo inventário de equipamentos, unidades de negócio, usuários, aluguéis e manutenções.

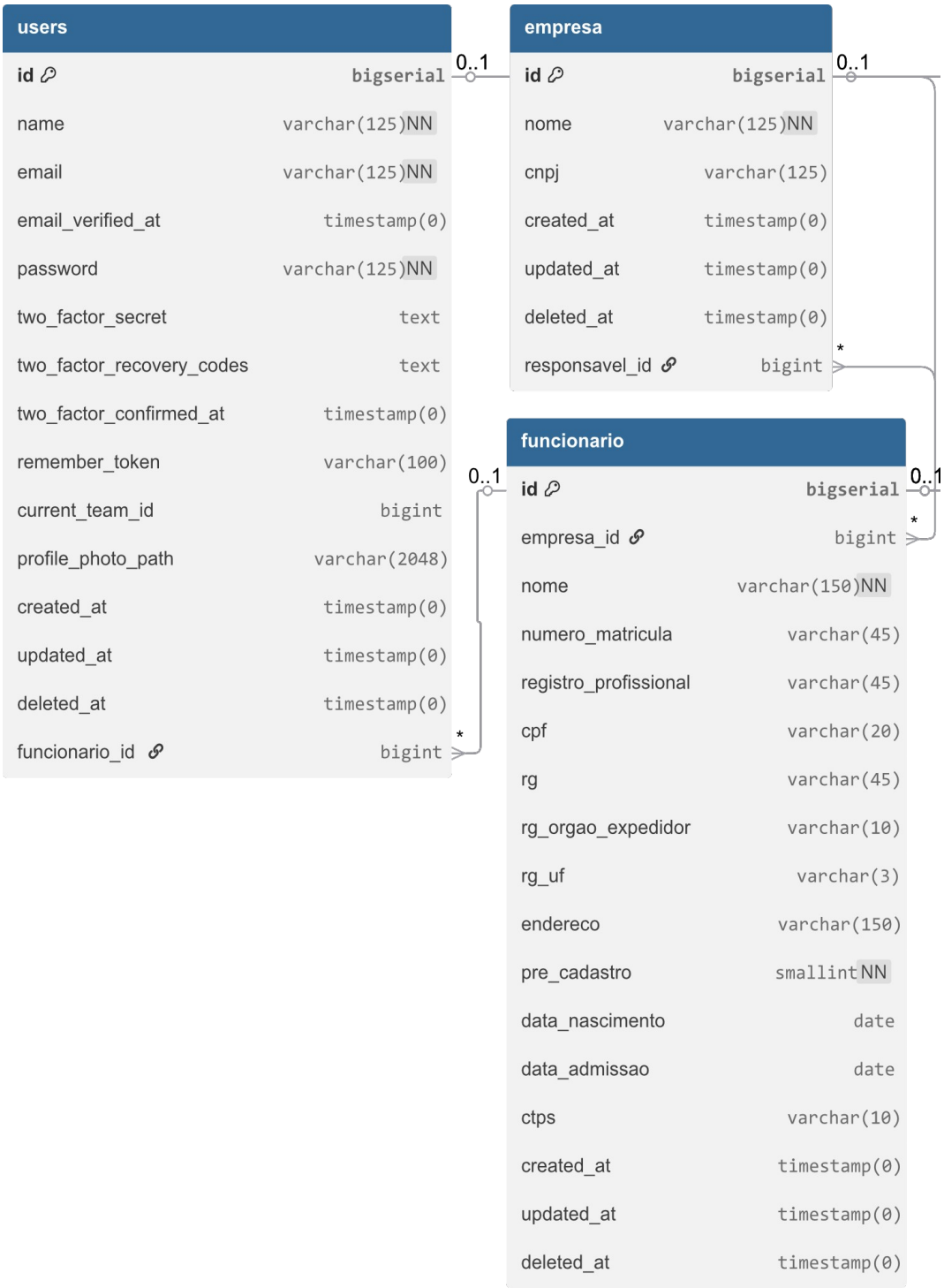
No entanto, é importante destacar que, embora uma versão inicial do modelo de dados tenha sido estruturada logo nas primeiras etapas da concepção do projeto, o banco de dados foi desenvolvido de forma concorrente às etapas de entrega. Essa abordagem teve como objetivo reduzir o trabalho em cada entrega apenas aos pontos correspondentes às funcionalidades esperadas naquela fase específica, evitando o desenvolvimento prematuro de estruturas que ainda não seriam utilizadas.

Além disso, essa estratégia se mostrou fundamental para acomodar as diversas alterações e incrementos de escopo que ocorreram durante todo o processo de desenvolvimento. Conforme os requisitos foram sendo refinados e novas necessidades identificadas, o banco de dados sofreu múltiplas modificações: novos campos precisaram ser criados para armazenar informações adicionais, campos que se mostraram desnecessários foram removidos para simplificar a estrutura, e relacionamentos entre tabelas foram reestruturados para refletir com mais precisão as regras de negócio da empresa. A versão final da estrutura do banco construída pode ser visualizada nas figuras 8, 9, 10 e 11.

O uso do sistema de *migrations* do Laravel foi essencial nesse processo, permitindo versionar todas as alterações na estrutura do banco de dados de forma controlada e documentada. Cada mudança significativa no modelo de dados era registrada em uma nova

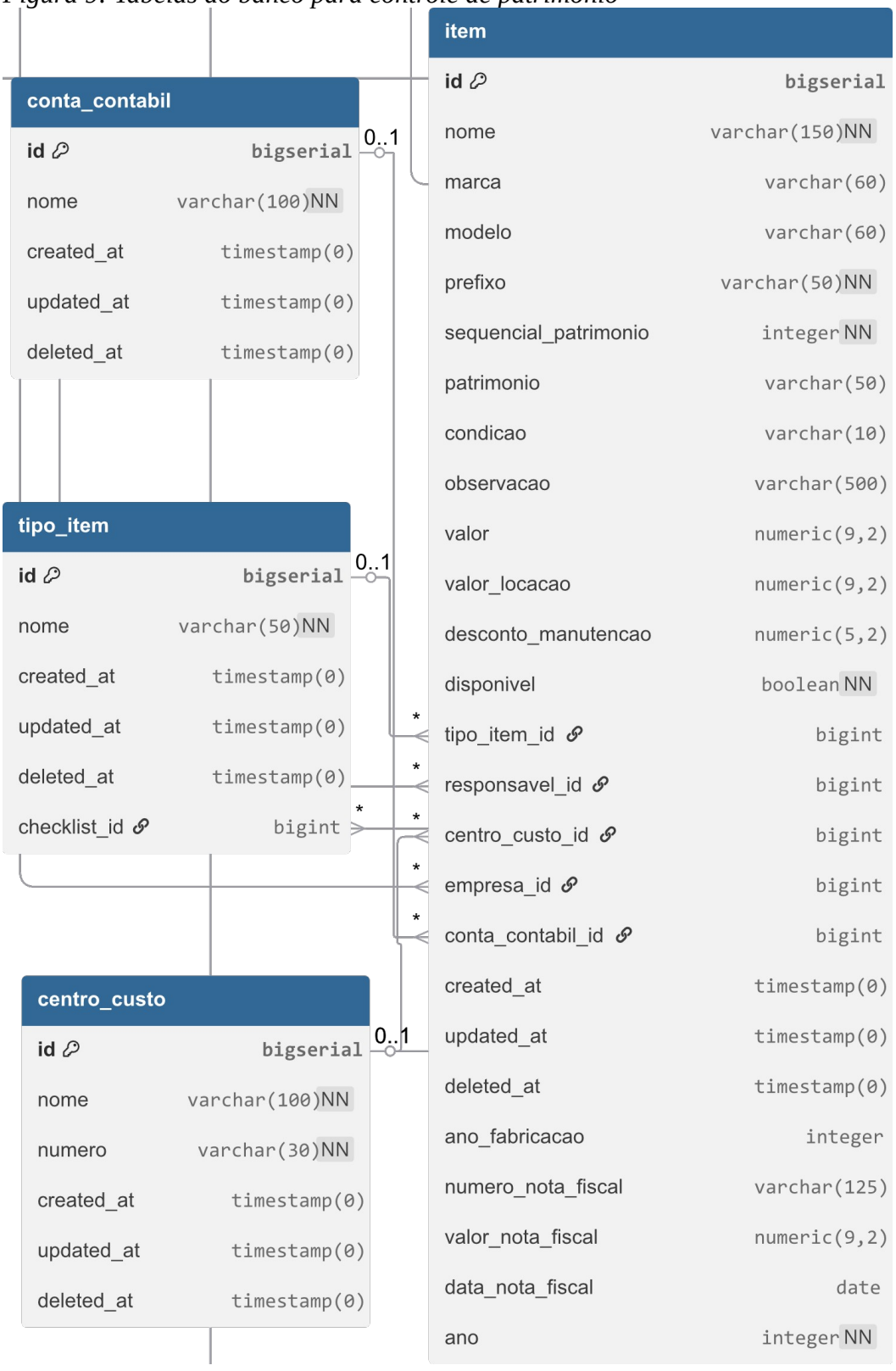
*migration*, facilitando o rastreamento da evolução do esquema e garantindo que o ambiente de desenvolvimento permanecesse sincronizado com as necessidades do projeto.

