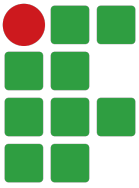


INSTITUTO FEDERAL DA BAHIA
CAMPUS VALENÇA
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAINER VÍTOR SANTOS DE SOUSA

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE ESTOQUE UTILIZANDO PHP E LARAVEL

Valença-BA
2025

RAINER VÍTOR SANTOS DE SOUSA

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE ESTOQUE UTILIZANDO PHP E LARAVEL

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia da Bahia como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Addson Araújo da Costa

Valença-BA
2025

Dedico este trabalho à minha família, pelo apoio incondicional e pela inspiração constante, e a todos que, de alguma forma, contribuíram para a realização deste sonho. A vocês, minha eterna gratidão.

Agradecimentos

Agradeço primeiramente ao meu orientador, pela valiosa orientação, paciência e incentivo que foram fundamentais para a concretização deste trabalho. Sua expertise e disponibilidade foram cruciais em cada etapa do processo.

Estendo meus sinceros agradecimentos aos professores e ao corpo técnico da instituição que, com seu apoio e conhecimento, contribuíram diretamente para o desenvolvimento e aprimoramento deste Trabalho de Conclusão de Curso (TCC). Suas contribuições foram essenciais para superar os desafios e alcançar os objetivos propostos.

Por fim, agradeço a todos que, direta ou indiretamente, ofereceram apoio e compreensão durante a elaboração deste projeto.

FICHA CATALOGRÁFICA ELABORADA PELO SISTEMA DE BIBLIOTECAS DO IFBA, COM OS DADOS FORNECIDOS PELO(A) AUTOR(A)

S725d Sousa, Rainer Vitor Santos de

Desenvolvimento de um sistema de controle de estoque utilizando PHP e LARAVEL: / Rainer Vitor Santos de Sousa; orientadora Addson Araújo da Costa -- Valença : IFBA, 2025.

37f.

Trabalho de Conclusão de Curso (Análise e Desenvolvimento de Sistemas) -- Instituto Federal da Bahia, 2025.

1. Laravel Framework. 2. Eloquent ORM. 3. Arquitetura MVC. 4. API RESTful. 5. Desenvolvimento de sistema WEB. I. Costa, Addson Araújo da, orient. II. TÍTULO.

CDD:004.43

FOLHA DE APROVAÇÃO

DISCENTE:

RAINER VITOR SANTOS DE SOUSA

TEMA:

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE ESTOQUE USANDO PHP E LARAVEL

Trabalho de Conclusão de Curso de graduação apresentado como requisito parcial para obtenção do título de Bacharel em Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA), *campus* Valença.

Aprovado em

Valença, 27 de Agosto de 2025

Banca examinadora

Addson Araújo da Costa

Prof. Orientador

Instituto Federal de Educação, Ciência e Tecnologia – Campus Valença

André Luiz Romano Medeiros

Prof. Examinador Banca

Instituto Federal de Educação, Ciência e Tecnologia – Campus Valença

Prof. Examinador Banca

Instituto Federal de Educação, Ciência e Tecnologia – Campus Valença

"A desordem vem da ordem, a covardia da coragem, a fraqueza da força."

Sun Tzu, A Arte da Guerra

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE ESTOQUE UTILIZANDO PHP E LARAVEL

Resumo

Este trabalho tem como objetivo o desenvolvimento de um sistema WEB de controle de estoque, direcionado a pequenas empresas. Uma dificuldade encontrada por grande parte desses negócios é a realização do controle de entrada e saída de produtos de forma manual, um processo suscetível a erros que frequentemente ocasiona prejuízos financeiros. O desenvolvimento do sistema aqui descrito visa a melhora na gestão do estoque, permitindo o registro das movimentações de produtos e o acesso a relatórios precisos e facilitados. A ferramenta busca, assim, auxiliar os gestores na tomada de melhores decisões e na otimização de suas operações. A validação da eficácia do sistema e o cumprimento dos objetivos foram confirmados por meio de um questionário aplicado a usuários reais, que atestaram a usabilidade e a relevância das funcionalidades.

Por fim, todos os objetivos estabelecidos inicialmente foram cumpridos, visto que o sistema foi desenvolvido e possui todas as funcionalidades essenciais, demonstrando sua viabilidade e potencial para otimizar a gestão de estoque em pequenas empresas, contribuindo para a redução de perdas e a melhoria da eficiência operacional.

Palavras-chave: Laravel Framework. Eloquent ORM. Arquitetura MVC. API RESTful.

DEVELOPMENT OF A STOCK CONTROL SYSTEM USING PHP AND LARAVEL

Abstract

This work aims to develop a web-based inventory control system targeted at small businesses. A common difficulty faced by many of these companies is managing product inflows and outflows manually, a process prone to errors that often leads to financial losses.The development of the system described herein seeks to improve inventory management by allowing the recording of product movements and providing access to accurate and easily generated reports. The tool is intended to assist managers in making better decisions and optimizing their operations. The effectiveness of the system and the achievement of its objectives were validated through a questionnaire applied to real users, who confirmed the usability and relevance of its functionalities. Finally, all initially established objectives were accomplished, as the system was developed with all essential functionalities, demonstrating its feasibility and potential to optimize inventory management in small businesses, contributing to the reduction of losses and the improvement of operational efficiency.

Keywords: Laravel Framework. Eloquent ORM. MVC Architecture. RESTful API.

Lista de figuras

Lista de abreviaturas e siglas

ADS	Análise e Desenvolvimento de Sistemas
CRUD	*Create, Read, Update, Delete*
CSRF	*Cross-Site Request Forgery*
CSS	*Cascading Style Sheets*
DER	Diagrama de Entidade-Relacionamento
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MVC	*Model-View-Controller*
ORM	*Object Relational Mapper*
PHP	*Hypertext Preprocessor*
REST	*Representational State Transfer*
SQL	*Structured Query Language*
TCC	Trabalho de Conclusão de Curso
URI	*Uniform Resource Identifiers*
URL	*Uniform Resource Locator*
WEB	*World Wide Web*

Sumário

1 Introdução

O controle de estoque é um pilar fundamental para a saúde financeira e operacional de qualquer empresa, especialmente as de pequeno porte. Em um cenário de mercado cada vez mais competitivo, a gestão eficiente de produtos torna-se um diferencial estratégico. Pequenas empresas, muitas vezes, operam com recursos limitados e margens de lucro apertadas, o que torna qualquer desperdício ou ineficiência no estoque um fator crítico para sua sobrevivência e crescimento.

Estudos recentes indicam que grande parte das micro e pequenas empresas brasileiras ainda enfrentam dificuldades com a gestão de seus processos internos, incluindo o controle de estoque, frequentemente utilizando métodos manuais ou pouco informatizados (SEBRAE, 2022). Tal cenário pode acarretar em perdas financeiras significativas e dificuldade na tomada de decisões estratégicas, evidenciando a urgência de soluções tecnológicas acessíveis e eficientes para este segmento.

Apesar da ampla disponibilidade de tecnologias para gestão de estoque, estudos mostram que uma parcela significativa das micro e pequenas empresas ainda não utiliza sistemas informatizados para esse fim, sendo que 76% das pequenas e médias empresas (PMEs) ainda não utilizam nenhum sistema específico para gestão dos negócios. Dessas, 15% fazem seus controles em papel e 61% dependem de planilhas ou outras soluções não-sistêmicas (Sebrae; Gartner, 2024).

Outras pesquisas mais recentes reforçam essa realidade. Segundo levantamento feito pelo Conta Simples e Visa, realizado entre novembro e dezembro de 2024, 39% das micro, pequenas e médias empresas brasileiras ainda utilizam planilhas ou cadernos para controle de recursos, índice que ainda sobe para 45% entre microempresas. Os principais motivos apontados para a não adoção de sistemas informatizados foram o custo elevado das soluções (44%) e a comodidade com os métodos manuais já utilizados (45%) (Amazonas Atual, 2025).

A ausência de um sistema de controle adequado pode levar a uma série de problemas, como a falta de produtos para atender à demanda, o excesso de itens parados gerando custos de armazenagem, a perda de validade de produtos perecíveis, e a dificuldade em identificar gargalos na cadeia de suprimentos. Esses problemas não apenas impactam diretamente o faturamento, mas também a satisfação do cliente e a reputação da empresa.

A motivação para o desenvolvimento deste projeto surgiu desta observação das dificuldades enfrentadas por pequenas empresas em gerenciar seus estoques. Muitos desses negócios ainda dependem de métodos manuais, como planilhas e cadernos, que são pro-

pensos a erros, consomem tempo e não fornecem informações em tempo real para tomadas de decisão.

Os conhecimentos obtidos durante a formação em Análise e Desenvolvimento de Sistemas proporcionou o ambiente ideal para aplicar os conhecimentos teóricos na construção de uma solução prática que pudesse mitigar esses desafios. Acreditamos que um sistema de controle de estoque acessível e eficiente pode empoderar esses empreendedores, otimizando suas operações e permitindo que se concentrem no crescimento de seus negócios.

A relevância acadêmica deste trabalho reside na aplicação prática de conceitos e metodologias de Desenvolvimento Web, Banco de Dados, Engenharia de Software, dentre outros conhecimentos adquiridos ao longo da formação em Análise e Desenvolvimento de Sistemas (ADS). O projeto demonstra a capacidade de transformar uma necessidade de mercado em uma solução tecnológica, utilizando ferramentas e padrões de arquitetura modernos.

A escolha do *framework* Laravel, com sua arquitetura MVC e rotas RESTful, juntamente com o uso do Eloquent ORM e MySQL, oferece um estudo de caso valioso sobre o desenvolvimento de aplicações web escaláveis e de fácil manutenção. Do ponto de vista prático, o sistema desenvolvido visa oferecer uma ferramenta intuitiva e eficaz para pequenas empresas, permitindo-lhes um controle mais preciso de seus produtos, a redução de perdas e a otimização de seus processos de compra e venda.

1.1 Objetivos

O objetivo geral deste trabalho é prover o gerenciamento eficiente de produtos, entradas, saídas e relatórios através do desenvolvimento de um sistema de controle de estoque web para pequenas empresas, utilizando PHP e Laravel. Para alcançar este objetivo geral, foram definidos os seguintes objetivos específicos:

1. Otimizar a organização do estoque, provendo uma interface intuitiva para o cadastro e gerenciamento de produtos, fornecedores/clientes e categorias.
2. Apoiar a tomada de decisões gerenciais, através da geração de relatórios simplificados e funcionalidades de controle de acesso de usuários.
3. Assegurar a escalabilidade e manutenibilidade da aplicação através da adoção de padrões de arquitetura modernos e tecnologias robustas.

