INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS PICOS

CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ÍTALO PAIXÃO GOMES

RELATÓRIO TÉCNICO DO DESENVOLVIMENTO DE UMA PLATAFORMA WEB WHITE LABEL DE E-COMMERCE

PICOS

2026

ÍTALO PAIXÃO GOMES

RELATÓRIO TÉCNICO DO DESENVOLVIMENTO DE UMA PLATAFORMA WEB WHITE LABEL DE E-COMMERCE

Trabalho de Conclusão de Curso Monografia apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador(a): Prof. Mestre Ciro Ferreira de Carvalho Junior

PICOS
2026

ÍTALO PAIXÃO GOMES

RELATÓRIO TÉCNICO DO DESENVOLVIMENTO DE UMA PLATAFORMA WEB WHITE LABEL DE E-COMMERCE

Trabalho de Conclusão de Curso Monografia apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Aprovada em: 24 / 02 / 2026.

BANCA EXAMINADORA

____

Prof. Me. Ciro Ferreira de Carvalho Junior (Orientador)
Instituto Federal do Piauí (IFPI)

____

Prof. Me. Juciê Xavier da Silva
Instituto Federal do Piauí (IFPI)

____

Prof. Esp. Denis Costa Paiva
Instituto Federal do Piauí (IFPI)

AGRADECIMENTOS

Expresso minha gratidão a todos os docentes e servidores do IFPI – Campus Picos, que, direta ou indiretamente, colaboraram para a viabilização deste trabalho. Agradeço, imensamente, à minha família pelo suporte incondicional em cada etapa desta jornada. Acima de tudo, agradeço a Deus, por ser meu refúgio e renovar minhas forças nos momentos de maior dificuldade.

RESUMO

A crescente transformação digital consolidou o comércio eletrônico como um canal fundamental para a economia, mas a complexidade técnica e os custos elevados ainda representam barreiras significativas para a entrada de micro e pequenos empreendedores neste mercado. Diante desse cenário, este trabalho apresenta o desenvolvimento de uma plataforma de *e-commerce white label*, projetada para ser personalizável e atender às necessidades de pequenos comerciantes. O objetivo central consiste em disponibilizar uma aplicação *web* que permita o cadastro de produtos, controle de estoque, gestão de usuários e realização de vendas integradas, democratizando o acesso a ferramentas tecnológicas de varejo. Para a construção da solução, utilizou-se a linguagem TypeScript aliada aos *frameworks* Next.js e Node.js, com persistência de dados em PostgreSQL, visando garantir desempenho e escalabilidade. Os resultados obtidos demonstram a viabilidade técnica de um sistema que simplifica a gestão de lojas virtuais, permitindo configurações visuais e operacionais distintas para cada lojista. Conclui-se que a solução proposta contribui efetivamente para a inclusão digital de pequenos negócios, oferecendo uma alternativa funcional, de baixo custo e tecnicamente acessível para a expansão comercial no ambiente *online*.

Palavras-chave: *e-commerce*, Varejo, Plataforma White Label, Web

ABSTRACT

The growing digital transformation has consolidated e-commerce as a fundamental channel for the economy. However, technical complexity and high costs still represent significant barriers for micro and small entrepreneurs to enter this market. In this context, this work presents the development of a white-label e-commerce platform, designed to be customizable and meet the needs of small retailers. The central objective consists of providing a web application that enables product registration, inventory control, user management, and integrated sales, thereby democratizing access to technological retail tools. For the solution's construction, TypeScript was used alongside the Next.js and Node.js frameworks, with data persistence in PostgreSQL, aiming to ensure performance and scalability. The results demonstrate the technical feasibility of a system that simplifies virtual store management, allowing distinct visual and operational configurations for each merchant. It is concluded that the proposed solution effectively contributes to the digital inclusion of small businesses, offering a functional, low-cost, and technically accessible alternative for commercial expansion in the online environment.

Keywords: e-commerce, Retail, White Label Platform, Web

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

De acordo com Rogers (2017), a transformação digital não se resume à atualização de tecnologias, mas a uma mudança estratégica que redefine a forma como as empresas geram valor e interagem com redes de clientes cada vez mais conectados. Nesse contexto, o comércio eletrônico consolidou-se como uma alternativa vital para empreendedores, movimentando mais de R$170 bilhões no Brasil em 2022, segundo dados da ABCOMM (2025).