*Figura 8: Tabelas do banco para controle de acesso*



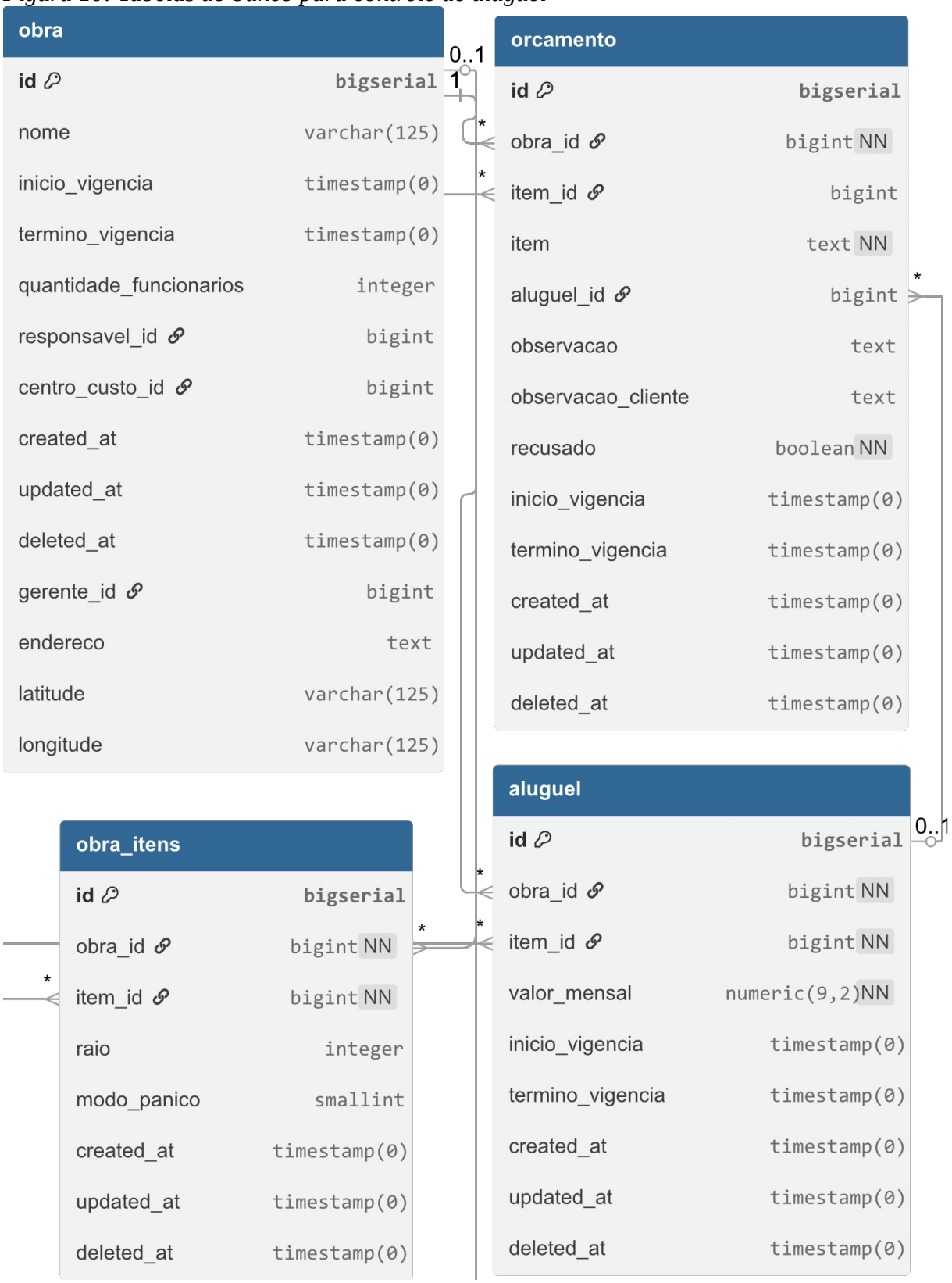

*Fonte: Elaborada pelo autor*

*Figura 9: Tabelas do banco para controle de patrimônio*



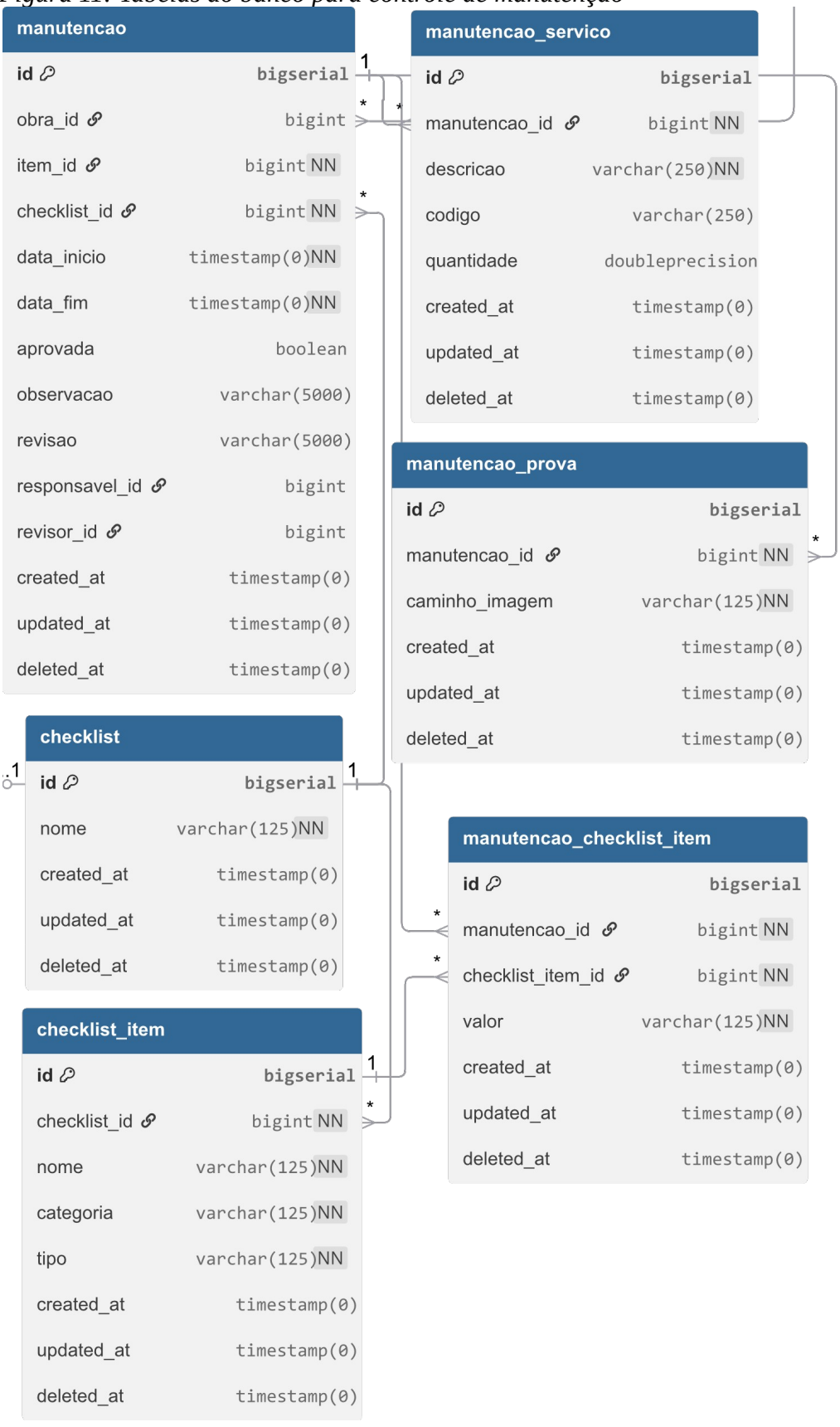

*Fonte: Elaborada pelo autor*

*Figura 10: Tabelas do banco para controle de aluguel*




*Fonte: Elaborada pelo autor*

*Figura 11: Tabelas do banco para controle de manutenção*




*Fonte: Elaborada pelo autor*

O banco de dados foi estruturado seguindo a divisão conceitual dos próprios módulos do sistema, organizando as informações de forma lógica e refletindo os principais domínios de negócio da aplicação. As entidades foram agrupadas em cinco áreas principais: Controle de Acesso e Usuários, Patrimônio, Unidades de Negócio, Aluguéis e Manutenções. Cada uma dessas áreas possui suas tabelas específicas e relacionamentos.

**Módulo de Controle de Acesso e Usuários**

Este módulo é responsável por gerenciar os usuários do sistema e controlar o acesso às funcionalidades através de um sistema de permissões baseado em papéis (roles). A tabela principal é a **“users”**, que armazena as informações cadastrais dos usuários, incluindo nome, e-mail, senha criptografada e dados de autenticação.

O sistema de permissões é implementado através dos próprios mecanismos de autenticação e controle de acessos do Laravel e, dessa forma, não estão incluídos no diagrama construído. Apesar de esses sistemas possuírem suas próprias tabelas no banco de dados, essas são automaticamente geradas e gerenciadas pelas bibliotecas relacionadas e, portanto, não foi necessária a modelagem desses armazenamentos. Esse sistema permite o controle de acesso através de cargos (como Administrador, Gerente de Unidade, Operador) e as permissões específicas que podem ser atribuídas a cada cargo (como visualizar inventário, editar aluguéis, excluir manutenções).

**Módulo de Inventário**

O módulo de Inventário diz respeito aos equipamentos cadastrados no sistema e mantém informações essenciais sobre cada ativo da empresa. A tabela principal é a **“item”** que armazena os dados propriamente ditos de cada equipamento, incluindo número identificador (patrimônio), nome descritivo, data de aquisição, valor de compra, custo de aluguel mensal, e campos adicionais com informações sobre fornecedor, nota fiscal, estado de conservação e situação cadastral (ativo, inativo, em manutenção, baixado).

Relacionada à tabela **“item”** está a tabela **“tipo_item”** que categoriza os equipamentos de acordo com seu porte e características. Essa classificação agrupa os itens em categorias como equipamentos de pequeno porte, médio porte, grande porte e outros, facilitando a organização e a aplicação de regras de negócio específicas para cada tipo.

**Módulo de Obras/Unidades de Negócio**

O módulo de Unidades de Negócio (inicialmente denominado “Obras”) gerencia as informações das unidades clientes que mantêm vínculo com a empresa, realizando aluguéis de equipamentos e registrando manutenções. A tabela principal deste módulo é a **“obra”**, que armazena os dados cadastrais completos de cada unidade, incluindo nome, endereço, data de início e término de vigência do contrato, e status atual.

Esta tabela estabelece diversos relacionamentos importantes com outras entidades do sistema. O relacionamento com a tabela **“users”** permite vincular um usuário específico como gerente responsável pela unidade, garantindo que cada projeto tenha um responsável claramente definido. Já o relacionamento com a tabela **centro_custo** possibilita o controle financeiro adequado, associando cada unidade a um centro de custo específico utilizado pela contabilidade da empresa para organização dos gastos e receitas.

Adicionalmente, a tabela **“obra”** se relaciona com os módulos de Aluguéis e Manutenções, permitindo rastrear todos os equipamentos alugados por cada unidade e todas as manutenções realizadas em seus equipamentos, formando assim um histórico completo das operações vinculadas a cada projeto.

**Módulo de Aluguéis**

O módulo de Aluguéis gerencia o processo de locação de equipamentos do inventário para as unidades de negócio registradas no sistema. A tabela central é a **“aluguel”** que registra cada operação de locação realizada, armazenando informações como qual equipamento foi alugado, para qual unidade de negócio, qual o valor da mensalidade cobrada, e qual o período de vigência do aluguel (datas de início e término).

Este módulo implementa relacionamentos com as tabelas **“item”** e **“obra”** estabelecendo as conexões necessárias entre o equipamento alugado e a unidade locatária. Além disso, a tabela de aluguéis mantém campos para controle de valores aplicados, que podem ser concedidos com base em diversos critérios, como duração do contrato ou realização de manutenções preventivas, conforme as regras de negócio estabelecidas.

**Módulo de Manutenções**

O módulo de Manutenções é um dos mais complexos do sistema, permitindo que as unidades de negócio registrem a realização de manutenções preventivas nos equipamentos

que possuem alugados. Este módulo envolve múltiplas tabelas que trabalham em conjunto para armazenar todas as informações relevantes sobre cada procedimento de manutenção realizado.

A tabela principal é a **“manutencao”** que registra as informações gerais de cada manutenção, incluindo qual equipamento foi mantido, qual unidade realizou a manutenção, as datas de início e término do procedimento, e observações gerais sobre o trabalho realizado.

Complementando as informações da manutenção, a tabela **“manutencao_servico”** armazena o registro detalhado de cada serviço individual executado durante a manutenção, como troca de peças, ajustes, limpezas e reparos específicos. Já a tabela **“manutencao_prova”** armazena as referências às imagens comprovatórias enviadas pelos usuários, que servem como evidência visual da realização adequada dos procedimentos.

Um aspecto importante deste módulo é o sistema de *checklist*, implementado através das tabelas **“checklist” “checklist_item”** e **“manutencao_checklist_item”** Esse sistema será detalhado posteriormente no tópico 4.1, mas em resumo, permite definir listas de verificação específicas para cada tipo de equipamento, garantindo que todos os procedimentos obrigatórios sejam realizados e documentados adequadamente durante cada manutenção.

O módulo de manutenções também se integra ao sistema de cálculo de descontos, pois o registro adequado de manutenções preventivas, seguindo regras específicas que serão apresentadas no tópico 4.1, pode gerar bônus para a unidade de negócio no formato de descontos nas mensalidades de aluguel quando determinadas metas são atingidas. Essa mecânica incentiva a realização regular de manutenções preventivas, aumentando a vida útil dos equipamentos e reduzindo custos com manutenções corretivas emergenciais.

### 3.5.2 Primeira entrega: A Base Operacional

A primeira etapa teve como objetivo principal estabelecer o alicerce funcional do sistema, criando os módulos essenciais para o gerenciamento dos ativos e o registro das operações fundamentais. Esta fase foi crucial para definir a estrutura sobre a qual todas as funcionalidades posteriores seriam construídas.

As seguintes funcionalidades foram implementadas nesta etapa:

- **Controle de inventário de equipamentos:** Foi desenvolvido o módulo central do sistema, permitindo a gestão completa do ciclo de vida dos equipamentos, incluindo cadastros detalhados, edições de informações (como status ou localização) e remoção de ativos.
- **Cadastro básico de unidades de negócio:** Para que os equipamentos pudessem ser alocados, foi implementado um cadastro de unidades de negócio, permitindo associar cada equipamento a um projeto ou local de uso específico.
- **Controle básico de usuários:** Um sistema inicial de gerenciamento de usuários foi adicionado, focado na autenticação e na diferenciação entre os primeiros administradores do sistema.
- **Medição de custos de aluguel:** Foi desenvolvida a funcionalidade para calcular e registrar os custos referentes ao aluguel dos equipamentos alocados nas obras, estabelecendo a base para o controle financeiro.
- **Importação de planilhas:** Para agilizar a migração dos dados existentes, foi criada uma ferramenta que permitia a importação em massa de equipamentos a partir de planilhas, populando o inventário rapidamente.