1.2 Organização do trabalho

Este trabalho está estruturado da seguinte forma: o Capítulo 2 descreve os fundamentos teóricos e tecnológicos do sistema, abrangendo PHP, Laravel, arquitetura MVC, serviços REST, bancos de dados relacionais e o Eloquent ORM; o Capítulo 3 apresenta a metodologia empregada; o Capítulo 4 aborda a implementação técnica, detalhando a estrutura do projeto e exemplos práticos de rotas REST e modelagem do banco de dados com a utilização do Eloquent ORM; o Capítulo 5 analisa os resultados obtidos e propõe melhorias para trabalhos futuros; e, por fim, o Capítulo 7 apresenta as considerações finais.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos e fundamentos teóricos que embasam o desenvolvimento do sistema de controle de estoque. Serão abordadas as principais áreas do conhecimento e tecnologias que foram cruciais para a concepção e implementação da solução, fornecendo o suporte necessário para a compreensão das escolhas técnicas realizadas.

2.1 Desenvolvimento Web

O desenvolvimento *World Wide Web* (WEB) refere-se ao processo de criação e manutenção de aplicações e sites para a internet (DIO, 2025). Essa área abrange uma vasta gama de tecnologias e conceitos, que se dividem principalmente em duas grandes frentes: o *frontend* e o *backend*.

O *frontend* é responsável pela interface com o usuário, ou seja, tudo o que o usuário vê e com que interage diretamente no navegador. Isso inclui o design, a estrutura e o comportamento da página. As tecnologias predominantes no frontend são *HyperText Markup Language* (HTML) para a estruturação do conteúdo, *Cascading Style Sheets* (CSS) para a estilização visual e JavaScript para a interatividade e dinamismo da interface (Docs, 2025).

O *backend*, por sua vez, lida com a lógica de negócio, o gerenciamento de dados e a comunicação com o servidor. Ele é responsável por processar as requisições do *frontend*, interagir com o banco de dados, realizar cálculos, executar um conjunto de tarefas e retornar as informações necessárias para a interface do usuário.

Linguagens de programação como PHP, Python, Java e Node.js são comumente utilizadas no *backend*. O desenvolvimento de programas utilizando essas linguagens pode ser facilitado pelo uso de *frameworks*, que são ferramentas que fornecem estruturas e bibliotecas pré-definidas, agilizando o processo de desenvolvimento e promovendo a organização do código (Kinsta, 2025).

A comunicação entre o *frontend* e o *backend* geralmente ocorre por meio de protocolos como *Hypertext Transfer Protocol* (HTTP) e padrões de arquitetura como *Representational State Transfer* (REST). Protocolos são um conjunto de regras de comunicação previamente definidas que governam a troca de informações entre emissores e receptores (W3Schools, 2023). Ao se utilizar protocolos como HTTP e padrões como REST, todo o processo de comunicação segue regras padronizadas, garantindo a interoperabilidade e a consistência na troca de dados entre as diferentes partes da aplicação.

2.2 PHP

Hypertext Preprocessor (PHP) é uma linguagem de *script* de código aberto, amplamente utilizada para o desenvolvimento web do lado do servidor. Criada em 1994 por Rasmus Lerdorf, o PHP se tornou uma das linguagens mais populares para a construção de sites dinâmicos e aplicações web, devido à sua facilidade de aprendizado, vasta documentação e grande comunidade de desenvolvedores. O PHP é executado no servidor, gerando conteúdo HTML que é enviado para o navegador.

A escolha do PHP como linguagem principal para o desenvolvimento do sistema se fundamenta em sua vasta adoção global e na maturidade de seu ecossistema. Sendo uma linguagem de *script* de código aberto, oferece a vantagem de ser gratuito e contar com uma comunidade de desenvolvedores extremamente ativa, o que se traduz em uma rica documentação e um suporte contínuo para a resolução de problemas (W3Schools, 2023).

Sua arquitetura e funcionalidades são intrinsecamente adequadas para o desenvolvimento de aplicações web dinâmicas, proporcionando flexibilidade e escalabilidade para projetos de diferentes portes.

2.3 Laravel

Laravel é um *framework* PHP de código aberto para o desenvolvimento de aplicações web. Ele foi criado por Taylor Otwell em 2011 e rapidamente se tornou um dos *frameworks* mais populares do ecossistema PHP, conhecido por sua sintaxe elegante e expressiva, que visa tornar o desenvolvimento mais agradável e produtivo. Laravel oferece uma série de ferramentas e recursos que simplificam tarefas comuns de desenvolvimento web, como roteamento, autenticação, sessões, cache e integração com banco de dados.

O Laravel foi selecionado para este projeto devido à sua capacidade de acelerar o desenvolvimento e garantir a robustez da aplicação. Ele implementa nativamente o padrão arquitetural *Model-View-Controller* (MVC), que promove a separação de responsabilidades e facilita a manutenção do código. Além disso, o Laravel oferece um sistema de roteamento elegante, que simplifica a definição de *Uniform Resource Locator* (URL) amigáveis e a organização das requisições.

O Eloquent ORM por sua vez, abstrai a interação com o banco de dados, permitindo a manipulação de dados de forma intuitiva e segura. A inclusão de um sistema de *migrations* para versionamento do banco de dados e recursos de segurança integrados, como proteção contra ataques de *Cross-Site Request Forgery* (CSRF) e *SQL Injection*, reforça a escolha do Laravel como uma plataforma completa e confiável para o desenvolvimento de aplicações web (Laravel, 2025b).

2.4 Padrão de Arquitetura MVC

O padrão de arquitetura MVC é amplamente utilizado no desenvolvimento de aplicações web para separar a lógica de negócio da interface do usuário. O MVC divide a aplicação em três componentes principais: *Model*, *View* e *Controller* (Docs, 2023). As três camadas são:

- *Model* (Modelo): Representa os dados e a lógica de negócio da aplicação. Ele interage diretamente com o banco de dados, realizando operações de persistência e validação de dados.
- *View* (Visão): É responsável pela apresentação dos dados ao usuário. Ela exibe as informações de forma visual, sem conter lógica de negócio.
- *Controller* (Controlador): Atua como um intermediário entre o *Model* e a *View*. Ele recebe as requisições do usuário, processa-as, interage com o Model para obter ou manipular dados e, em seguida, seleciona a *View* apropriada para exibir a resposta ao usuário.

Essa separação de responsabilidades facilita a manutenção, testabilidade e extensibilidade do código, tornando o desenvolvimento mais organizado e eficiente.

2.5 REST

REST é um estilo arquitetural que define um conjunto de princípios e restrições que, quando aplicados, resultam em um sistema escalável, flexível e fácil de integrar. Define como os sistemas devem se comunicar através do protocolo HTTP, utilizando operações padronizadas (GET, POST, PUT, DELETE) sobre recursos identificáveis por *Uniform Resource Identifiers* (URI). URI é um identificador de um recurso na web, podendo ser apenas um nome ou local (ex.: urn:produto:04514), enquanto URL é um tipo de URI que indica o endereço e como acessar esse recurso (ex.: https://www.sistema.com/produtos) (Fielding et al., 2005).

Essa abordagem promove a interoperabilidade e a escalabilidade, permitindo que o sistema desenvolvido possa ser facilmente integrado a outras plataformas, como aplicações móveis nativas ou sistemas de terceiros, ampliando seu potencial de uso e sua vida útil (Fielding, 2000). Os sistemas que seguem o estilo REST são chamados de RESTful.

2.6 Banco de Dados Relacional e Eloquent ORM

Um banco de dados relacional é um tipo de banco de dados que armazena e fornece acesso a pontos de dados que estão relacionados entre si. Ele é baseado no modelo relacional, que organiza os dados em tabelas compostas por linhas (registros) e colunas (atributos) (Oracle, 2021). As relações entre as tabelas são estabelecidas por meio de chaves primárias e estrangeiras, garantindo a integridade e a consistência dos dados.

Um *Object Relational Mapper* (ORM) é uma técnica de mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam (DevMedia, 2025). O Eloquent ORM é uma implementação de ORM fornecida pelo Laravel. Ele permite que os desenvolvedores interajam com o banco de dados usando objetos PHP, em vez de escrever consultas *Structured Query Language* (SQL) diretamente. Isso simplifica o desenvolvimento, torna o código mais legível e reduz a chance de erros. O Eloquent ORM mapeia as tabelas do banco de dados para classes PHP, e as linhas das tabelas para instâncias dessas classes, facilitando a manipulação dos dados (Laravel, s.d.).

2.7 Elementos da Gestão de Estoque

Ching (2010) apresenta conceitos fundamentais na área de gestão de estoques, definindo aspectos essenciais para organizar e gerir o estoque de forma eficiente, garantindo a disponibilidade de produtos quando necessário e evitando excessos que geram custos desnecessários. Além disso, a obra destaca a importância de categorizar os produtos de acordo com critérios como giro, valor e criticidade, permitindo priorizar o controle do estoque e a reposição dos itens mais relevantes, contribuindo para uma gestão mais eficiente.

Outro ponto importante abordado é a necessidade de manter cadastros atualizados de fornecedores e clientes, garantindo a rastreabilidade do fluxo de mercadorias, essencial para planejar compras, acompanhar entregas e manter um histórico confiável das movimentações. A divisão dos produtos em categorias também facilita a gestão, possibilita buscas rápidas, gera relatórios precisos e apoia tomadas de decisão mais fundamentadas.

Com base nesses conceitos, é possível identificar os recursos mais comuns e essenciais em sistemas de controle de estoque voltados a pequenas empresas, tais como: cadastro e gerenciamento de produtos, controle de entradas e saídas de estoque, geração de relatórios, alertas de baixo estoque, categorização de produtos e funcionalidades de busca e filtragem de dados. Esses recursos fornecem um referencial teórico para compreender as necessidades e expectativas do público-alvo e servem como base para o desenvolvimento de sistemas que apoiem a gestão operacional e estratégica de estoques.

3 Metodologia

Este capítulo detalha a metodologia empregada no desenvolvimento do sistema de controle de estoque, delineando as etapas e as abordagens técnicas que guiaram o projeto. Adotando um processo iterativo e incremental, a metodologia abrange desde a pesquisa inicial em bibliografia especializada em Gestão de estoque e o levantamento aprofundado dos requisitos funcionais e não funcionais, passando pela modelagem do sistema com diagramas de Casos de Uso, Entidade-Relacionamento e Classes e seguindo então para a implementação do sistema.

3.1 Pesquisa de soluções existentes

O desenvolvimento do sistema iniciou-se pela fase de levantamento de requisitos, fundamental para compreender as necessidades do público-alvo.

Foram definidos recursos como cadastro e gerenciamento de produtos, controle de entradas e saídas de estoque, geração de relatórios, alertas de baixo estoque, categorização de produtos e ferramentas de busca e filtragem de dados. Esses elementos orientaram a estruturação do sistema, garantindo suporte à rastreabilidade dos produtos, organização dos cadastros e monitoramento das movimentações, atendendo às necessidades operacionais e estratégicas do negócio.