Apesar desse cenário promissor, existe uma lacuna significativa no mercado, onde pequenos comerciantes enfrentam barreiras para ingressar no mercado digital, devido principalmente à complexidade técnica das soluções existentes e à falta de infraestrutura adequada, conforme apontado por Karimi e Walter (2016). Complementando essa visão, Turban et al. (2017) destacam que a manutenção de sistemas de comércio eletrônico exige investimentos em infraestrutura e suporte técnico que muitas vezes superam a capacidade de investimento de pequenos comerciantes, restringindo sua expansão digital a soluções limitadas ou de difícil operação.

Para superar tais limitações, as arquiteturas de aplicações web modernas têm evoluído para modelos distribuídos que priorizam a escalabilidade e a manutenção simplificada, permitindo que a lógica de negócio e a interface de usuário operem de forma independente (EKPOBIMI, 2024). Nesse panorama, o modelo de software como serviço, ou SaaS (*Software as a Service*), surge como uma alternativa estratégica, pois transfere a responsabilidade da gestão de infraestrutura para o provedor, reduzindo o custo de entrada para o usuário final. Quando aliado ao conceito de *white label*, este modelo permite a criação de plataformas genéricas onde a estrutura base do software é compartilhada, mas a identidade visual é adaptada para cada lojista, reduzindo o esforço de reconfiguração técnica (ROGERS, 2017).

Diante dessa realidade, este trabalho propõe o desenvolvimento de uma plataforma *web*, solução que consiste em um aplicação *e-commerce* do tipo *white label* com *interface* voltada para o ambiente *desktop*, projetado para permitir que um comerciante individual cadastre produtos, controle estoque e realize vendas online.

A escolha deste tema justifica-se pelo potencial de impacto social e econômico ao oferecer autonomia tecnológica ao pequeno empreendedor. Ao fornecer uma plataforma que abstrai a complexidade de infraestrutura e reduz custos operacionais, este trabalho contribui para o fortalecimento do comércio local no ecossistema digital. Além disso, a relevância técnica deste estudo reside na exploração do modelo *white label* como estratégia para a criação de sistemas genéricos, capazes de atender a diferentes nichos de mercado com um esforço reduzido de reconfiguração.

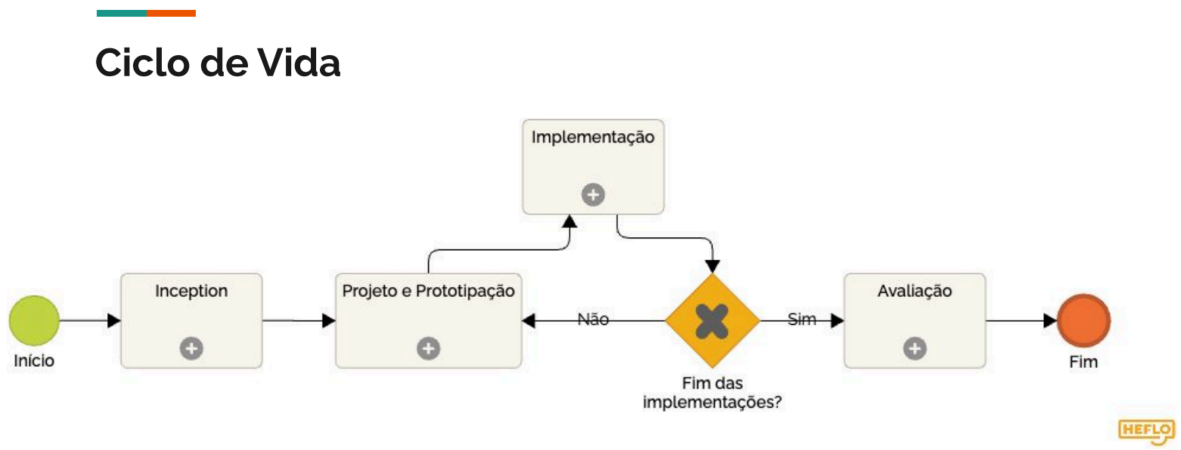
Tendo isto em vista, estabelece-se como objetivo geral o desenvolvimento de uma aplicação *web* de *e-commerce white label* voltada para comerciantes de pequeno porte, na qual será possível customizar a identidade visual e dados relacionados ao pagamento dos produtos, como a conta de destino e o valor de frete. Para alcançar esse objetivo, pretende-se propor um sistema adaptado a pequenos comerciantes, implementar a aplicação *web* com base nos requisitos levantados durante o processo de análise e, por fim, realizar uma avaliação do produto desenvolvido por meio da coleta de avaliações de usuários da plataforma.