### 3.5.3 Segunda entrega: Manutenção e Inteligência de Negócio

Com a estrutura base do inventário já estabelecida e funcionando, a segunda etapa concentrou-se em agregar valor ao controle desses ativos, implementando funcionalidades voltadas para a manutenção preventiva e corretiva dos equipamentos, além de adicionar automação de regras de negócio que otimizassem a operação.

As principais implementações desta fase foram:

- **Controle de manutenções**: Foi implementado um módulo completo para registrar o histórico de manutenções de cada equipamento (cadastros, edições e remoções), permitindo rastrear custos de reparo e identificar problemas recorrentes.
- **Cálculo automatizado de descontos**: O sistema passou a aplicar regras de negócio automaticamente, como o cálculo de descontos progressivos em aluguéis de longa duração.
- **Lembretes e avisos de manutenção preventiva**: Para aumentar a vida útil dos equipamentos, foi criado um sistema proativo de notificações. Ele alertava os gestores

sobre a proximidade de manutenções preventivas, inicialmente com investigações para envio por e-mail e outros meios de aviso.

### 3.5.4 Terceira entrega: Refinamento de Acessos e Comunicação

A terceira fase do desenvolvimento teve como foco principal o amadurecimento do sistema, especialmente no que diz respeito à segurança e ao controle de acesso. Além disso, esta etapa trouxe melhorias significativas na comunicação automática do sistema com seus usuários.

As funcionalidades implementadas nesta etapa incluem:

- **Controle de acesso administrativo**: O controle básico de usuários foi expandido para um sistema de "Controle de Acesso Baseado em Papéis". Isso permitiu a criação de cargos administrativos com permissões granulares (por exemplo, acesso apenas de leitura, acesso restrito a menus financeiros ou permissão apenas para ações específicas, como exclusão).
- **Perfil de gerente de obra**: Foi criado um tipo de usuário não-administrativo, destinado aos gerentes de obras. Este perfil possuía acesso limitado, focado apenas em suas responsabilidades.
- **Telas dedicadas ao gerente**: Para atender a esse novo perfil, foram desenvolvidas telas específicas que possibilitavam ao gerente consultar, de forma rápida, suas obras ativas e os aluguéis de equipamentos vinculados a elas.
- **Expansão do envio de e-mails**: O sistema de notificações da etapa 2 foi ampliado para disparar e-mails de lembrete sobre diversos eventos, como prazos de manutenção, vencimento de faturas de aluguel e outros alertas gerenciais.

### 3.5.5 Quarta entrega: Reformulação e Ajustes Estruturais

A quarta e última etapa do projeto teve um desenvolvimento particularmente desafiador. Inicialmente, esta fase havia sido planejada para contemplar a digitalização do fluxo de compras da empresa, incluindo o cadastro de cotações, o controle de compras e o cadastro automático de equipamentos através da leitura de Notas Fiscais. No entanto, pouco

antes do início desta etapa, a empresa parceira solicitou uma reformulação completa no escopo.

Apesar de os fluxos operacionais do sistema já terem sido previamente estabelecidos e validados com a empresa, a mesma manifestou insatisfação com alguns aspectos do resultado entregue nas etapas anteriores. Em resposta a isso, foi requisitada uma alteração substancial no funcionamento de diversas funcionalidades que já haviam sido aprovadas anteriormente. Essas mudanças de escopo impactaram significativamente o cronograma, fazendo com que a quarta etapa se estendesse por vários meses além do prazo inicialmente previsto.

Após as readequações necessárias, foram implementadas as seguintes funcionalidades:

- **Cadastro em lote de equipamentos via planilha:** Foi desenvolvida uma funcionalidade aprimorada de importação que permite o cadastro em massa de equipamentos através de planilhas, agora com a capacidade de gerar automaticamente números de patrimônio para cada item cadastrado, garantindo rastreabilidade e controle individual dos ativos.
- **Depreciação anual de equipamentos:** O sistema passou a calcular e registrar a depreciação anual dos equipamentos, funcionalidade essencial para o controle contábil e financeiro dos ativos da empresa.
- **Alterações de nomenclatura:** Foram realizadas mudanças significativas na nomenclatura utilizada no sistema para melhor refletir os termos utilizados pela empresa. O módulo “Inventário” passou a ser chamado de “Banco de Dados”, e “Obras” foi renomeado para “Unidades de Negócio”, tornando a interface mais alinhada com a linguagem operacional da empresa contratante.
- **Reordenação de menus:** A estrutura de navegação do sistema foi reorganizada para tornar o acesso às funcionalidades mais intuitivo e adequado ao fluxo de trabalho dos usuários.
- **Controle automático de centro de custo:** Foi implementado um sistema automatizado para gerenciar e atribuir centros de custo aos equipamentos e operações, facilitando o controle financeiro e a elaboração de relatórios contábeis.
- **Controle automático de valores de mensalidade e desconto:** O sistema passou a calcular automaticamente os valores de mensalidade dos aluguéis, aplicando os descontos pertinentes de acordo com as regras de negócio estabelecidas, sem necessidade de intervenção manual.

- **Modificação da lógica de medição de aluguéis:** A forma como os valores de aluguel eram contabilizados foi reestruturada. O sistema passou a calcular os valores trimestralmente, em vez de mensalmente, atendendo a uma necessidade específica do processo financeiro da empresa contratante.
- **Campos extras em múltiplos cadastros:** Diversos cadastros existentes foram incrementados com campos adicionais solicitados pela empresa, permitindo o armazenamento de informações mais detalhadas e específicas sobre os equipamentos e operações.

Esta última etapa, apesar dos desafios e do tempo adicional necessário, foi fundamental para garantir que o sistema atendesse completamente às expectativas e necessidades operacionais da empresa parceira.

# 4 Desenvolvimento do projeto

O desenvolvimento deste projeto representou uma excelente oportunidade de aprendizado prático sobre o processo completo de criação de um sistema de *software*. Foi possível vivenciar na prática as etapas fundamentais como a modelagem do sistema, a condução de reuniões com a empresa parceira para levantamento de requisitos, o planejamento de funcionalidades e a implementação das soluções propostas. Essa experiência permitiu compreender melhor como funciona o ciclo de desenvolvimento de software em um contexto real, com todas as suas particularidades e desafios.

O sistema desenvolvido foi batizado internamente de “GESUP”, sigla para Gestor de Suprimentos, nome dado pela própria empresa parceira. Ele está organizado em cinco módulos principais: autenticação e autorização, inventário, unidades de negócio, aluguéis e manutenções. O objetivo principal desta versão é resolver os pontos mais críticos da operação da empresa, atacando demandas específicas como o controle das manutenções preventivas realizadas nos equipamentos, o gerenciamento dos aluguéis e o cálculo automatizado de valores, além dos controles cadastrais básicos. Dessa forma, o sistema busca centralizar e facilitar a gestão das atividades operacionais que antes eram realizadas de maneira mais fragmentada.

No momento da apresentação e defesa do presente projeto de conclusão de curso, o protótipo desenvolvido se encontra na sua fase de testes, com a empresa parceira utilizado e avaliando a solução construída. Devido a isso, esse texto apresenta todo o processo de desenvolvimento, mas não apresenta os testes e resultados finais obtidos após a implementação e avaliação do resultado nas operações da empresa. Esse ponto é discutido mais no tópico 5.1, onde são detalhados os trabalhos futuros a serem realizados.

## 4.1 Regras de negócio

O sistema GESUP foi desenvolvido para atender a um conjunto específico de regras de negócio que refletem as necessidades operacionais e estratégicas da empresa parceira. Essas regras definem como os diferentes módulos do sistema interagem entre si e como os processos de trabalho devem ser executados.

### 4.1.1 Gestão de Inventário e Unidades de Negócio

No centro da operação do sistema está o controle do inventário de equipamentos pertencentes à empresa. Cada equipamento cadastrado no inventário possui informações detalhadas como número de patrimônio, descrição, categoria, data de aquisição, valor de compra e custo mensal de aluguel. Esses equipamentos representam os ativos que a empresa disponibiliza para locação.

Paralelamente ao inventário, o sistema gerencia uma lista de unidades de negócio (obras), que representam os clientes da empresa. Cada unidade de negócio possui seu cadastro completo, incluindo dados contratuais como período de vigência, gerente responsável e centro de custo associado para fins de controle financeiro.

### 4.1.2 Processo de Aluguel de Equipamentos

O sistema permite que cada unidade de negócio realize o aluguel de um ou mais equipamentos do inventário por um período determinado. Ao cadastrar um aluguel, são definidas informações essenciais como o equipamento locado, a unidade de negócio locatária, o valor da mensalidade a ser cobrada e o prazo de vigência do contrato (datas de início e término).

O aluguel é realizado através de um processo de orçamento, onde a unidade de negócio contratante requisita um equipamento específico à empresa e um período de aluguel. Esse pedido é então revisado pela empresa que fornece um valor ou nega a solicitação de acordo com a demanda. Por fim, a unidade de negócio cliente pode então aceitar os valores ou encerrar o orçamento.

Um mesmo equipamento só pode estar alugado para uma única unidade de negócio por vez, garantindo o controle adequado sobre a localização e o uso de cada ativo. O sistema mantém o histórico completo de todos os aluguéis realizados, permitindo rastrear o ciclo de vida de cada equipamento.

### 4.1.3 Sistema de Manutenções Preventivas e Bonificações

Um dos aspectos mais importantes das regras de negócio do sistema é o incentivo à realização de manutenções preventivas. As unidades de negócio podem (e são encorajadas a) realizar manutenções preventivas nos equipamentos que alugam, e em troca, podem receber bonificações na forma de descontos nas mensalidades de aluguel caso atinjam determinadas metas estabelecidas pela empresa.

Especificamente, se uma unidade de negócio mantiver uma determinada porcentagem de seus equipamentos alugados com manutenções preventivas realizadas e devidamente aprovadas, ela se torna elegível para receber descontos progressivos em suas mensalidades. Essa mecânica tem um duplo objetivo: incentiva os clientes a cuidarem adequadamente dos equipamentos, aumentando sua vida útil e reduzindo custos com manutenções corretivas emergenciais, e ao mesmo tempo fortalece o relacionamento com os clientes através de benefícios financeiros concretos.

O processo de cadastro de uma manutenção no sistema segue um fluxo estruturado que garante a documentação completa e adequada de todos os procedimentos realizados. Esse processo envolve diversos elementos que trabalham em conjunto para assegurar que as manutenções sejam executadas conforme os padrões estabelecidos pela empresa.

Antes que qualquer manutenção possa ser cadastrada no sistema, alguns pré-requisitos devem estar satisfeitos. Primeiramente, o equipamento a ser mantido já deve estar cadastrado no inventário do sistema. Além disso, deve existir um *checklist* previamente configurado e associado ao tipo de equipamento em questão.

Além do *checklist*, o usuário deve informar os serviços específicos realizados durante a manutenção. Esses serviços representam as ações concretas executadas, como “revisão completa das peças móveis”, “instalação de novos filtros de ar”, “substituição de componentes desgastados”, “calibragem de sistemas hidráulicos”, entre outros. Cada serviço pode incluir descrição detalhada, peças utilizadas e tempo de execução.

### 4.1.4 Sistema de *Checklist*

O *checklist* é um componente fundamental do processo de manutenção. Trata-se essencialmente de uma lista estruturada de tópicos e itens que indicam quais ações devem ser

realizadas em um equipamento durante sua manutenção preventiva. Cada tipo de equipamento possui seu próprio *checklist* específico, adaptado às suas características e necessidades particulares de manutenção.