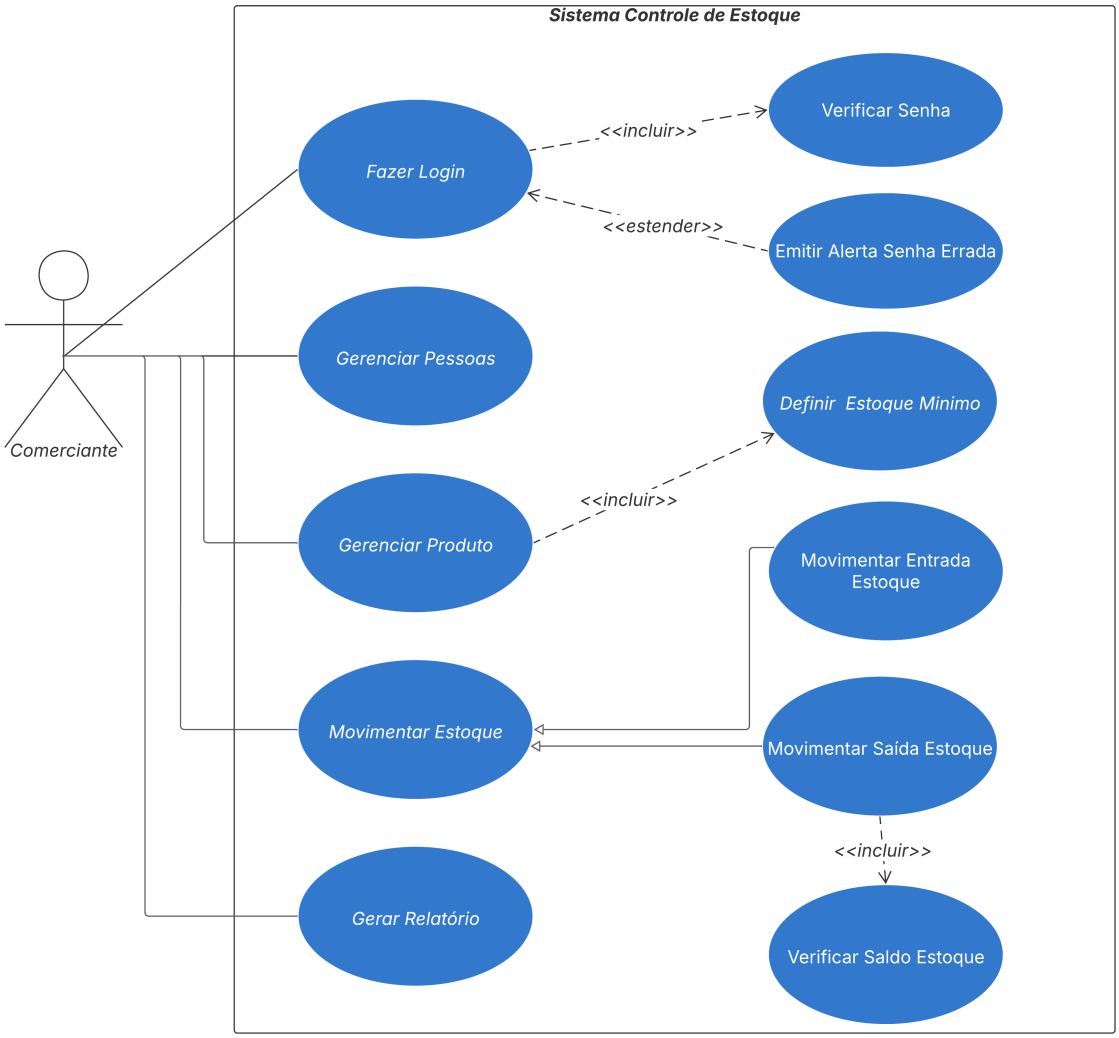
A análise dos requisitos permitiu priorizar funcionalidades de maior impacto para a gestão do estoque, assegurando que o sistema fosse desenvolvido de forma a proporcionar eficiência operacional e suporte à tomada de decisão, conforme fundamentado na literatura especializada (Ching, 2010).

3.2 Diagrama de Casos de Uso

O Diagrama de Caso de Uso é usado para identificar e descrever as interações entre os usuários (atores) e o sistema, representando as funcionalidades principais, como cadastro de produtos, registro de movimentações e geração de relatórios.

A Figura 1 identifica os principais atores e suas interações com o sistema, representando as funcionalidades essenciais para o funcionamento do sistema. O ator principal identificado no sistema é o "Comerciante", que representa o usuário final responsável pela gestão do estoque e pelas operações do dia a dia. Os casos de uso representam as funcionalidades que o sistema oferece ao "Comerciante":

Figura 1 – Diagrama de Casos de Uso do Sistema de Controle de Estoque



Fonte: Elaborado pelo autor.

- Fazer Login: permite ao Comerciante acessar o sistema de forma segura. Este caso de uso inclui a funcionalidade "Verificar Senha", pois a verificação da senha é parte integrante do processo de login e estende a funcionalidade "Emitir Alerta Senha Errada"que ocorre opcionalmente em caso de falha na verificação.

- Cadastrar Fornecedor/Clientes: permite ao "Comerciante"registrar novos fornecedores e clientes no sistema.

- Cadastrar Produto: permite ao "Comerciante"inserir novos produtos no sistema. Este caso de uso inclui a funcionalidade "Definir Estoque Mínimo", pois ao cadastrar um produto é necessário definir esse limite para exibir alerta de estoque abaixo do mínimo.

- Movimentar Estoque: representa a funcionalidade de registrar a entrada ou saída

de produtos do estoque. Este caso de uso inclui as funcionalidades "Movimentar Entrada Estoque"e "Movimentar Saída Estoque"que são as ações específicas de movimentação. Para a operação de saída de estoque é necessário "Verificar Saldo Estoque"para garantir a consistência de estoque durante a movimentação.

- Gerar Relatório: permite ao "Comerciante"gerar relatórios sobre o estoque, como saldos e verificar o histórico de movimentações.

Os relacionamentos de inclusão (*include*) indicam que um caso de uso incorpora a funcionalidade de outro caso de uso, sendo essencial para sua execução. Já os relacionamentos de extensão (*extend*) indicam que um caso de uso pode, opcionalmente, adicionar funcionalidades a outro caso de uso sob certas condições.

3.3 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve possuir para atender às necessidades dos usuários. Para o sistema de controle de estoque, foram identificados os seguintes requisitos funcionais:

- RF01: o sistema deve permitir o cadastro e a gestão de produtos.
- RF02: o sistema deve permitir o cadastro e a gestão de fornecedores e clientes.
- RF03: o sistema deve possibilitar o cadastro e a gestão de categorias de produtos, de movimentação (entrada/saída) e unidades de medida.
- RF04: o sistema deve permitir o registro de entradas e saídas de produtos, atualizando automaticamente o estoque.
- RF05: o sistema deve gerar relatórios de estoque, movimentações e produtos com estoque baixo.
- RF06: o sistema deve autenticar o usuário para garantir o acesso seguro às suas funcionalidades.

3.4 Requisitos Não Funcionais

Os requisitos não funcionais correspondem às restrições ou qualidades que definem como o sistema deve se comportar, sem especificar diretamente suas funcionalidades. Eles abrangem aspectos como desempenho, segurança, usabilidade e compatibilidade. Para este projeto, foram definidos os seguintes requisitos não funcionais:

- RNF01: o sistema não deve permitir a saída de produtos cuja quantidade em estoque seja insuficiente.
- RNF02: o sistema deve proteger os dados sensíveis dos usuários e do estoque contra acessos não autorizados, utilizando criptografia para senhas no banco de dados.

3.5 Modelagem de Dados

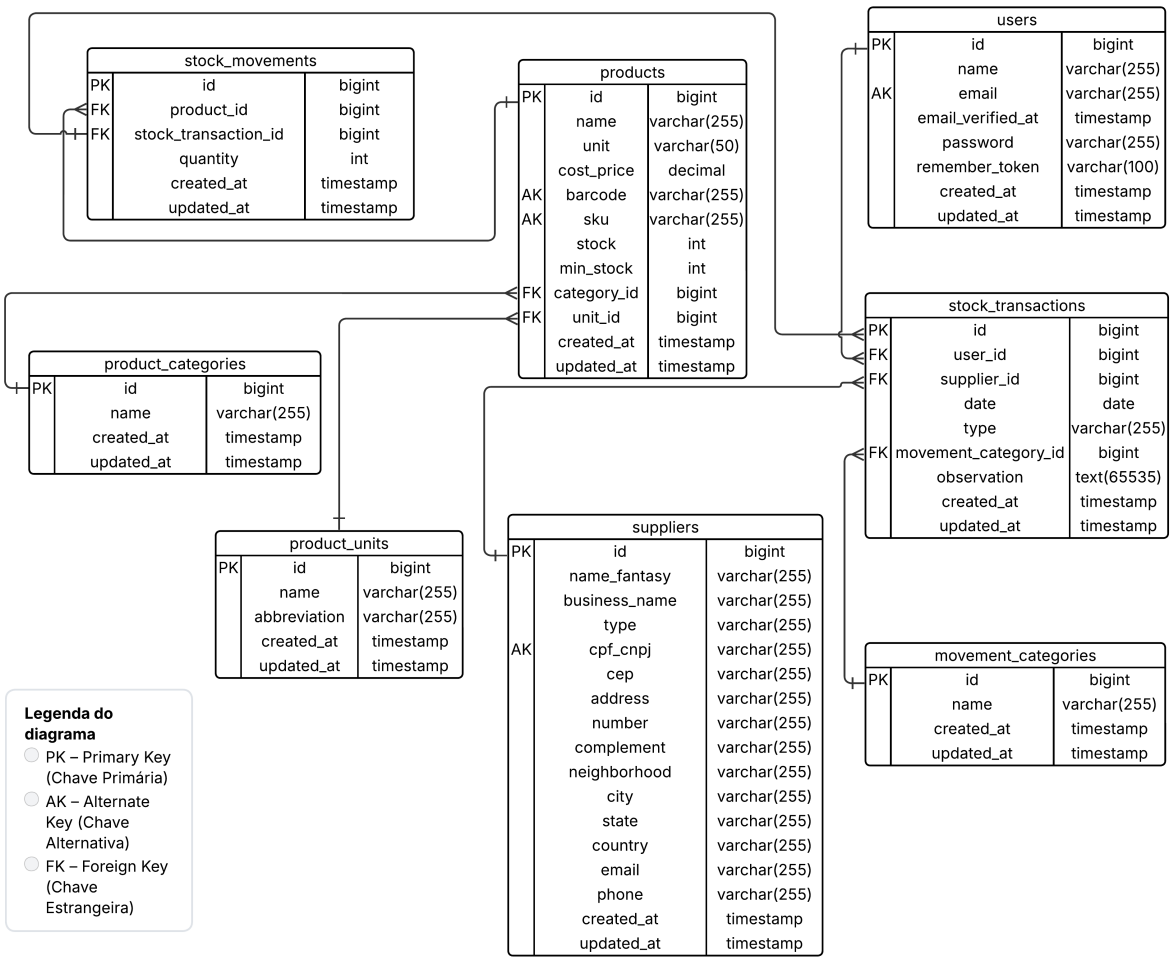
A modelagem de dados é uma etapa crucial no desenvolvimento de sistemas, pois define a estrutura lógica do banco de dados, garantindo a integridade, consistência e eficiência no armazenamento e recuperação das informações.