2 METODOLOGIA

De acordo com Gil (2002), a pesquisa pode ser classificada de diferentes maneiras conforme seus objetivos, procedimentos e abordagem. No contexto deste trabalho, a investigação se caracteriza como exploratória, por buscar compreender e propor uma solução tecnológica para um problema real enfrentado por pequenos comerciantes. Além disso, é também uma estudo de caso, por se fundamentar em estudos teóricos sobre comércio eletrônico e desenvolvimento de *software*, e aplicada, realizou à construção de um sistema funcional que poderá ser utilizado na prática. Quanto à abordagem, trata-se de uma pesquisa qualitativa, uma vez que a avaliação do produto foi realizada com base na percepção e avaliações de usuários reais da aplicação desenvolvida.

Para conduzir o desenvolvimento do sistema proposto, foi adotada uma abordagem iterativa e incremental, inspirada em um fluxo adaptado proposto pelo professor Aislan Rafael na disciplina de Projeto Integrador III do Instituto Federal de Educação, Ciência e Tecnologia do Piauí – Campus Picos (Sousa, 2023). Esse fluxo é composto por quatro etapas principais: *Inception*, Projeto e Prototipação, Implementação e Avaliação. Cada fase contempla atividades específicas que visam garantir a construção gradual e fundamentada da aplicação *web* proposta.

Figura 1 - Ciclo de vida



Fonte: Sousa (2023)

2.1 INCEPTION

A etapa de *Inception* tem como objetivo o levantamento inicial de informações e requisitos, bem como a compreensão do contexto do problema e do público-alvo. Nessa fase, foram utilizados diversos artefatos de design e análise de produto, com o intuito de fundamentar a visão do sistema e alinhar expectativas. Nesta fase foram produzidos os seguintes artefatos:

- ***Briefing* do projeto**: documento inicial que resume os objetivos, necessidades e escopo da aplicação;
- **Diagrama de casos de uso**: ilustração das interações entre os usuários e o sistema;
- ***Product backlog***: lista inicial de funcionalidades priorizadas para o desenvolvimento.

2.2 PROJETO E PROTOTIPAÇÃO

Com base nas informações coletadas na etapa anterior, foi realizada a modelagem técnica da aplicação e a prototipação das interfaces. Foram produzidos os seguintes artefatos:

- **Modelo Entidade-Relacionamento**: representação das entidades, atributos e relacionamentos do banco de dados;
- ***Wireframes***: layout refinado e organizadas por telas funcionais.

2.3 IMPLEMENTAÇÃO

A fase de implementação corresponde ao desenvolvimento efetivo da aplicação, com base nos protótipos e modelos definidos. O desenvolvimento seguiu uma abordagem iterativa: a cada ciclo, novas funcionalidades foram desenvolvidas e integradas ao sistema. Esse processo continuou até que o sistema atendeu aos requisitos definidos e atingisse um nível satisfatório de estabilidade, funcionalidade e usabilidade.

2.4 AVALIAÇÃO

A última etapa consiste na avaliação do sistema implementado. Para isso, foi disponibilizada uma versão funcional da aplicação a um grupo de usuários. A avaliação foi realizada por meio de formulários de avaliações, com perguntas relacionadas à usabilidade, funcionalidades e clareza da interface.