Os itens do *checklist* podem ter diferentes formatos de resposta. Alguns são do tipo "sim/não", indicando simplesmente se determinada ação foi ou não realizada (por exemplo, "foi realizada a troca de óleo?"). Outros podem aceitar valores numéricos, como quantidades de peças substituídas, medições realizadas ou horas de operação. Há também itens que permitem respostas em formato de texto livre, para observações e detalhes adicionais sobre procedimentos específicos.

Além de registrar o que foi feito, o *checklist* também identifica quais itens são obrigatórios e quais são opcionais. Itens obrigatórios representam procedimentos essenciais que devem necessariamente ser realizados em toda manutenção preventiva daquele tipo de equipamento (como troca de óleo, revisão de correias, verificação de sistemas de segurança, etc.). A não realização de um item obrigatório pode impedir a aprovação da manutenção pela equipe administrativa.

### 4.1.5 Processo de Revisão e Aprovação Administrativa

Após o cadastro completo de uma manutenção pela unidade de negócio, a mesma entra em um processo de revisão pela equipe administrativa da empresa. Nesta etapa, os administradores verificam se todos os itens obrigatórios do *checklist* foram devidamente preenchidos, se os serviços declarados condizem com as necessidades do equipamento, e se as fotografias comprobatórias demonstram adequadamente a execução dos trabalhos.

A equipe administrativa possui autoridade para aprovar ou reprovar cada manutenção cadastrada. Uma manutenção aprovada é contabilizada para fins de cálculo das metas de bonificação da unidade de negócio. Já uma manutenção reprovada pode retornar para a unidade com observações sobre o que precisa ser corrigido ou complementado.

Quando uma unidade de negócio atinge a porcentagem mínima estabelecida de equipamentos com manutenções aprovadas, o sistema automaticamente aplica o desconto correspondente nas mensalidades de aluguel daquela unidade. Esse desconto permanece ativo enquanto a meta for mantida, incentivando a continuidade das boas práticas de manutenção preventiva.

### 4.1.6 Cálculos Financeiros

Um aspecto importante da gestão financeira é o cálculo da depreciação dos equipamentos ao longo do tempo. O sistema aplica automaticamente a depreciação anual sobre o valor dos ativos, reduzindo gradualmente o custo de aluguel de cada equipamento de acordo com sua idade. Essa regra reconhece que equipamentos mais antigos, mesmo quando bem mantidos, têm valor de mercado reduzido e portanto devem ser alugados por valores proporcionalmente menores. A depreciação é calculada com base na data de aquisição do equipamento e segue critérios preestabelecidos pela empresa, sendo aplicada de forma consistente em todo o inventário.

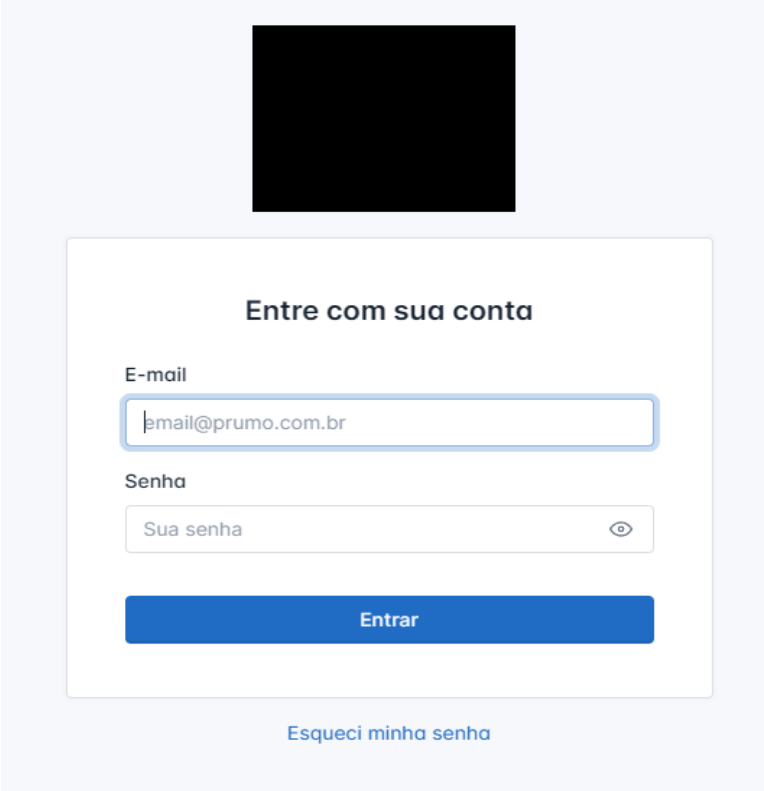
## 4.2 O sistema

Neste capítulo é apresentada a aplicação construída como resultado do presente projeto. O objetivo é fornecer uma visão completa e estruturada da solução entregue, demonstrando como os conceitos estudados e as tecnologias selecionadas foram aplicados na prática para atender às necessidades da empresa.

### 4.2.1 Tela de autenticação

A tela de autenticação (Figura 12) é enxuta, possuindo apenas os campos de e-mail e senha para acesso do usuário. Nela também é incluída a funcionalidade de recuperação de senha, essa que pode ser acessada através do link “Esqueci minha senha”.

*Figura 12: Tela de autenticação do sistema*




*Fonte: Elaborada pelo autor*

### 4.2.2 Tela principal

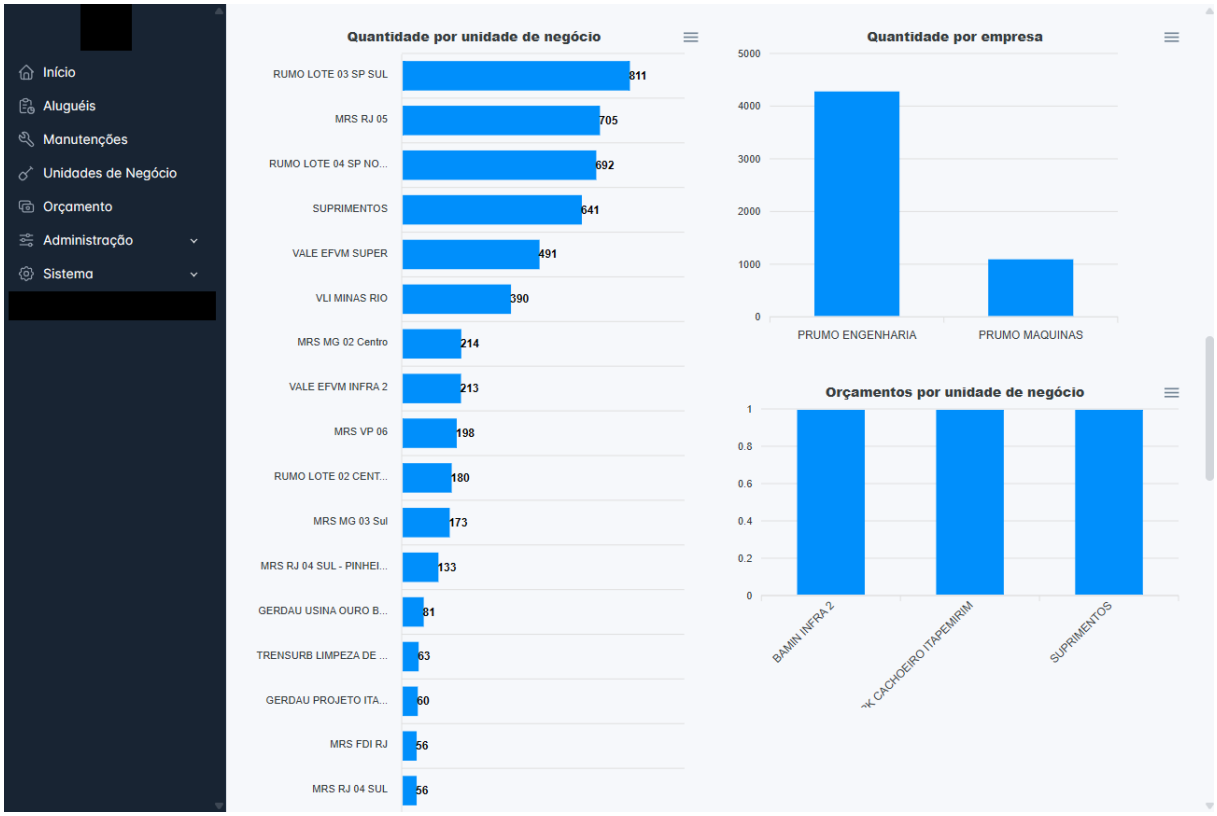
Após o acesso ao sistema essa é a primeira tela que o usuário visualiza. Nela é exibido um panorama do sistema permitindo ao usuário rapidamente verificar os valores transitados pelo sistema. Nessa tela são incluídos diversos gráficos, como por exemplo: valor do aluguel por unidade de negócio, valor do aluguel por tipo e quantidade de itens por unidade de negócio. Nas figuras 13 e 14 é possível visualizar uma base de exemplo demonstrando a funcionalidade da tela.

*Figura 13: Tela principal*


*Fonte: Elaborada pelo autor*

É uma tela flexível já que os gráficos exibidos são construídos de forma modular, isso é, podem ser facilmente realocados e modificados de acordo com a necessidade da empresa. Isso permitirá que a tela evolua naturalmente com o crescimento do sistema e seja adaptada às futuras necessidades do sistema, sempre se adequando àquilo que é esperado pelo usuário.

*Figura 14: Continuação da tela principal*




*Fonte: Elaborada pelo autor*

### 4.2.3 Tela de aluguéis

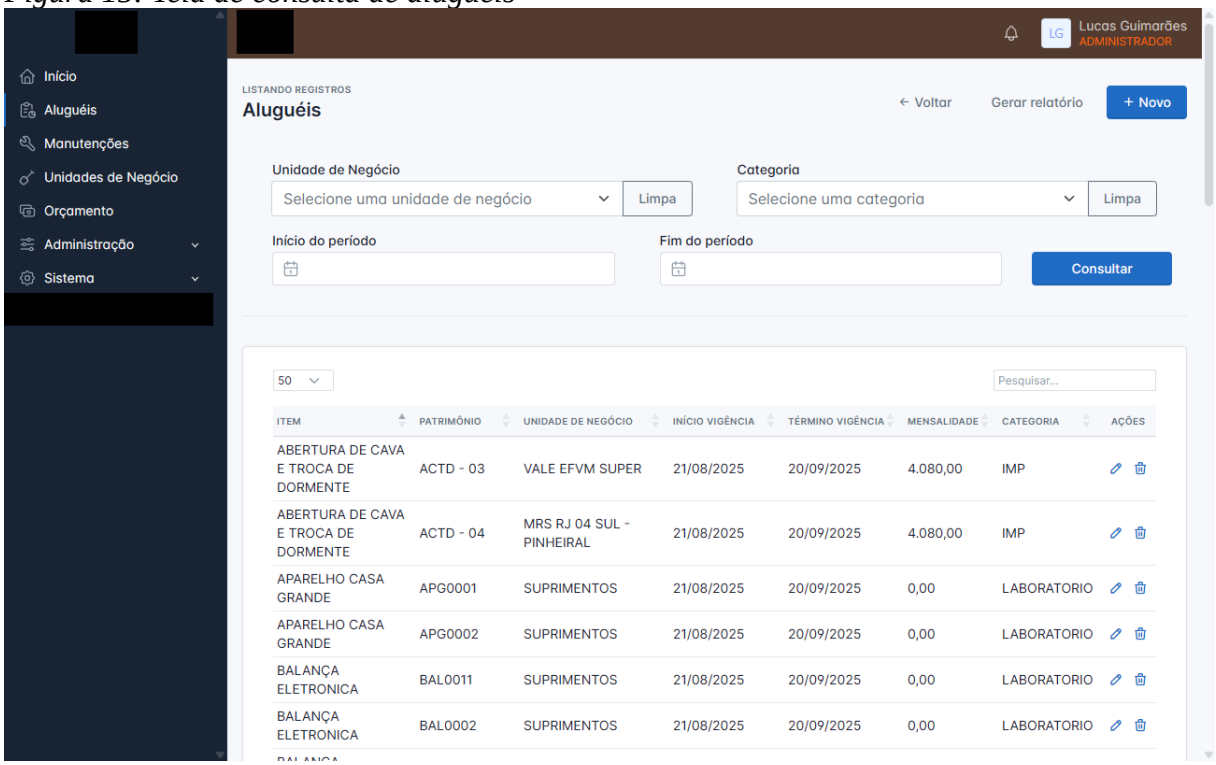
Ao acessar o menu Aluguéis, a tela principal exibe uma listagem completa de todos os registros gerenciados pelo sistema, conforme mostrado na Figura 15. Essa listagem inclui tanto os aluguéis ativos quanto os já finalizados, permitindo que o usuário tenha uma visão geral do histórico.