Para o sistema de controle de estoque, a modelagem foi realizada utilizando o paradigma relacional, com o banco de dados MySQL como sistema gerenciador. A interação com este banco de dados é facilitada pelo Eloquent ORM do Laravel, que mapeia as tabelas para modelos PHP, permitindo a manipulação dos dados de forma orientada a objetos.

O Diagrama de Entidade-Relacionamento (DER) do sistema, apresentado na Figura 2, ilustra as principais entidades e seus relacionamentos, refletindo a estrutura de dados necessária para o gerenciamento do estoque. As tabelas essenciais para o funcionamento do sistema são:

- "*products*": armazena as informações detalhadas de cada produto, como nome, preço de custo, unidade de medida, código de barras, quantidade em estoque e estoque mínimo. Possui uma chave estrangeira para a tabela "*category_id*", relacionando o produto à sua categoria.
- "*product_categories*": contém as categorias às quais os produtos pertencem, como eletrônicos, alimentos, limpeza etc.
- "*stock_transaction*": registra as transações de estoque, sejam elas de entrada ou saída. Cada transação possui a associação com um usuário, um fornecedor, data, tipo (entrada/saída) e uma categoria de movimentação.
- "*stock_movements*": detalha os produtos envolvidos em cada transação de estoque. Cada registro indica a quantidade de um produto e a associação com uma transação.
- "*suppliers*": armazena os dados dos fornecedores, incluindo nome fantasia, razão social, cnpj, endereço e contatos.
- "*movement_categories*": define as categorias de movimentação de estoque, como compra, venda, perda, devolução etc.

Figura 2 – Diagrama de Entidade-Relacionamento do Sistema



Fonte: Elaborado pelo autor.

- "*users*": contém as informações dos usuários do sistema, como nome, e-mail e senha, sendo a tabela de autenticação.

Existem algumas tabelas auxiliares como "*migrations*", "*password_reset_tokens*", "*failed_jobs*", "*jobs*", "*job_batches*", "*cache*"e "*sessions*"que são tabelas geradas automaticamente pelo *framework* Laravel para gerenciar funcionalidades internas do sistema, como migrações de banco de dados, autenticação, filas de processamento e *cache*, e não representam entidades de negócio diretamente relacionadas ao controle de estoque.

3.6 Diagramas de Classes

É utilizado para representar a estrutura estática do sistema, mostrando as classes, seus atributos, métodos e os relacionamentos entre elas, refletindo a modelagem do banco

de dados e a estrutura do Eloquent ORM.

O Diagrama de Classes do sistema de controle de estoque, apresentado na Figura 3, reflete a arquitetura MVC adotada, com uma clara separação entre as classes de controladores e as classes de modelos, que interagem com o banco de dados.

Controladores (*Controllers*)

A seção superior do diagrama ilustra as classes de controladores, responsáveis por processar as requisições dos usuários, interagir com os modelos e retornar as respostas apropriadas. Todas as classes de controladores herdam de uma classe abstrata "Controller", que fornece funcionalidades básicas. Os principais controladores do sistema são:

- "*ProductController*": gerencia as operações relacionadas aos produtos, como listagem (*index*), criação (*create*, *store*), visualização (*show*), edição (*edit*, *update*) e exclusão (*destroy*).
- "*SupplierController*": responsável pelas operações CRUD dos fornecedores.
- "*StockMovementController*": lida com o registro e gestão das movimentações de estoque.
- "*ProductCategoryController*": gerencia as categorias de produtos.
- "*MovementCategoryController*": controla as categorias de movimentação de estoque.
- "*UserController*": administra as operações relacionadas aos usuários do sistema.

Cada método nos controladores indica o tipo de retorno esperado, como "Response"para *JavaScript Object Notation* (JSON) ou "RedirectResponse"para redirecionamentos após operações de formulário.

Modelos (*Models*)

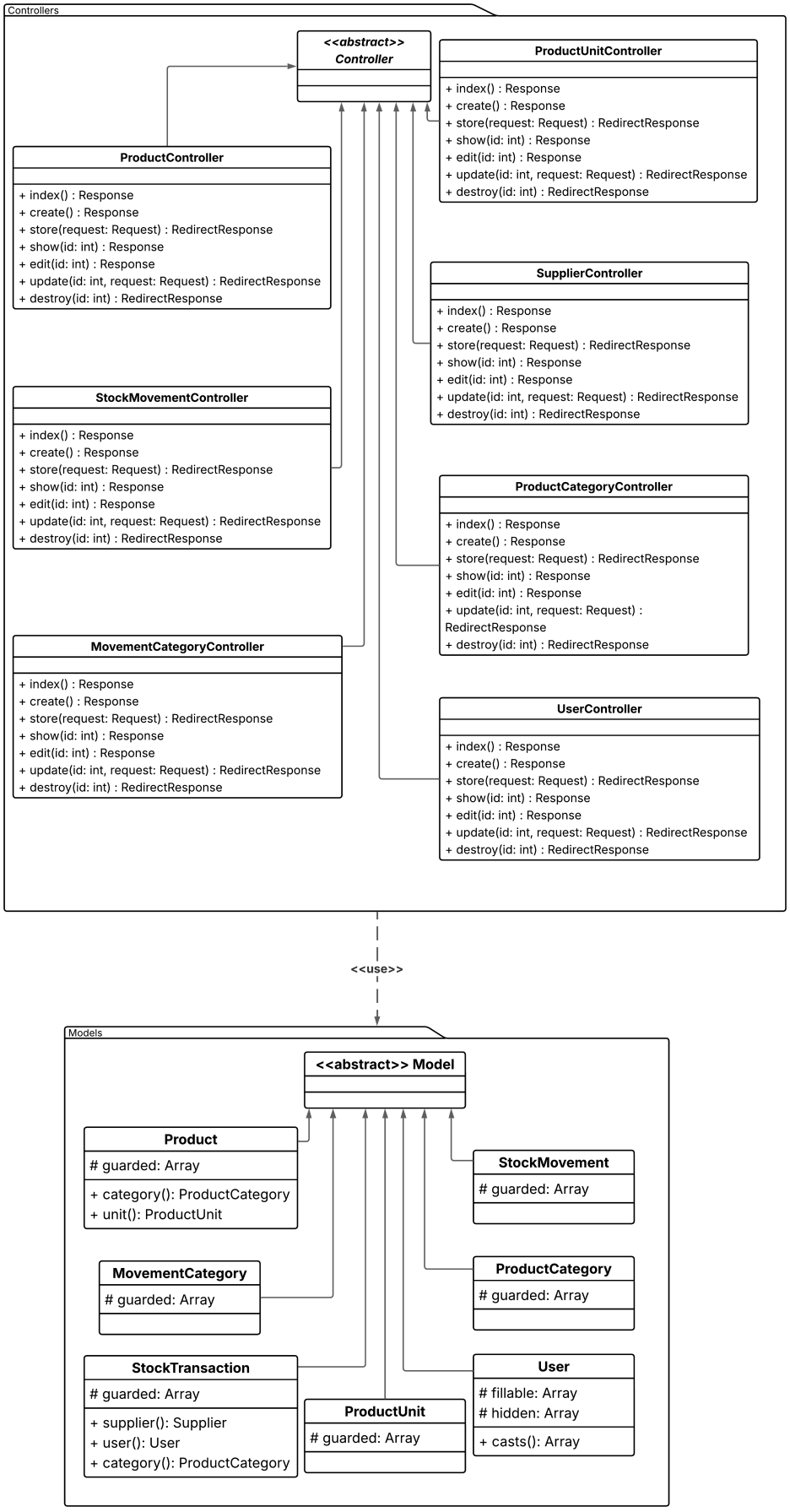
A seção inferior do diagrama apresenta as classes de modelos, que representam as entidades do banco de dados e fornecem uma interface para interagir com as tabelas correspondentes. Todas as classes de modelos herdam de uma classe abstrata "*Model*", base do Eloquent ORM do Laravel. Os principais modelos do sistema são:

- "*Product*": representa a entidade produto, com atributos como "*guarded*"(proteção contra atribuição em massa) e o método "*category*()"que define o relacionamento com "*ProductCategory*".

- "*StockMovement*": representa uma movimentação individual de estoque.
- "*MovementCategory*": representa as categorias de movimentação.
- "*ProductCategory*": representa as categorias de produtos.
- "*StockTransaction*": representa uma transação de estoque, com relacionamentos para "*Supplier*", "*User*"e "*ProductCategory*".
- "*User*": representa os usuários do sistema, incluindo atributos como "*fillable*"(atribuição em massa), "*hidden*"(ocultação de atributos na serialização) e "*casts*"(conversão de tipos).

Os relacionamentos entre os modelos, indicados pelas setas no diagrama, demonstram como as entidades estão conectadas no banco de dados, refletindo a estrutura relacional e a capacidade do Eloquent ORM de gerenciar essas associações de forma programática.

Figura 3 – Diagrama de Classes do Sistema



Fonte: Elaborado pelo autor.

4 Implementação

Este capítulo detalha o processo de implementação do sistema de controle de estoque, cujo código-fonte está disponível no GitHub (Vitor, 2025) permitindo o acesso e avaliação por outros interessados. Serão abordadas a estrutura do projeto, a criação das rotas REST, a modelagem de dados com o Eloquent ORM e a apresentação de exemplos de código relevantes.

4.1 Estrutura do Projeto em Laravel

O Laravel organiza o código de uma aplicação de forma lógica e intuitiva, seguindo o padrão MVC e outras convenções que promovem a manutenibilidade e a escalabilidade. A estrutura de diretórios padrão do Laravel é a seguinte:

- app/: contém o código-fonte principal da aplicação, incluindo *Models*, *Controllers*, *Providers* e outras classes de lógica de negócio.
  - app/Http/Controllers/: armazena os controladores que lidam com as requisições HTTP e a lógica de negócio.
  - app/Models/: contém os modelos Eloquent que representam as tabelas do banco de dados e suas interações.
- bootstrap/: contém os arquivos de inicialização do *framework*.
- config/: armazena os arquivos de configuração da aplicação.
- database/: inclui as *migrations* (para criação e modificação de tabelas), *seeders* (para popular o banco de dados) e *factories* (para gerar dados de teste).
  - database/migrations/: contém os arquivos de migração do banco de dados.
- public/: diretório raiz da web, contendo o arquivo "index.php (ponto de entrada da aplicação) e os "assets"públicos (CSS, JavaScript, imagens).
- resources/: contém as *views* (*templates* Blade), *assets* não compilados (Sass, Less, JavaScript) e arquivos de linguagem.
  - resources/views/: armazena os *templates* Blade que compõem a interface do usuário.
- routes/: define as rotas da aplicação (web, api, *console*, *channels*).

  - routes/api.php: define as rotas para a API RESTful.
  - routes/web.php: define as rotas para as páginas web tradicionais.

- storage/: contém arquivos gerados pelo *framework*, como *logs*, sessões e arquivos de cache.
- tests/: armazena os testes automatizados da aplicação.
- vendor/: contém as dependências gerenciadas pelo Composer.

Essa organização facilita a localização e a manutenção dos diferentes componentes da aplicação, promovendo um desenvolvimento mais eficiente e colaborativo.

4.2 Criação das Rotas

O Laravel simplifica significativamente a definição de rotas permitindo que os desenvolvedores mapeiem URLs de forma intuitiva para ações específicas em controladores. Para a construção do sistema de controle de estoque, foram utilizadas as rotas definidas no arquivo "routes/web.php", que gerenciam tanto as requisições de dados quanto a apresentação das interfaces ao usuário.

As rotas seguem um padrão que mapeia as operações *Create, Read, Update, Delete* (CRUD) para métodos HTTP específicos (POST, GET, PUT/PATCH, DELETE) e URLs que representam recursos, promovendo uma arquitetura de comunicação padronizada.

Para exemplificar a definição de rotas para um recurso, considere o recurso "produtos". A seguir, é apresentado o trecho de código utilizado no arquivo "routes/web.php"para a sua configuração:

Código 4.1 – Exemplo de código: Rotas Pordutos

```
1 <?php
2 use Illuminate\Http\Request;
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\ProductController;
5 Route::resource("produtos", ProductController::class);
```

O método "Route::resource"é um recurso poderoso do Laravel que gera automaticamente um conjunto completo de rotas para um determinado recurso, mapeando-as para as ações correspondentes em um controlador. Este método é o ideal para aplicações web que necessitam de rotas para a manipulação de dados e também para a exibição de formulários (como *create* e *edit*). Para o recurso "produtos", este método gera as seguintes rotas:

- GET /produtos: Exibe uma lista de todos os produtos (método *index* no "Product-Controller").
- GET /produtos/create: Exibe o formulário para criação de um novo produto (método *create/* no "ProductController").
- POST /produtos|: Armazena um novo produto no banco de dados (método *store* no "ProductController").
- GET /produtos/produto: Exibe os detalhes de um produto específico (método *show* no "ProductController").
- GET /produtos/produto/edit: Exibe o formulário para edição de um produto específico (método *edit* no "ProductController").
- PUT/PATCH /produtos/produto: Atualiza um produto específico no banco de dados (método *update* no "ProductController").
- DELETE /produtos/produto: Exclui um produto específico do banco de dados (método *destroy* no "ProductController").

Essa abordagem padronizada facilita a comunicação entre o *frontend* e o *backend*, garantindo que as operações sejam realizadas de forma consistente e previsível.

4.3 Modelagem no `Eloquent ORM`

O `Eloquent ORM` do `Laravel` proporciona uma maneira elegante e intuitiva de interagir com o banco de dados, mapeando tabelas para classes PHP chamadas Modelo. Cada modelo corresponde a uma tabela no banco de dados e permite realizar operações CRUD de forma orientada a objetos. Abaixo, um exemplo do modelo Produto e sua *migration* correspondente:

4.3.1 Modelo Produto

O modelo "Produto"representa a tabela "products"no banco de dados e é responsável por gerenciar as informações dos produtos cadastrados no sistema. Para garantir segurança contra atribuição em massa de forma indevida, a propriedade "$fillable"foi definida com os campos que podem ser preenchidos em massa, como "nome", "descricao", "preco"e "categoria_id".

Além disso, o modelo estabelece o relacionamento com a tabela "product_categories"por meio do método "category()", que utiliza a associação "belongsTo", permitindo vincular

cada produto a uma categoria específica e facilitar consultas relacionadas. O código 4.2 exemplifica a implementação do modelo, definindo a configuração básica necessária para garantir a segurança na atribuição de dados e a correta organização dos relacionamentos dentro do sistema.:

Código 4.2 – Exemplo de código Model Produtos

```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Product extends Model {
    use HasFactory;
    protected $fillable = [
        "name",
        "unit",
        "cost_price",
        "barcode",
        "sku",
        "stock",
        "min_stock",
        "category_id"
    ];
    public function category() {
        return $this->belongsTo(ProductCategory::class);
    }
}
```

4.3.2 Migration para a Tabela Produtos

A *migration* é um recurso do Laravel que permite gerenciar, de forma versionada, o esquema do banco de dados. A seguir, apresenta-se a *migration* responsável pela criação da tabela "products":

Código 4.3 – Exemplo de código: Migration Produtos

```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use App\Models\ProductCategory;
class CreateProductsTable extends Migration {
    public function up(): void {
```

```
8           Schema::create("products", function (Blueprint $table) {
9               $table->id();
10              $table->string("name");
11              $table->string("unit", 50);
12              $table->decimal("cost_price", 10, 2);
13              $table->string("barcode")->unique();
14              $table->string("sku")->unique();
15              $table->integer("stock")->default(0);
16              $table->integer("min_stock")->default(0);
17              $table->foreignIdFor(ProductCategory::class, '
                    category_id');
18              $table->timestamps();
19          });
20      }
21      public function down(): void {
22          Schema::dropIfExists("products");
23      }
24  }
```

- Schema::create("products", function (Blueprint $table) { ... }): Cria a tabela `products` no banco de dados com as colunas definidas dentro da função.
- $table->id(): Cria a coluna "id"como chave primária auto-incrementada.
- $table->string("name"): Coluna para armazenar o nome do produto, tipo texto.
- $table->string("unit", 50): Coluna para a unidade de medida do produto, com limite de 50 caracteres.
- $table->decimal("cost_price", 10, 2): Coluna para o preço de custo, número decimal com até 10 dígitos no total e 2 casas decimais.
- $table->string("barcode")->unique(): Coluna para o código de barras, com restrição de unicidade para evitar valores duplicados.
- $table->string("sku")->unique(): Coluna para o SKU (Stock Keeping Unit), também com restrição de unicidade.
- $table->integer("stock")->default(0): Quantidade atual em estoque, com valor padrão 0.
- $table->integer("min_stock")->default(0): Estoque mínimo configurado para alerta, com valor padrão 0.

- $table->foreignIdFor(ProductCategory::class, "category_id"): Cria a coluna `category_id` como chave estrangeira, referenciando a tabela de categorias dos produtos (product_categories).
- $table->timestamps(): Cria automaticamente as colunas "created_at"e "updated_at", que armazenam as datas de criação e atualização dos registros.

4.4 Controladores (*Controllers*)

No Laravel, os controladores são responsáveis por gerenciar as requisições feitas ao sistema, centralizando a lógica que interage com os modelos e as *views*. Eles organizam o fluxo de dados entre a interface do usuário e o banco de dados, tornando o código mais modular e fácil de manter.

No contexto deste sistema de controle de estoque, os controladores são responsáveis por ações como cadastrar, editar, listar e excluir produtos, bem como gerenciar categorias e movimentações de estoque. O código 4.4 exemplifica algumas ações de um controlador para produtos:

Código 4.4 – Exemplo de Controller para produtos

```
<?php
class ProdutoController extends Controller {
    public function index() {
        $produtos = Product::all();
        return view("produtos.index", compact("produtos"));
    }
    public function create() {
        return view("produtos.create");
    }
    public function store(Request $request) {
        Product::create($request->all());
        return redirect()->route("produtos.index");
    }
}
```

- public function index(): Recupera todos os registros da tabela "products"por meio do método "Product::all()"e os armazena na variável "$produtos". Em seguida, retorna a *view* "produtos.index", passando os dados para exibição na interface do usuário. Essa ação corresponde à operação *read* do CRUD.

- public function create(): Retorna a *view* "produtos.create", que contém o formulário para o cadastro de um novo produto. Esta ação corresponde à etapa inicial da operação *create* do CRUD, onde o usuário insere as informações.

- `public function store(Request $request)`: Recebe os dados enviados pelo formulário de criação através do objeto "Request". Utiliza o método "Product::create()"para inserir um novo registro na tabela "products", com base nos campos definidos no atributo "$fillable"do modelo. Ao final, redireciona o usuário para a rota "produtos.index", exibindo a listagem atualizada. Esta ação representa a finalização da operação *create* do CRUD.

4.5 Views

As *Views* compõem a camada de apresentação do sistema, responsáveis por exibir a interface para o usuário. No Laravel, as *Views* são construídas utilizando o Blade, que é o sistema de *templates* nativo do *framework*. Ele permite a utilização de diretivas e componentes que facilitam a organização e reutilização do código HTML.

Para a estilização, o projeto adotou o Tailwind CSS, uma biblioteca utilitária de classes CSS que possibilita a criação rápida e eficiente de interfaces responsivas e modernas. O uso do Tailwind CSS aliado ao Blade promove uma separação clara entre lógica, estrutura e estilo, garantindo que a interface seja flexível, escalável e visualmente atraente (Laravel Framework, 2025; Tailwind CSS, 2024).

Este capítulo demonstrou como o Laravel, com sua estrutura MVC, rotas RESTful e o poderoso Eloquent ORM, facilita o desenvolvimento de aplicações web robustas e organizadas. Os exemplos de código ilustram a aplicação prática desses conceitos na construção do sistema de controle de estoque.

5 Resultados e Discussão

Este capítulo apresenta os resultados obtidos com o desenvolvimento do sistema de controle de estoque, discute os benefícios práticos observados e o potencial de aplicação futura da solução.

5.1 Resultados

O sistema de controle de estoque, desenvolvido em PHP com o *framework* Laravel, demonstrou ser funcional e aderente aos objetivos propostos. Todas as funcionalidades essenciais para o gerenciamento de estoque foram implementadas com sucesso, sendo desenvolvidas para serem intuitivas e de fácil preenchimento. Além disso, sua eficiência e usabilidade foram avaliadas e comprovadas por meio de uma pesquisa realizada com alguns usuários que utilizaram o sistema, confirmando que ele atende às necessidades operacionais e estratégicas do público-alvo, conforme detalhado a seguir:

- Cadastro e Gerenciamento de Fornecedores/Clientes: Permite cadastros de novos fornecedores/clientes com informações detalhadas de dados cadastrais, assim como a edição e exclusão de registros existentes. A funcionalidade foi ilustrada na Figura 7.

- Cadastro e Gerenciamento de Produtos: Conforme ilustrado na Figura 8, permite o registro de novos produtos com informações detalhadas (nome, descrição, preço, quantidade em estoque e categoria), bem como a edição e exclusão de registros existentes.

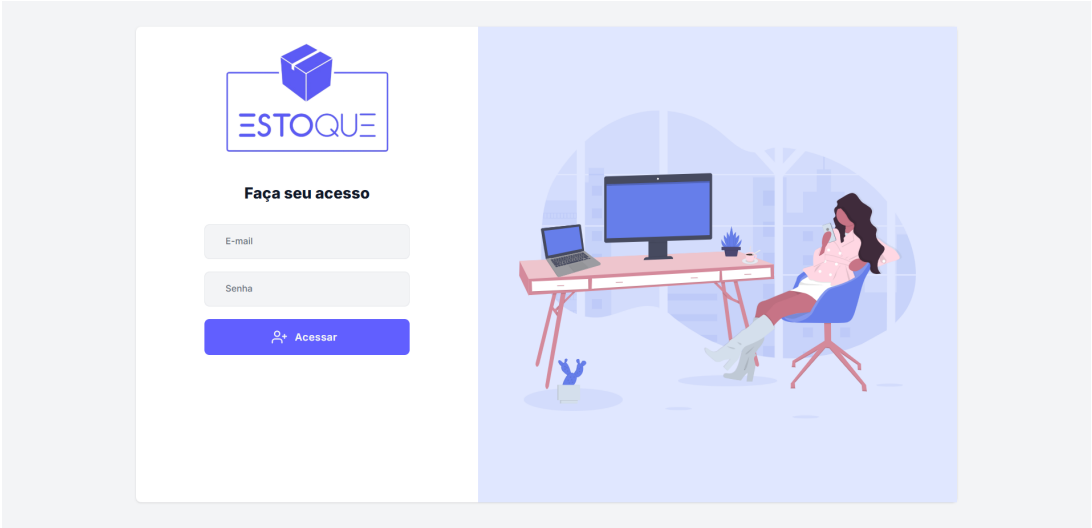
- Cadastro e Gerenciamento de Categorias: Possibilita a organização dos produtos por categorias, facilitando a gestão de dados. Esta funcionalidade é ilustrada na Figura 9.