Nessa tela, é possível verificar informações importantes sobre cada equipamento, como sua situação atual no sistema, se está alugado no momento e qual o prazo previsto para devolução. Essas informações ajudam no acompanhamento e controle dos equipamentos disponíveis.

A tela também oferece a opção de cadastrar um novo aluguel, representada por um botão que direciona o usuário para a página de cadastro apresentada na Figura 16. No entanto, essa função é restrita aos usuários com perfil administrativo. Isso acontece porque o cadastro

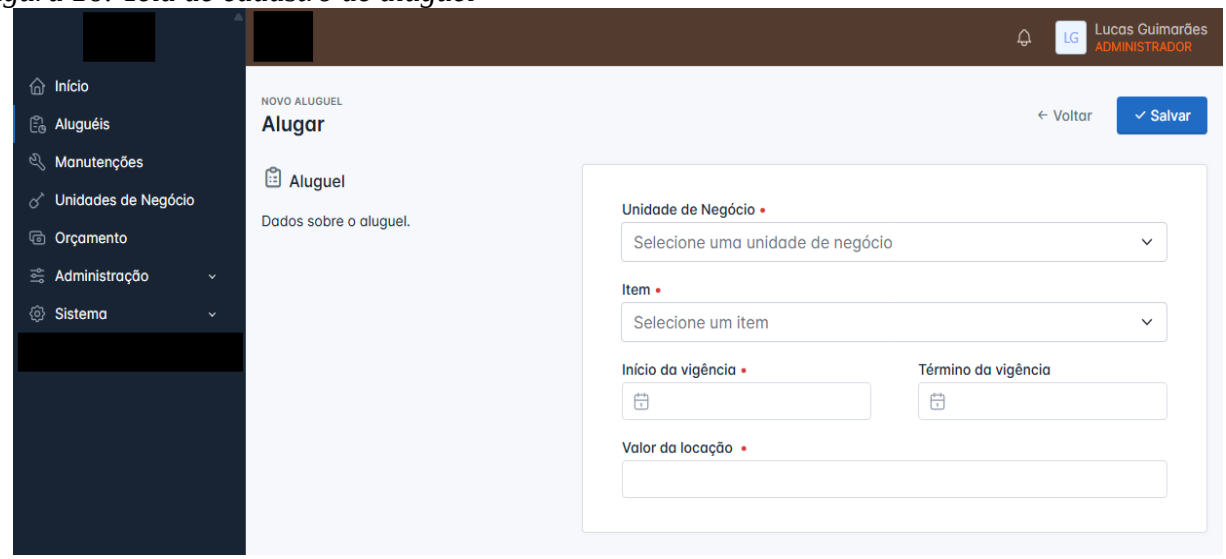
realizado por essa tela é direto no sistema, sem passar pelas verificações de disponibilidade do equipamento ou pelos cálculos de orçamento que normalmente são aplicados. Essa função é disponibilizada como uma ferramenta auxiliar administrativa, permitindo o cadastro de aluguéis anteriores ao estabelecimento do sistema, ou que foram realizados por fora do mesmo.

*Figura 15: Tela de consulta de aluguéis*




*Fonte: Elaborada pelo autor*

*Figura 16: Tela de cadastro de aluguel*

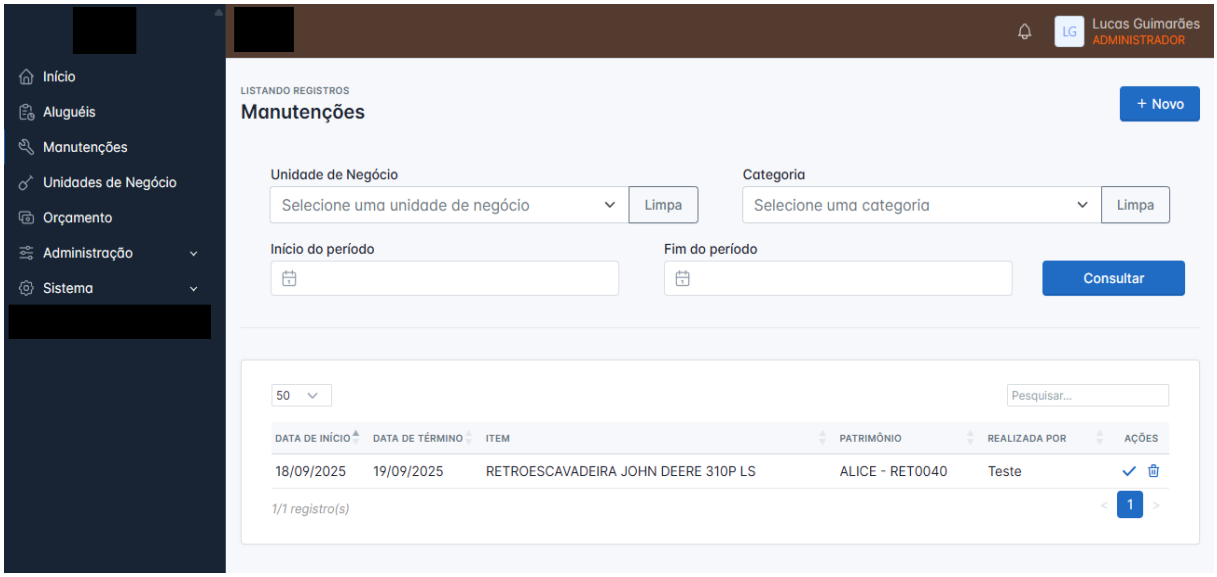



*Fonte: Elaborada pelo autor*

### 4.2.4 Tela de manutenções

A tela de manutenções (Figura 17) apresenta uma listagem completa de todos os registros de manutenções realizadas no sistema. Nessa interface, é possível consultar informações essenciais como o equipamento que recebeu a manutenção, a data em que o procedimento foi executado e qual unidade de negócio foi responsável pela realização. A tela também disponibiliza funcionalidades de filtragem e busca para facilitar a localização de manutenções específicas, além de fornecer acesso direto ao formulário de cadastro de novas manutenções através de um botão dedicado.

*Figura 17: Tela de listagem de manutenções*

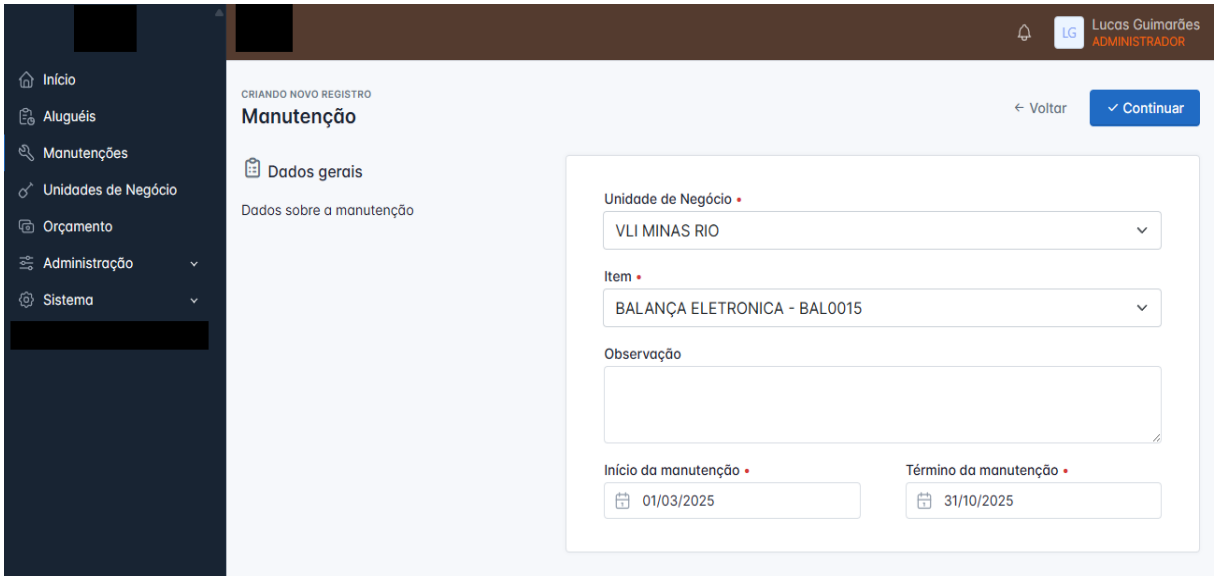



*Fonte: Elaborada pelo autor*

O processo de cadastro de uma nova manutenção segue três etapas simples:

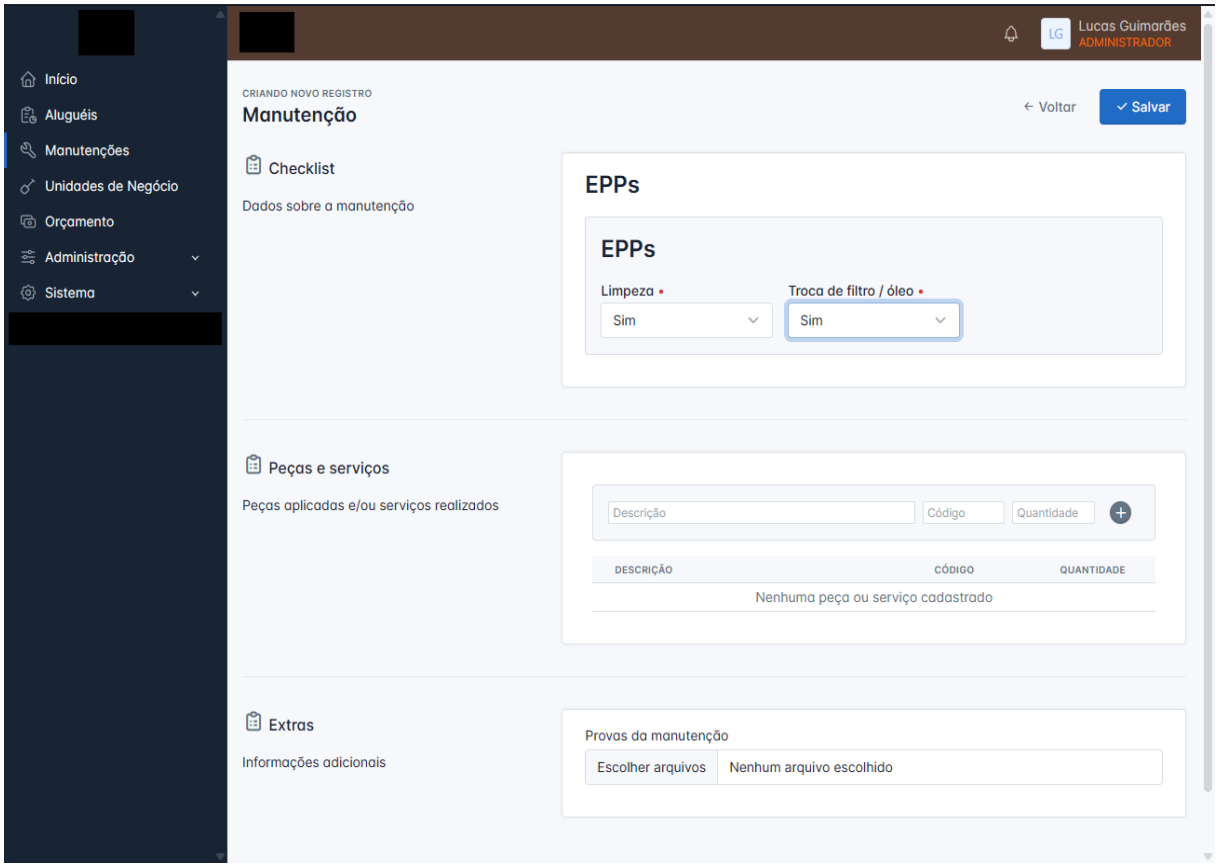
1. Na primeira etapa (Figura 19), o usuário informa os dados básicos: a unidade de negócio responsável, o item que será mantido, o prazo previsto para a manutenção e um campo para observações adicionais, caso necessário.
2. Na segunda etapa (Figura 18), são preenchidos os itens do *checklist* específico do equipamento em manutenção. Nessa mesma etapa, o usuário pode adicionar detalhes sobre os serviços executados e anexar fotos que comprovem a realização da manutenção.
3. Após completar o preenchimento de todas as informações, basta clicar no botão “Salvar” para finalizar e registrar a manutenção no sistema.