- Registro de Movimentações (Entrada e Saída): O sistema possibilita o registro detalhado das movimentações de entrada e saída de itens do estoque. Essa funcionalidade inclui a atualização automática da quantidade disponível e a associação da movimentação a uma categoria específica (como compra, venda ou perda), garantindo um controle preciso do fluxo de produtos. Este recurso é ilustrado na Figura 10.

- Controle de Usuários: Implementação de um sistema básico de gerenciamento de usuários, acompanhado de uma interface de autenticação, garantindo acesso controlado ao sistema. Essa funcionalidade está ilustrada nas Figuras 6 e 4.

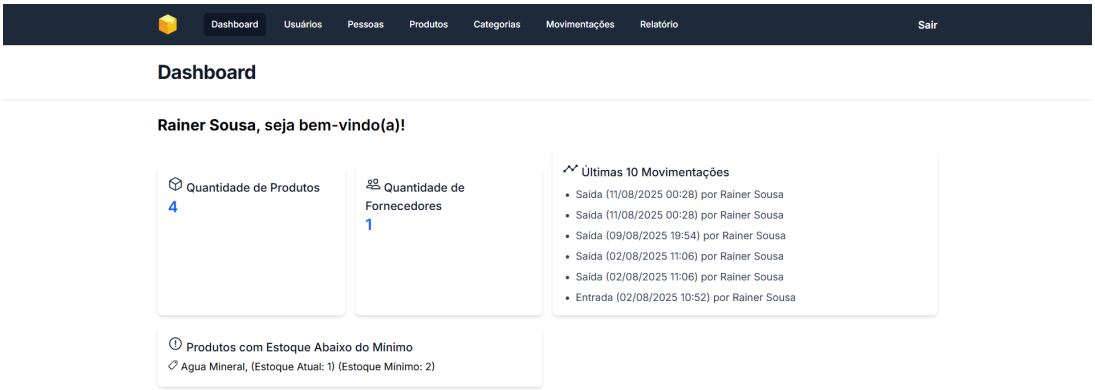
- Geração de Relatórios Simples: Capacidade de visualizar o estoque atual, movimentações recentes e outras informações relevantes para a tomada de decisão. Esta funcionalidade é ilustrada na Figura 11

Figura 4 – Tela de Controle de Acesso ao Sistema



Fonte: Elaborado pelo autor.

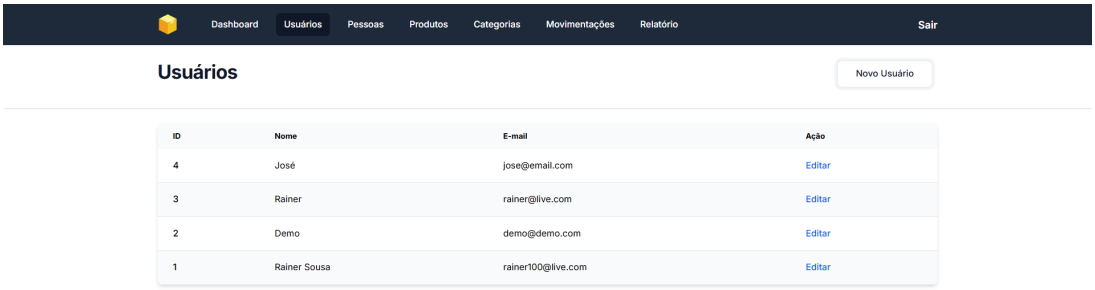
Figura 5 – Tela com Painel Inicial



Fonte: Elaborado pelo autor.

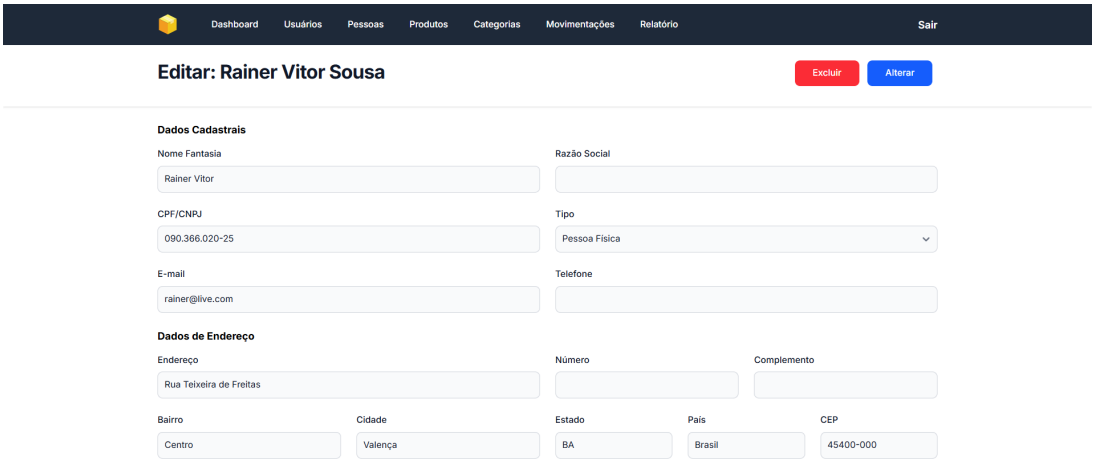
A avaliação do sistema, realizada junto a 34 usuários, evidencia que os objetivos específicos propostos foram atingidos. A primeira meta, de prover uma interface intuitiva para cadastro e gerenciamento de produtos, fornecedores/clientes e categorias, foi confirmada pelo fato de que 79,4% dos respondentes avaliaram a interface como totalmente intuitiva e fácil de usar, e 88,2% conseguiram compreender o processo de cadastro de produtos em menos de cinco minutos. Além disso, a organização visual da interface contribuiu significativamente para a usabilidade, com 82,4% dos usuários afirmando que encontraram rapidamente as funções necessárias.

Figura 6 – Tela de Listagem de Usuários



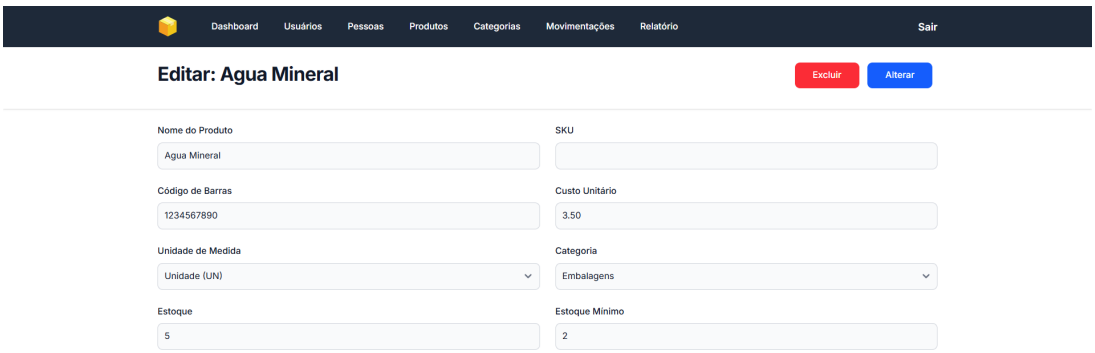
Fonte: Elaborado pelo autor.

Figura 7 – Tela de Edição de Fornecedores/Clientes



Fonte: Elaborado pelo autor.

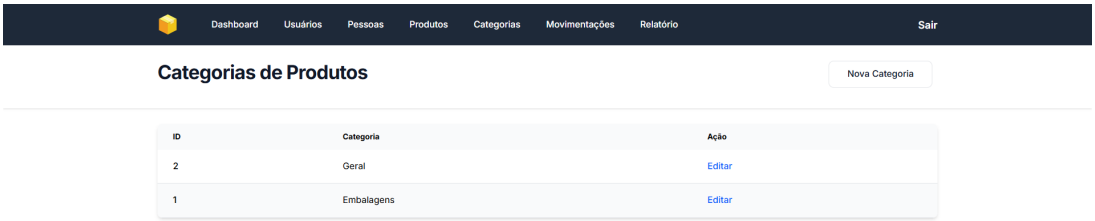
Figura 8 – Tela de Edição de Produtos



Fonte: Elaborado pelo autor.

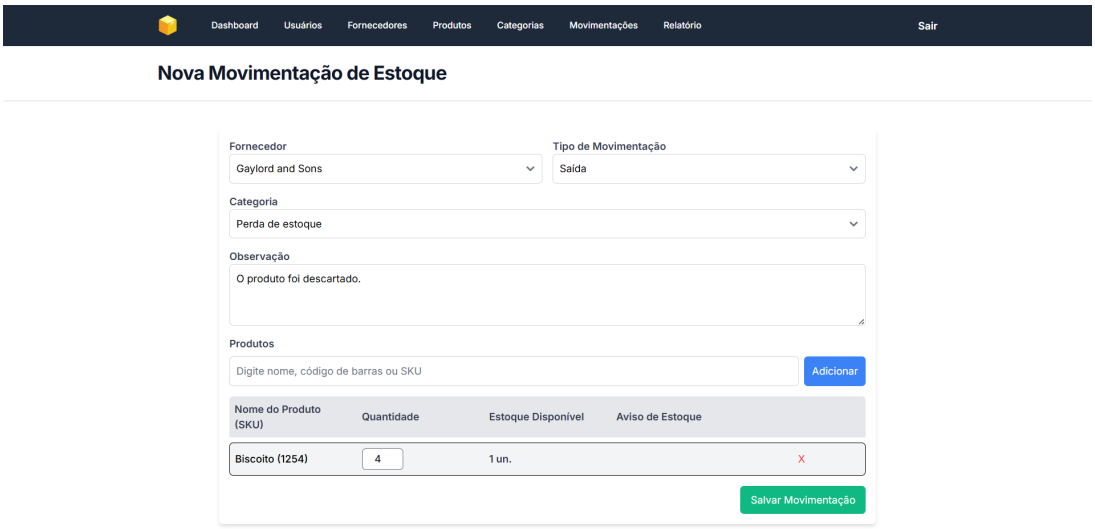
A segunda meta, focada em oferecer funcionalidades de apoio à decisão gerencial, foi plenamente alcançada, como evidenciado pela avaliação dos relatórios. Com 91,2% dos usuários concordando (notas 4 e 5) que os relatórios gerados são úteis para a tomada de decisões, a funcionalidade se prova um dos pilares do sistema. Esse alto índice de

Figura 9 – Tela de Gerenciamento de Categorias



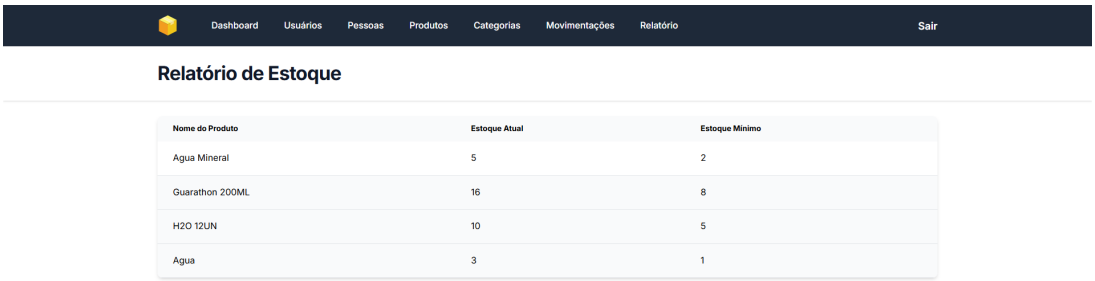
Fonte: Elaborado pelo autor.