*Figura 19: Tela de cadastro de manutenção*




*Fonte: Elaborada pelo autor*

*Figura 18: Tela de checklist de cadastro de manutenção*



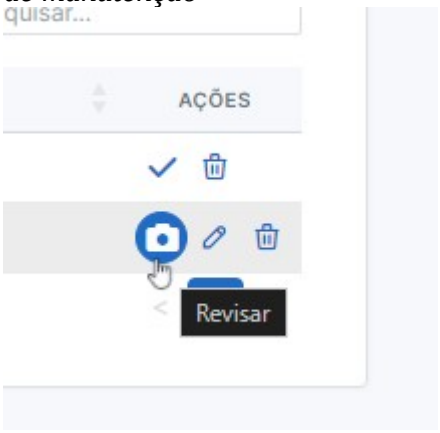

*Fonte: Elaborada pelo autor*

A revisão de manutenções ocorre através do botão “Revisar” disponível na listagem de manutenções (Figura 20). Ao clicar nesse botão, o usuário é direcionado para a tela de visualização detalhada da manutenção (Figuras 21 e 22).

Nessa tela são exibidas todas as informações registradas pela unidade de negócio durante o cadastro da manutenção, incluindo o preenchimento completo do *checklist*, a descrição dos serviços executados e as fotografias comprobatórias enviadas. Essa apresentação estruturada permite que o administrador analise criteriosamente se todos os procedimentos obrigatórios foram realizados de forma adequada e se a documentação fornecida é suficiente para validar a manutenção.

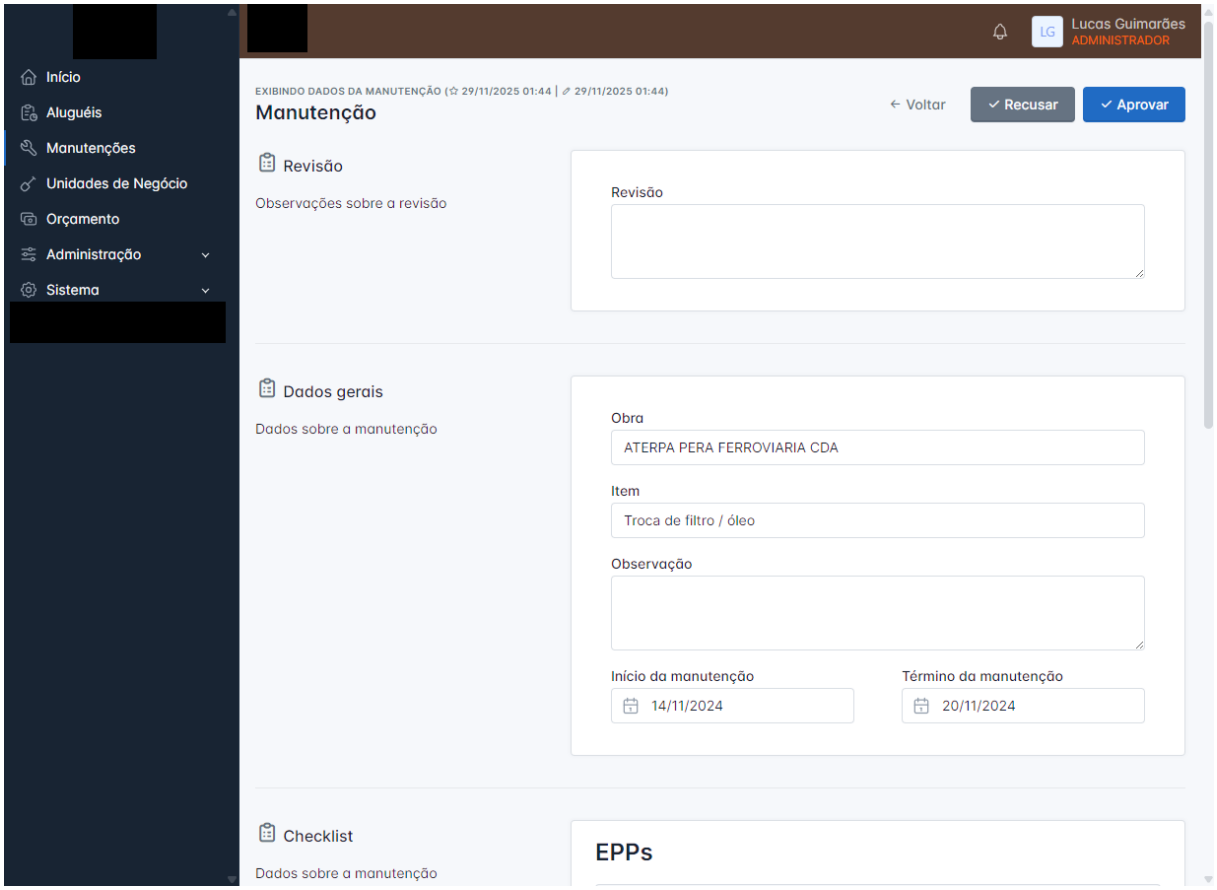
A partir dessa interface, o usuário com perfil administrativo pode tomar a decisão de aprovar ou reprovar a manutenção. Junto à decisão, o administrador pode adicionar um texto de observações, permitindo comunicar à unidade de negócio quaisquer problemas identificados, solicitar informações complementares ou simplesmente registrar comentários sobre a manutenção analisada. Essa funcionalidade de *feedback* facilita a comunicação entre a equipe administrativa e os clientes, garantindo transparência no processo de validação.

*Figura 20: Acesso para revisão de manutenção*



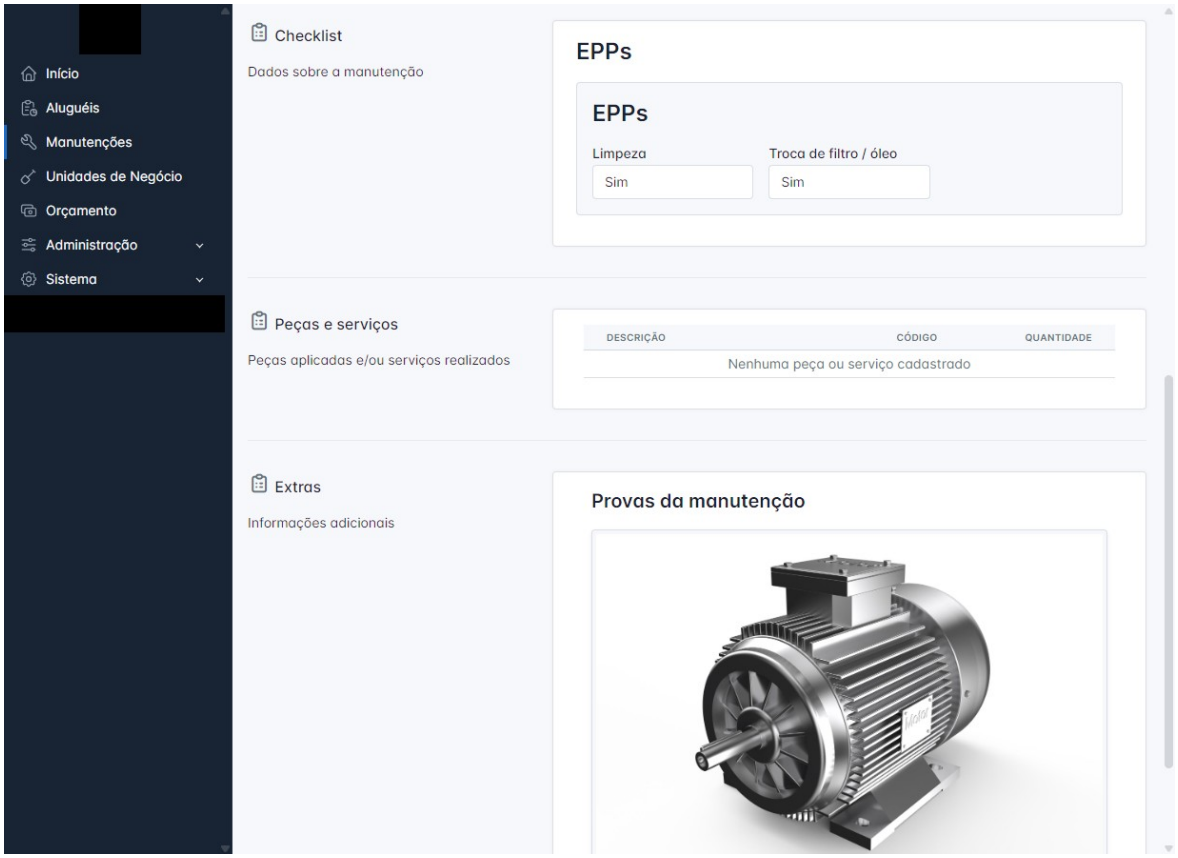

*Fonte: Elaborada pelo autor*

*Figura 21: Tela de revisão de manutenção*




*Fonte: Elaborada pelo autor*

*Figura 22: Continuação da tela de revisão de manutenção*



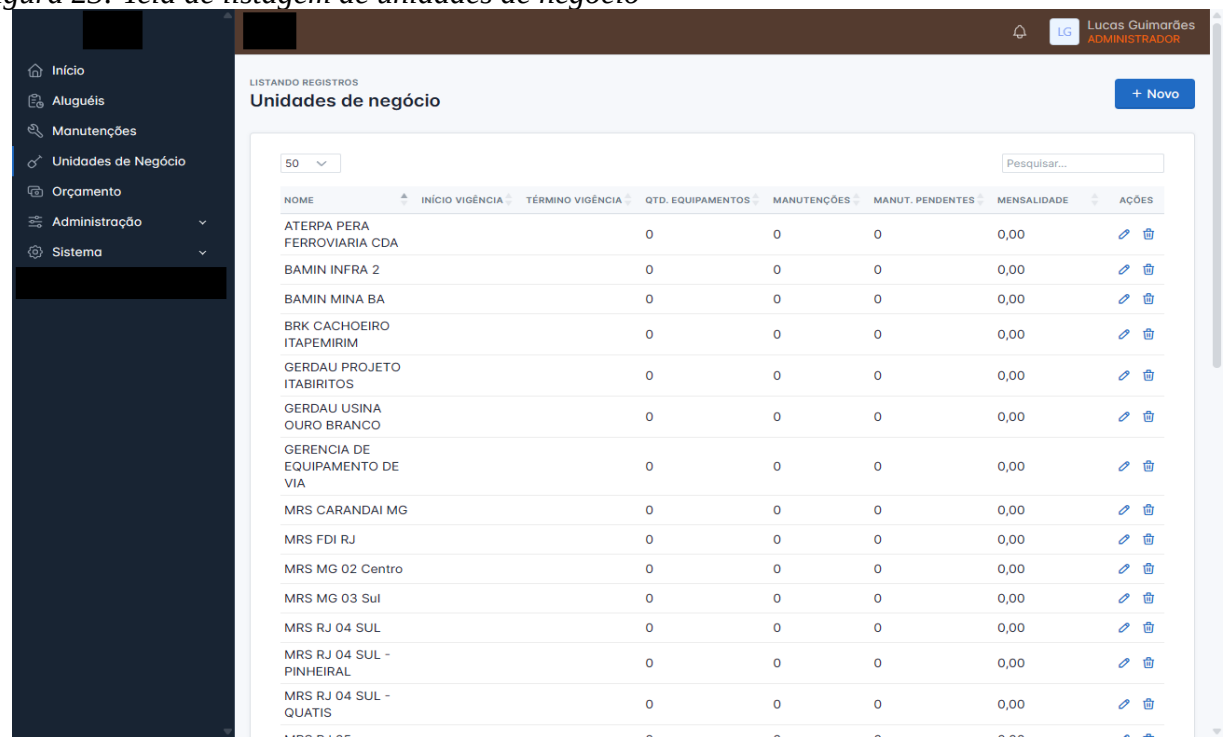

*Fonte: Elaborada pelo autor*

### 4.2.5 Tela de unidades de negócio

A tela de Unidades de Negócio (Figura 23) apresenta uma listagem completa de todas as unidades cadastradas no sistema. Para cada unidade, são exibidas informações importantes como o nome, a quantidade de equipamentos atualmente alugados, o número de manutenções realizadas e pendentes, além do valor mensal pago pela unidade referente aos aluguéis.