Figura 10 – Tela de Movimentação de Estoque



Fonte: Elaborado pelo autor.

Figura 11 – Tela de Relatório de Estoque



Fonte: Elaborado pelo autor.

aprovação, com quase dois terços dos participantes (64,7%) concedendo a nota máxima, valida a abordagem de simplicidade e clareza adotada, confirmando que o sistema entrega informações de valor para a gestão estratégica do estoque.

A avaliação geral do sistema foi altamente positiva, com 79,4% dos usuários classificando-o como excelente e 88,2% demonstrando disposição em recomendá-lo a pequenas empresas, o que indica a aceitação e relevância do sistema para o público-alvo. O questionário utilizado encontra-se disponível no Apêndice A.

No que tange a escalabilidade e a manutenibilidade de sistemas de informação, esses conceitos estão diretamente relacionadas à adoção de padrões de arquitetura modernos e tecnologias robustas. Segundo Softtech (2023), práticas como arquitetura modular, separação de responsabilidades através do padrão MVC e otimizações de desempenho — como *cache*, indexação e filas de processamento assíncrono — permitem que aplicações possam crescer de forma sustentável, mantendo estabilidade e eficiência.

Nesse contexto, a escolha do *framework* Laravel se mostrou estratégica. Ele adota nativamente o padrão MVC, favorecendo a organização do código e a separação clara entre camadas de dados, regras de negócio e apresentação. Essa abordagem facilita tanto a correção de falhas quanto a implementação de novas funcionalidades, sem comprometer módulos já existentes (Design, 2024).

Além disso, o Laravel oferece suporte a arquiteturas modulares e até mesmo microsserviços, permitindo que funcionalidades específicas sejam escaladas independentemente, conforme a demanda. Essa característica é amplamente utilizada em sistemas corporativos que necessitam de tolerância a falhas e distribuição de carga (Solution, 2024).

O sistema incorpora boas práticas de segurança: o Laravel utiliza por padrão o algoritmo Bcrypt para o *hash* de senhas, um processo que transforma a senha original em uma sequência criptográfica irreversível, impedindo que seja lida mesmo que o banco de dados seja comprometido. Esse método é amplamente adotado no mercado por sua resistência a ataques de força bruta e por permitir aumentar essa dificuldade conforme as capacidades de hardware evoluem, mantendo a proteção sempre eficaz (Laravel, 2025a).

Dessa forma, a arquitetura adotada neste projeto garante não apenas o atendimento às necessidades atuais das pequenas empresas, mas também possibilita expansão e manutenção contínua, confirmando o alcance do objetivo específico relacionado à escalabilidade e manutenibilidade.

5.2 Potencial de Aplicação Futura

O sistema de controle de estoque desenvolvido possui um grande potencial para futuras expansões e melhorias, que podem agregar ainda mais valor às pequenas empresas e ampliar seu escopo de atuação:

- Módulo de Vendas (PDV): Desenvolvimento de um Ponto de Venda integrado para facilitar o registro automático das vendas e a saída de estoque, promovendo maior agilidade e eficiência no fluxo de trabalho.
- Módulos de Compras e Fornecedores: Inclusão de funcionalidades para gerenciar pedidos de compra, fornecedores e recebimento de mercadorias, completando o ciclo

de gestão de estoque.

- Relatórios Avançados e *Dashboards*: Implementação de relatórios detalhados e *dashboards* interativos com gráficos e indicadores de desempenho para análises aprofundadas.
- Notificações e Alertas: Sistema de notificações para alertar sobre estoque baixo, produtos próximos da validade ou outras condições críticas.
- Interface de Usuário Aprimorada: Otimização da interface para garantir usabilidade em diversos dispositivos, incluindo *smartphones* e *tablets*, ampliando o acesso e a flexibilidade do sistema.

Em resumo, o sistema demonstrou ser uma solução eficaz e viável para as necessidades das pequenas empresas, oferecendo uma base sólida para futuras evoluções e adaptações.

6 Considerações Finais

Este Trabalho de Conclusão de Curso teve como objetivo principal o desenvolvimento de um sistema web de controle de estoque para pequenas empresas, utilizando PHP e o *framework* Laravel. Durante o estudo, buscou-se apresentar uma solução tecnológica capaz de mitigar os desafios do controle manual de estoque, caracterizado pela propensão a erros e prejuízos financeiros.

Com base nos resultados obtidos, pode-se afirmar que os objetivos foram plenamente alcançados. O sistema desenvolvido mostrou-se funcional e eficaz, permitindo o gerenciamento de produtos, fornecedores/clientes, categorias e, fundamentalmente, o registro das movimentações de entrada e saída de estoque, com atualização automática das quantidades disponíveis. A implementação de relatórios básicos e do sistema de autenticação reforça a capacidade da aplicação em garantir um controle preciso e seguro.

As escolhas tecnológicas adotadas, como a arquitetura MVC do Laravel, o uso do Eloquent ORM para interação com MySQL e a padronização das rotas, mostraram-se adequadas para a construção de uma solução robusta e de fácil manutenção. Apesar dos desafios inerentes ao desenvolvimento, o sistema atende às funcionalidades essenciais, oferecendo benefícios práticos relevantes para o público-alvo.

Em síntese, o sistema desenvolvido representa uma contribuição valiosa para a otimização da gestão em pequenas empresas, promovendo a redução de erros, a melhoria da eficiência operacional e o suporte à tomada de decisões estratégicas. Este projeto valida a aplicação dos conhecimentos adquiridos durante a formação em Análise e Desenvolvimento de Sistemas na resolução de problemas reais de mercado.

Referências

AMAZONAS ATUAL. **Pequenas empresas ainda usam caderno para controle de recursos**. Acesso em: 03 ago. 2025. Jan. 2025. Disponível em: <https://amazonasatual.com.br/pequenos-empresas-ainda-usam-caderno-para-controle-de-recursos/>.

CHING, Hong Yuh. **Gestão de estoques na cadeia de logística integrada – Supply Chain**. 4. ed. São Paulo: Atlas, 2010.

DESIGN, Prateeksha Web. **Benefits of Laravel Web App Development**. Acesso em: 09 ago. 2025. 2024. Disponível em: <https://prateeksha.com/blog/benefits-of-laravel-web-app-development>.

DEVMEDIA. **ORM (Object Relational Mapper)**. 2025. Disponível em: <https://www.devmedia.com.br/orm-object-relational-mapper/19056>.

DIO. **Introdução ao Desenvolvimento Web: Conceitos Essenciais e Tecnologias Fundamentais**. 2025. Disponível em: <https://www.dio.me/articles/introducao-ao-desenvolvimento-web-conceitos-essenciais-e-tecnologias-fundamentais>.

DOCS, MDN Web. **Introdução à Web - Aprendendo desenvolvimento web**. 2025. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Getting_started/Your_first_website>.

_____. **MVC - Glossary**. 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>.

FIELDING, R. et al. **RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax**. 2005. https://www.rfc-editor.org/rfc/rfc3986. Acesso em: 15 ago. 2025.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Ph.D. Thesis – University of California, Irvine, Irvine, CA, USA. Acesso em: 09 ago. 2025. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

KINSTA. **O que é Arquitetura de Aplicativos Web? Quebrando um...** 2025. Disponível em: <https://kinsta.com/pt/blog/arquitetura-aplicativos-web/>.

LARAVEL. **Eloquent: Getting Started - Laravel 12.x**. Disponível em: <https://laravel.com/docs/12.x/eloquent>.

_____. **Hashing - Laravel Documentation**. Acesso em: 10 ago. 2025. 2025. Disponível em: <https://laravel.com/docs/11.x/hashing>. Acesso em: 10 ago. 2025.

LARAVEL. **The PHP Framework For Web Artisans**. 2025. Disponível em: <https://laravel.com/do>.

LARAVEL FRAMEWORK. **Blade Templating**. 2025. https://laravel.com/docs/10.x/blade. Acesso em: 08 ago. 2025.

ORACLE. **O que é um Banco de Dados Relacional? (RDBMS)?** 2021. Disponível em: <https://www.oracle.com/br/database/what-is-a-relational-database/>.

SEBRAE. **Perfil das Micro e Pequenas Empresas Brasileiras**. 2022. Acesso em: 03 ago. 2025. Disponível em: <https://sebraepr.com.br/impulsiona/perfil-das-mpes-no-brasil-insights-estrategicos-para-2022>.

SEBRAE; GARTNER. **PMEs: 76% não possuem nenhum sistema para a gestão dos negócios**. Acesso em: 18 jul. 2025. Jun. 2024. Disponível em: <https://www.contabeis.com.br/noticias/65816/pmes-76-nao-possuem-nenhum-sistema-para-a-gestao-dos-negocios/>.

SOFTTECH, Acquaint. **Strategies for Scalable Laravel Solutions**. Acesso em: 09 ago. 2025. 2023. Disponível em: <https://acquaintsoft.com/blog/strategies-for-scalable-laravel-solutions>.

SOLUTION, AddWeb. **Laravel Microservices – A Scalability Blueprint**. Acesso em: 09 ago. 2025. 2024. Disponível em: <https://www.addwebsolution.com/blog/laravel-microservices>.

TAILWIND CSS. **Utility-First CSS Framework**. 2024. https://tailwindcss.com/. Acesso em: 08 ago. 2025.

VITOR, Rainer. **Sistema de Controle de Estoque Simples**. 2025. Acesso em: 02 ago. 2025. Disponível em: <https://github.com/rainervitorrv/stock/>.

W3SCHOOLS. **PHP Introduction**. 2023. Disponível em: <https://www.w3schools.com/php/php_intro.asp>.