O sistema também disponibiliza uma tela de cadastro, que pode ser visualizada na Figura 24, para incluir novas unidades de negócio. Nessa tela, o usuário pode informar o nome da unidade, as datas de início e término da vigência do contrato, selecionar o usuário responsável pela gerência da unidade e associar o centro de custo correspondente para fins de controle financeiro.

*Figura 23: Tela de listagem de unidades de negócio*




*Fonte: Elaborada pelo autor*

*Figura 24: Tela de cadastro de unidade de negócio*

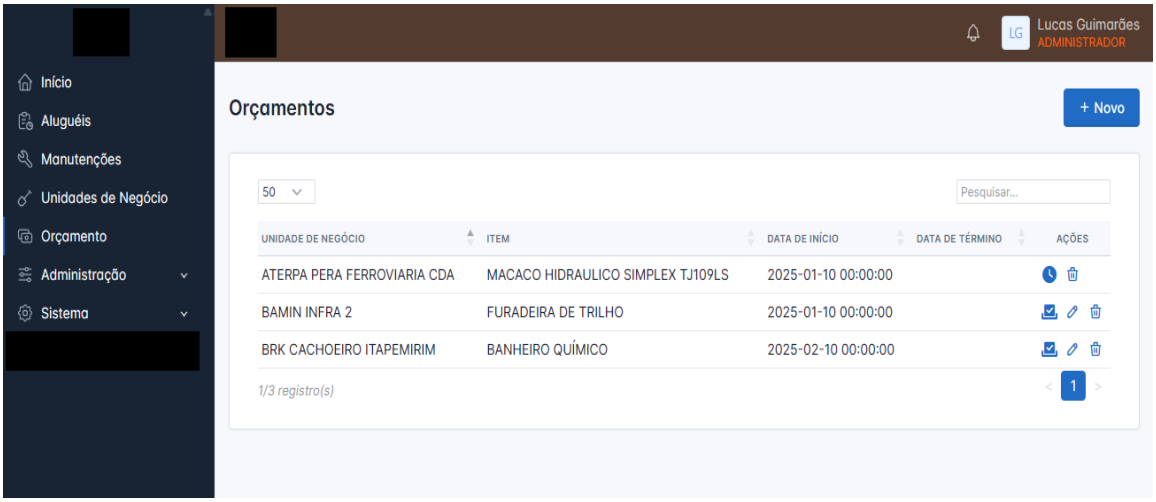

*Fonte: Elaborada pelo autor*

### 4.2.6 Tela de orçamentos

A tela de orçamentos (Figura 25) exibe uma listagem de todas as solicitações de orçamento registradas no sistema. Nessa interface, é possível visualizar informações como a unidade de negócio solicitante, o tipo de equipamento orçado e o período requisitado para o aluguel. Essa organização permite à equipe administrativa acompanhar as demandas de orçamento e planejar adequadamente a disponibilidade dos equipamentos.

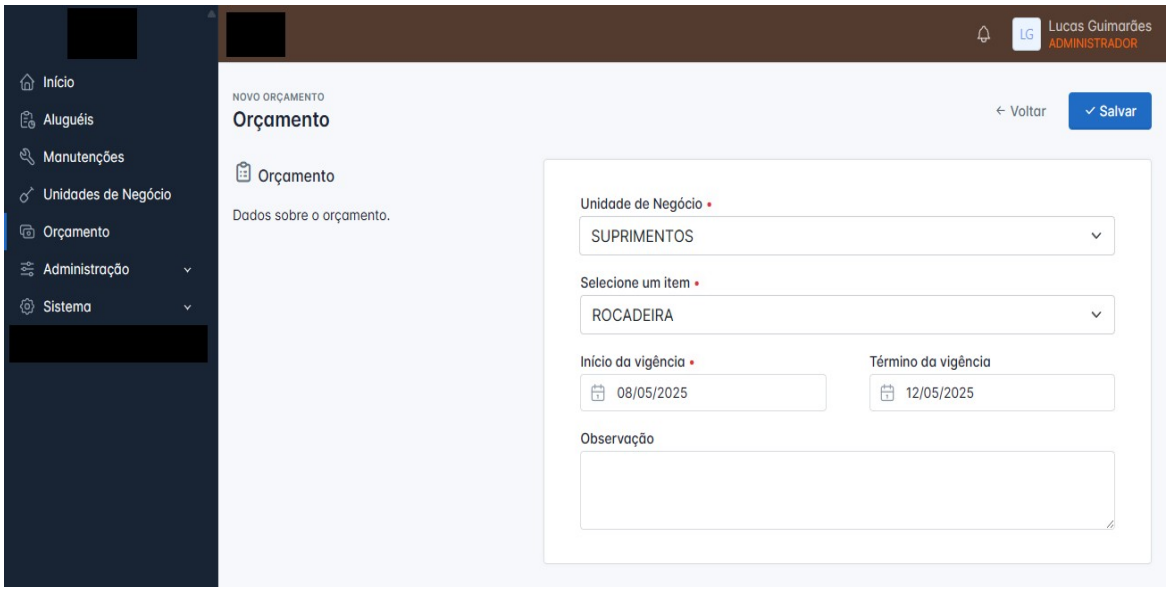
A tela também fornece acesso ao formulário de cadastro de novos orçamentos (Figura 26), onde os clientes podem requisitar uma cotação para um tipo específico de equipamento, informando o prazo desejado para o aluguel. Essa funcionalidade facilita o processo de negociação e permite que a empresa responda de forma ágil às necessidades de seus clientes, mantendo um registro organizado de todas as solicitações recebidas.

*Figura 25: Tela de listagem de orçamentos*




*Fonte: Elaborada pelo autor*

*Figura 26: Tela de cadastro de orçamento*



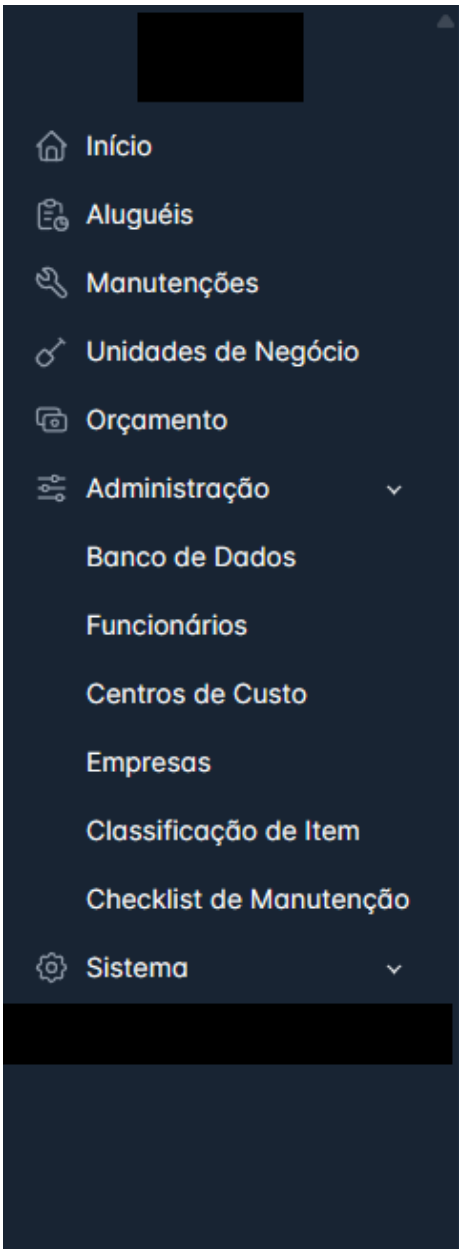

*Fonte: Elaborada pelo autor*

### 4.2.7 Menu do sistema

A estrutura do menu de navegação do sistema pode ser visualizada nas Figuras 27 e 28. A organização foi pensada para facilitar o uso no dia a dia, colocando em destaque os

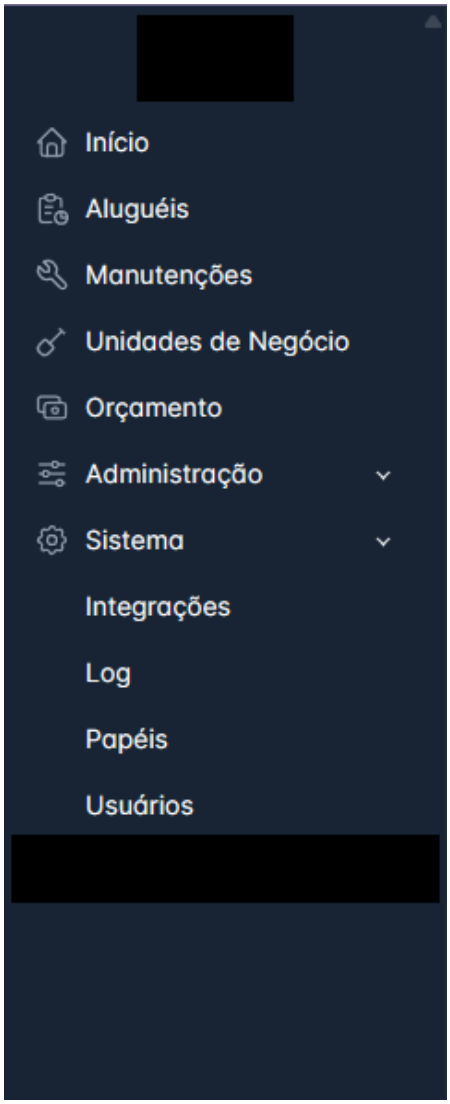
módulos mais utilizados como itens principais do menu. Já os módulos que têm função auxiliar ou que são acessados com menos frequência foram agrupados em submenus, mantendo a interface mais limpa e organizada.

*Figura 27: Menu do sistema*




*Fonte: Elaborada pelo autor*

*Figura 28: Menu do sistema continuado*




*Fonte: Elaborada pelo autor*

### 4.2.8 Métricas do código-fonte

A Tabela 1 apresenta um resumo das métricas do código fonte do sistema. Nela, é possível visualizar a quantidade de arquivos desenvolvidos em cada linguagem de programação e marcação utilizadas no projeto: CSS, HTML, SCSS, JavaScript, TypeScript e PHP.

Além disso, a tabela mostra o total de linhas de código escritas em cada uma dessas linguagens. Esses dados oferecem uma visão geral sobre o tamanho do projeto e como o desenvolvimento foi distribuído entre as diferentes tecnologias empregadas.

*Tabela 1: Métricas do código-fonte*

| Tipo de arquivo | Quantidade de arquivos | Quantidade de linhas |
|---|---|---|
| css | 114 | 642 |
| html | 4 | 928 |
| js | 229 | 1325 |
| php | 523 | 160 |
| scss | 1 | 455 |
| ts | 15 | 124 |
| Total | 886 | 3634 |

## 4.3 Desafios no desenvolvimento

Apesar das inúmeras vantagens e do amplo reconhecimento da linguagem PHP e do *framework* Laravel no contexto de desenvolvimento web, essas ferramentas apresentaram desafios significativos ao longo do desenvolvimento deste projeto, que precisaram ser compreendidos e superados para que o trabalho fosse concluído com sucesso.