Apêndices

APÊNDICE A – Formulário de Avaliação do Sistema de Controle de Estoque Web

E-mail: _____

Seção 1 – Perfil do Participante

- Qual é o seu nível de familiaridade com sistemas de controle de estoque?
  - Nenhuma
  - Baixa
  - Média
  - Alta
- Qual é o seu papel na empresa?
  - Proprietário
  - Gerente
  - Colaborador operacional
  - Outros

Seção 2 – Avaliação da Interface e Usabilidade

- O sistema apresentou uma interface intuitiva e fácil de usar para o cadastro e gerenciamento de produtos e categorias?
  Escala de 1 a 5 (1 = Discordo totalmente, 5 = Concordo totalmente)
  – 1 – 2 – 3 – 4 – 5
- Quanto tempo, em média, você levou para entender como cadastrar um produto?
  - Menos de 5 minutos
  - Entre 5 e 10 minutos
  - Mais de 10 minutos
- A organização visual da interface ajudou a encontrar rapidamente as funções necessárias?
  Escala de 1 a 5 (1 = Discordo totalmente, 5 = Concordo totalmente)
  – 1 – 2 – 3 – 4 – 5

- O controle de acesso com usuário e senha garante que apenas pessoas autorizadas possam acessar as informações do sistema?
  Escala de 1 a 5 (1 = Discordo totalmente, 5 = Concordo totalmente)
  – 1 – 2 – 3 – 4 – 5

- Os relatórios gerados contribuem para a tomada de decisões relacionadas à gestão de estoque?
  Escala de 1 a 5 (1 = Discordo totalmente, 5 = Concordo totalmente)
  – 1 – 2 – 3 – 4 – 5

- Você sugeriria alguma melhoria para os relatórios? _____
  _____

Seção 3 – Avaliação Geral

- Em uma escala de 1 a 5, como você avalia o sistema como um todo?
  - Excelente
  - Bom
  - Regular
  - Ruim
  - Péssimo

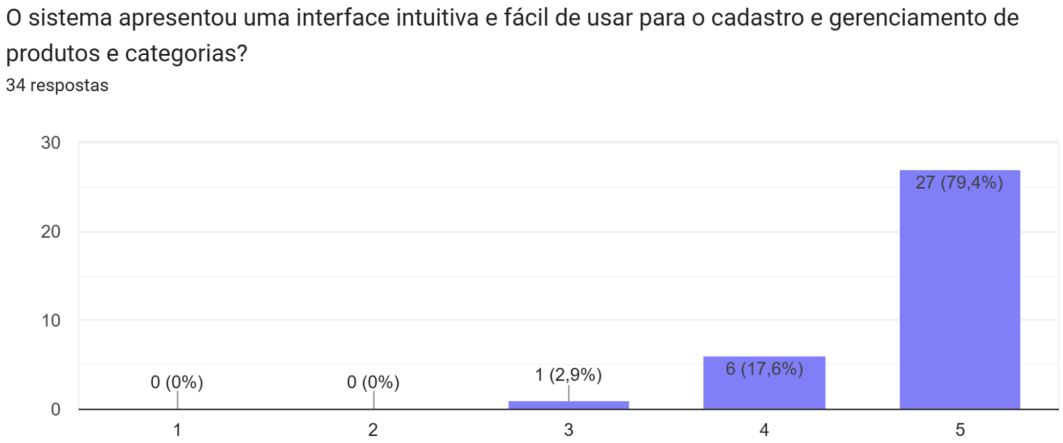
- Você recomendaria este sistema para pequenas empresas?
  - Sim
  - Talvez
  - Não

- Deixe seu comentário ou sugestão: _____
  _____
  _____

.1 Resultados da Avaliação

Nesta seção, apresentam-se os gráficos gerados a partir das respostas dos usuários no questionário de avaliação do sistema de controle de estoque. Cada gráfico ilustra aspectos específicos da usabilidade, funcionalidades e satisfação geral dos participantes.

Figura 12 – Avaliação da Intuitividade da Interface para Cadastro e Gerenciamento



Fonte: Elaborado pelo autor.

Conforme apresentado na Figura 12, 79,4% dos participantes avaliaram a interface do sistema como totalmente intuitiva e fácil de usar para o cadastro e gerenciamento de produtos e categorias. Este resultado confirma que a preocupação com a usabilidade foi eficaz e contribui para a rápida adaptação dos usuários ao sistema.

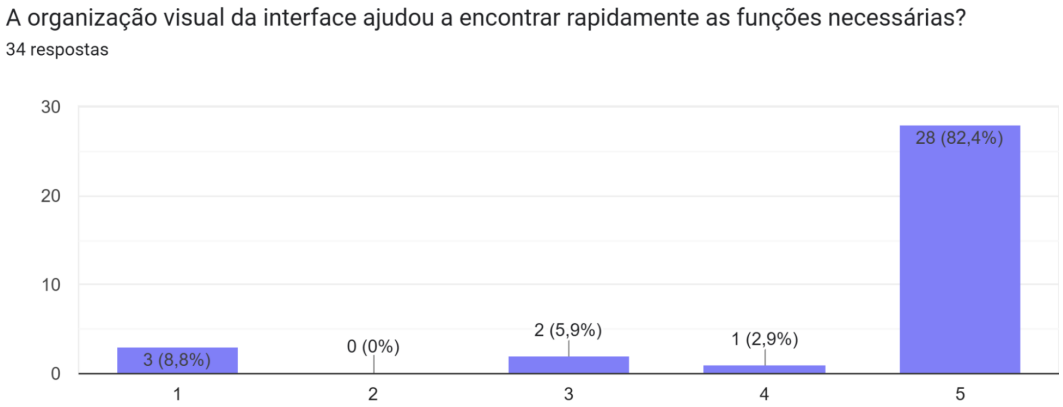
Figura 13 – Tempo Médio para Compreensão do Cadastro de Produtos

Fonte: Elaborado pelo autor.

A Figura 13 demonstra que 88,2% dos usuários compreenderam o processo de cadastro de produtos em menos de cinco minutos, evidenciando a simplicidade e clareza da interface e dos fluxos de trabalho implementados.

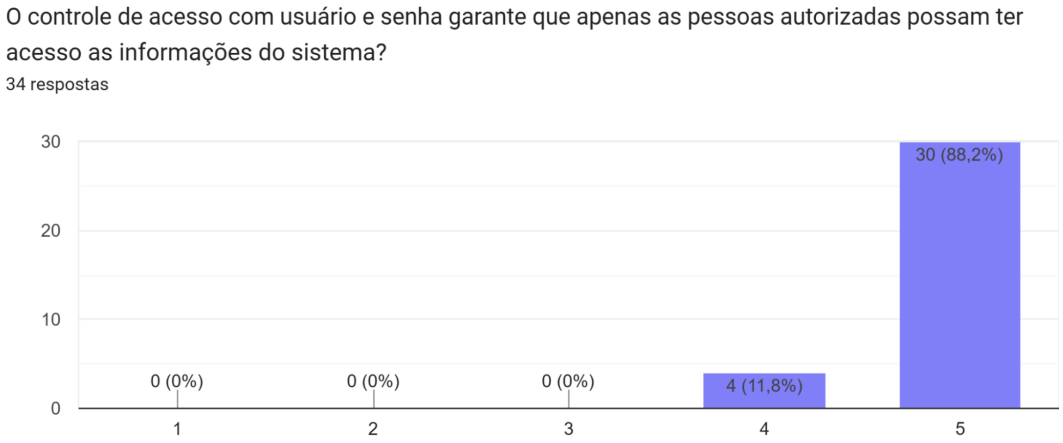
Observa-se na Figura 14 que 82,4% dos participantes consideraram que a organização visual da interface facilitou a localização das funções necessárias, reforçando a eficácia do design adotado.

Figura 14 – Avaliação da Organização Visual da Interface



Fonte: Elaborado pelo autor.

Figura 15 – Avaliação do Controle de Acesso por Usuário e Senha



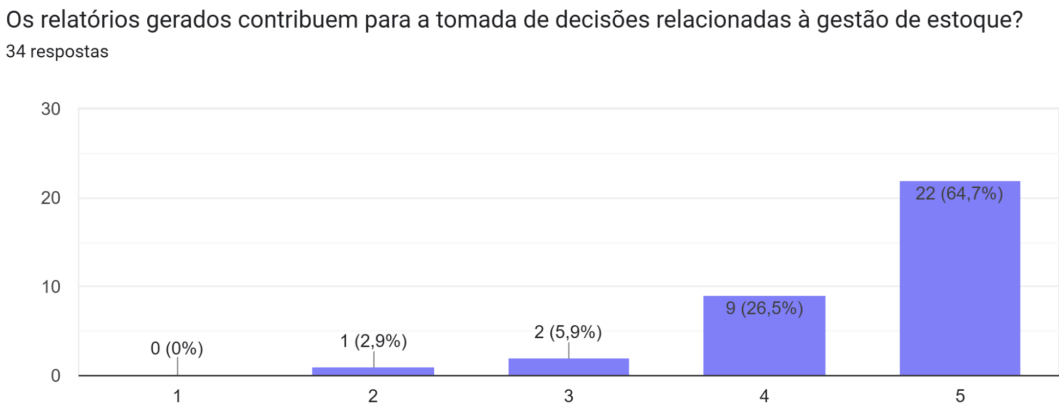
Fonte: Elaborado pelo autor.

Conforme ilustrado na Figura 15, 88,2% dos respondentes consideraram que o sistema oferece segurança adequada para o controle de acesso, garantindo proteção às informações armazenadas.

A Figura 16 revela que 91,2% dos usuários concordam (notas 4 e 5) que os relatórios gerados são úteis para suportar a tomada de decisões gerenciais, indicando espaço para futuras melhorias, mas confirmando a funcionalidade básica da ferramenta.

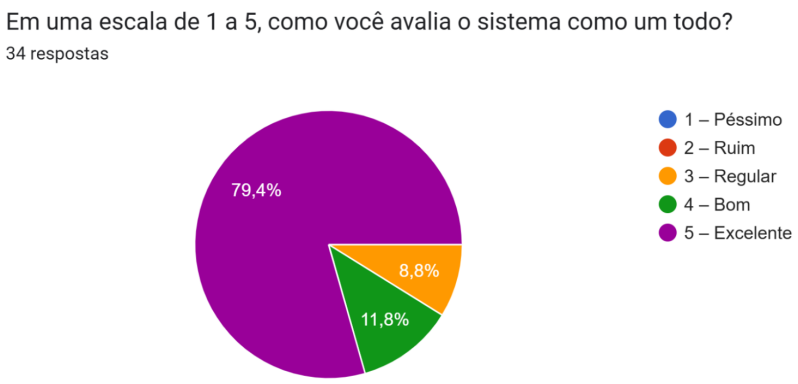
Finalmente, na Figura 17, observa-se que 79,4% dos participantes classificaram o sistema como excelente, demonstrando alta satisfação geral e potencial de aceitação no mercado-alvo.

Figura 16 – Avaliação da Utilidade dos Relatórios para a Tomada de Decisão



Fonte: Elaborado pelo autor.

Figura 17 – Avaliação Geral do Sistema



Fonte: Elaborado pelo autor.