A natureza dinâmica do PHP, embora proporcione flexibilidade e rapidez na escrita do código, ocasionalmente acarretou dificuldades relacionadas aos tipos de dados utilizados. Esses problemas, muitas vezes sutis, geraram conflitos que demandaram atenção cuidadosa durante a depuração, provocando comportamentos inesperados durante a execução da aplicação que precisaram ser investigados e corrigidos.

Essa característica dinâmica permite uma escrita ágil e prototipação rápida, possibilitando que funcionalidades sejam implementadas com velocidade e baixo esforço inicial, sem a necessidade constante de definir explicitamente tipos, interfaces ou assinaturas. No entanto, ao longo do projeto, foi possível observar que essa mesma vantagem pode se

tornar um desafio para a legibilidade e manutenção do código em sistemas mais complexos. A ausência de tipagem explícita reduz a capacidade do código em oferecer auto-documentação clara sobre o comportamento esperado das variáveis e funções. Para contornar essa limitação, foi necessário adotar práticas complementares como documentação através de comentários, convenções consistentes de nomenclatura e validações mais rigorosas em pontos críticos do sistema, o que contribuiu para manter a qualidade e a compreensibilidade do código mesmo em uma base de código extensa.

Outro aspecto que exigiu adaptação foi a abstração promovida pelo Laravel, especialmente na camada de acesso ao banco de dados. A utilização do Eloquent ORM, embora simplifique significativamente a interação com bancos de dados por meio de métodos intuitivos e expressivos, ocasionalmente mascarou a complexidade das consultas realizadas. Em situações onde o desempenho ou o comportamento das queries precisavam ser otimizados, foi necessário investigar mais profundamente as operações executadas, utilizando ferramentas de *log* de consultas e análise de performance. Com o tempo e a prática, essas investigações tornaram-se mais eficientes, permitindo identificar e resolver problemas de forma mais ágil.

Adicionalmente, a atividade de depuração enfrentou obstáculos relacionados à instabilidade ocasional de ferramentas utilizadas para análise em tempo real do código PHP, como o Xdebug. Em determinados momentos, essa ferramenta apresentou comportamentos inconsistentes, não respondendo adequadamente a comandos básicos como inserção ou remoção de *breakpoints*. Tal situação resultou em interrupções no fluxo de desenvolvimento e exigiu a busca por abordagens alternativas de depuração, como o uso mais intensivo de *logs* detalhados e técnicas de isolamento de problemas através de redução progressiva do código. Essas estratégias alternativas, embora inicialmente mais trabalhosas, foram essenciais no processo de desenvolvimento.

É importante destacar que o processo de enfrentar e resolver essas dificuldades contribuiu significativamente para o aprendizado e para o desenvolvimento de habilidades importantes de resolução de problemas. Ao final do projeto, foi possível entregar um sistema funcional, estável e que atende às necessidades da empresa, demonstrando que os obstáculos encontrados, embora reais e exigentes, não impediram o sucesso da implementação.

# 5 Considerações finais

O desenvolvimento deste Trabalho de Conclusão de Curso permitiu alcançar o objetivo principal de entregar uma solução de software funcional, capaz de atender às demandas iniciais da empresa parceira. Além do produto final, esta foi uma etapa fundamental para a consolidação do conhecimento acadêmico e para a aquisição de experiência prática direta nos quesitos de engenharia de software e no planejamento de um sistema. Lidar com o levantamento de requisitos, as constantes mudanças de escopo e a definição de prioridades em um cronograma limitado foi um desafio que gerou aprendizado significativo sobre a gestão de um projeto real. A execução também exigiu uma imersão prática em tecnologias web, servindo como uma oportunidade para revisar o funcionamento da linguagem PHP, aplicar de forma robusta os conceitos do *framework* Laravel e reforçar o conhecimento prático no gerenciamento de banco de dados, consolidando as habilidades de modelagem e manipulação de dados com o PostgreSQL.

Apesar do estudo de novas tecnologias ter sido a base para a conclusão deste trabalho, muitos conhecimentos adquiridos nas disciplinas do Bacharelado em Ciência da Computação do IFMG Campus Formiga foram de suma importância para o desenvolvimento desta versão do sistema. Destacam-se os conceitos fundamentais aprendidos em Engenharia de Software, que guiaram o levantamento de requisitos e o gerenciamento das mudanças de escopo; Algoritmos e Estruturas de Dados, essenciais para a implementação da lógica de negócio; além das bases fornecidas por Desenvolvimento Web, Banco de Dados e Interface Humano-computador, que foram aplicadas diretamente na construção da plataforma. Esta experiência prática comprovou como a base teórica sólida é indispensável para a resolução de problemas complexos no desenvolvimento de software.

O projeto, apesar de concluído com sucesso, entregando um núcleo funcional que serve como alicerce sólido para uma plataforma de gestão robusta, representa apenas um ponto de partida, com um vasto potencial de expansão. O cronograma de desenvolvimento, sendo inerentemente limitado, exigiu uma priorização rigorosa e, consequentemente, um escopo reduzido para garantir uma entrega viável e de qualidade dentro do prazo estipulado.

Além disso, o processo de desenvolvimento foi dinâmico, marcado por diversas mudanças de requisitos solicitadas pela empresa ao longo do ciclo do projeto. Essas alterações, embora importantes para alinhar o produto às necessidades emergentes do cliente,

impactaram o cronograma. Em especial, mudanças substanciais requisitadas já na fase final do desenvolvimento exigiram uma renegociação e uma extensão de prazo de vários meses. A consequência direta desses desafios de tempo e das mudanças de escopo foi a redução da quantidade de funcionalidades que a empresa desejava ver implementadas nesta primeira versão do sistema. Portanto, na seção “Trabalhos futuros” são detalhadas as evoluções naturais e as novas funcionalidades identificadas como cruciais para as próximas fases do projeto.

## 5.1 Trabalhos futuros

Tratando-se de um protótipo, o próximo grande passo para se realizar no contexto desse projeto é uma análise do impacto operacional obtido como resultado da implantação da solução. Espera-se que com essa solução sejam obtidos os objetivos de melhoria de eficiência e segurança operacional, porém para o veredito é necessário um prazo extenso entre a implantação dessa solução e de uma análise dos resultados. A implantação ocorre no mesmo período em que ocorre a defesa desse trabalho de conclusão de curso e, portanto, não pôde ser incluída no mesmo.

Quanto a melhorias do sistema, a expansão mais significativa e de maior impacto seria o desenvolvimento do aplicativo móvel auxiliar de manutenção. Esta funcionalidade, que não foi sequer incluída no escopo deste trabalho, é vista como essencial para a operação em campo. O objetivo seria permitir que as equipes técnicas realizassem o cadastro e a consulta de manutenções diretamente de seus dispositivos móveis, agilizando o fluxo de trabalho e, principalmente, possibilitando que o mesmo seja realizado em áreas remotas sem a cobertura de serviços de telecomunicação. A criação deste aplicativo representa, talvez, o passo mais importante para aumentar a eficiência operacional que o sistema se propõe a resolver.

Outra funcionalidade de grande valor que foi deixada para uma implementação futura é o módulo de gestão de documentos, especificamente focado no cadastro de manuais de equipamentos. A ideia original era que os gestores pudessem anexar os manuais técnicos, guias de operação e especificações de cada equipamento cadastrado na plataforma. Isso permitiria que as equipes de manutenção nas obras pudessem acessar rapidamente essa documentação de referência, centralizando a informação e garantindo que todos estivessem consultando as versões corretas. Esta adição tornaria o sistema uma fonte central de verdade

não apenas para o histórico de manutenção, mas também para o conhecimento técnico dos ativos.

Olhando para as funcionalidades que foram implementadas, o controle de usuários é um ponto claro de melhoria. A estrutura atual é funcional, porém bastante crua, atendendo apenas aos requisitos mínimos de diferenciação de acesso. Um caminho natural de evolução é refinar esse sistema de permissões. A melhoria mais urgente identificada é a implementação do suporte a múltiplos usuários com perfil de gerente para uma mesma unidade de negócio. A limitação atual de apenas um gerente por unidade não reflete a realidade operacional de muitas empresas, onde a gestão é compartilhada. Expandir isso para um sistema de papéis e permissões mais granular permitiria ao sistema adaptar-se a estruturas organizacionais mais complexas.

Finalmente, à medida que a base de usuários do sistema crescer e o volume de dados aumentar, será fundamental investir em monitoramento. O sistema, em seu estado atual, carece de métricas robustas de performance, uso e saúde da aplicação. A implementação de ferramentas de observabilidade, como *logs* estruturados, métricas de tempo de resposta e painéis de acompanhamento, é um caminho natural para garantir a escalabilidade e a confiabilidade da plataforma. Essas métricas permitirão à equipe de desenvolvimento identificar gargalos de performance de forma proativa, entender como os usuários interagem com as funcionalidades e tomar decisões mais informadas sobre a alocação de infraestrutura e sobre quais áreas do sistema necessitam de otimização.

# REFERÊNCIAS

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro: Elsevier, 2007

BONIATI, Bruno Batista; SILVA, Teresinha Letícia. **Fundamentos de Desenvolvimento Web.** Rio Grande do Sul: [s. n.], 2013.

CORREA, Daniel; VALLEJO, Paola. **Practical Laravel**: Decelop clean MVC web applications. [S. l.: s. n.], 2022.

DB-ENGINES Ranking - Trend Popularity. [S. l.], 2024. Disponível em: https://db-engines.com/en/ranking_trend. Acesso em: 18 ago. 2024.

DUCKETT, Jon. **HTML & CSS**: Design and build Websites. Indianapolis, Indiana: John Wiley & Sons, Inc., 2011.

TEAM EMB. **Manual Process vs. Automation**: What‘s Best For Your Business?. [*S. l.*], 2024. Disponível em: https://blog.emb.global/manual-process-vs-automation/. Acesso em: 18 ago. 2024.

GIT. Disponível em: https://git-scm.com. Acesso em: 18 ago. 2024.

GOŁOFIT, Piotr. **Is PHP still relevant in 2023?**. Medium, 2023. Disponível em: https://medium.com/accesto/is-php-still-relevant-in-2023-16801dc84700. Acesso em: 28 ago. 2024.

GRILLO, Filipe Del Nero; FORTES, Renata Pontin de Mattos. **Aprendendo JavaScript**. São Carlos: [s. n.], 2008.

HIRAMA, Kechi. **ENGENHARIA DE SOFTWARE**: Qualidade e Produtividade com Tecnologia. Rio de Janeiro: Elsevier, 2011.

JETBRAINS. Disponível em: https://www.jetbrains.com. Acesso em: 18 ago. 2024.

LARAGON. Disponível em: https://laragon.org. Acesso em: 18 ago. 2024.

NASCIMENTO, Lucas Gennari. **Modelos de desenvolvimento de software**: Análise comparativa entre os modelos ágeis e tradicionais. São Paulo: [s. n.], 2015.

NONATO, Livia. **Automação de processos**: o que é, quais os tipos e como fazer. [S. l.], 2024. Disponível em: https://blog.aevo.com.br/automacao-de-processos. Acesso em: 28 ago. 2024.

**PHP: História do PHP – Manual**. Disponível em: https://www.php.net/manual/pt_BR/history.php.php. Acesso em: 10 nov. 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software**: Uma abordagem profissional. 9. ed. Porto Alegre: AMGH Editora Ltda., 2021.

SILVA, Maurício Samy. **HTML5**: A linguagem de marcação que revolucionou a web. 2. ed. São Paulo: Novatec, 2019.

SOMMERVILLE, Ian. **ENGENHARIA DE SOFTWARE**. 9. ed. São Paulo: Pearson, 2011.

STAUFFER, Matt. **Laravel Up & Running**: A Framework for Building Modern PHP Apps. 2. ed. [S. l.]: O'REILLY, 2019.

VIEIRA, Mariana Barbosa. **O impacto da Transformação Digital na Contabilidade**: O impacto da Transformação Digital na Contabilidade. [S. l.: s. n.], 2023.