3 TECNOLOGIAS ENVOLVIDAS

Todo o desenvolvimento do projeto foi realizado no ambiente de desenvolvimento integrado *Visual Studio Code*, onde as aplicações foram confeccionadas com Typescript, utilizando o *framework* Next.js no *frontend* e Node.js no *backend.* Para a persistência de dados, o PostgreSQL foi o gerenciador de banco de dados escolhido.

Outras tecnologias escolhidas, mas que não estão diretamente ligadas ao desenvolvimento, são o Git e Github para versionamento do código.

3.1 TYPESCRIPT

TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft que estende o JavaScript adicionando tipagem estática opcional. Segundo Emmanni (2021), o uso do TypeScript em conjunto com *frameworks* modernos oferece maior robustez ao desenvolvimento *web*, favorecendo a escalabilidade e a confiabilidade das aplicações. Ainda segundo o autor, essa abordagem contribui para a detecção precoce de erros e melhora a legibilidade e manutenção do código, especialmente em projetos de grande porte.

3.2 NEXT.JS

Desenvolvido pela Vercel, Next.js é um *framework* de código aberto baseado em React que permite a criação de aplicações *web*. Ele oferece recursos como renderização do lado do servidor, geração de páginas estáticas e rotas automáticas, facilitando a construção de aplicações. Segundo EKPOBIMI (2024), essas funcionalidades fazem do Next.js uma solução eficaz para o desenvolvimento de sistemas *web* robustos, combinando simplicidade de uso com alta performance.

3.3 NODE.JS

O Node.js é um ambiente de execução JavaScript baseado no motor V8 que utiliza uma arquitetura orientada a eventos e operações de entrada/saída (I/O) não bloqueantes. Essa abordagem permite o processamento eficiente de múltiplas requisições simultâneas através de um ciclo de eventos, eliminando a necessidade de gerenciar múltiplas *threads* para cada conexão. De acordo com Chisnall (2018), essa estrutura é fundamental para a criação de programas de rede de alta performance, pois permite que o sistema lide com milhares de conexões concorrentes com um baixo overhead de memória. O autor ressalta que a eficiência do Node.js reside na sua capacidade de manter a responsividade da aplicação mesmo sob alta carga, tornando-o uma solução robusta para o desenvolvimento de serviços web modernos e escaláveis.

3.4 POSTGRESQL

PostgreSQL é um sistema gerenciador de banco de dados relacional de código aberto, amplamente reconhecido por sua estabilidade, flexibilidade e conformidade com os padrões *structured query language* (SQL). Em um estudo comparativo entre os principais sistemas gerenciadores de banco de dados relacionais, Wodyk e Skublewska-Paszkowska (2020) destacam o desempenho consistente do PostgreSQL em operações de leitura e escrita em ambientes de aplicações *web*, especialmente em situações de maior carga e volume de dados. Os autores ressaltam ainda que o sistema apresenta vantagens notáveis em termos de eficiência no uso de recursos e suporte a operações complexas, o que o torna adequado para cenários que demandam confiabilidade e processamento intensivo de dados estruturados.

3.5 VISUAL STUDIO CODE

O Visual Studio Code é um editor de código fonte leve e robusto, desenvolvido pela Microsoft, que combina a simplicidade de um editor de texto com recursos avançados de desenvolvimento. Conforme descrito pela Microsoft (2025), a ferramenta redefine a edição de código ao oferecer suporte nativo a depuração, controle Git integrado e o recurso IntelliSense, que fornece conclusões inteligentes de código baseadas em tipos de variáveis, definições de função e módulos importados. Sua arquitetura extensível permite a personalização do ambiente para diversas linguagens e *runtimes*, como Node.js (MICROSOFT, 2025).

3.6 GIT

O Git é um sistema de controle de versão distribuído, gratuito e de código aberto, projetado para lidar com projetos de qualquer dimensão com velocidade e eficiência. Segundo a documentação oficial do Git (2025), a ferramenta diferencia-se por sua arquitetura distribuída, na qual cada desenvolvedor possui uma cópia completa do histórico do código, e não apenas a versão atual dos arquivos. Essa característica garante a integridade dos dados e permite fluxos de trabalho não lineares, possibilitando o desenvolvimento paralelo através de ramificações locais independentes (GIT, 2025).

3.7 GITHUB

O GitHub é uma plataforma de desenvolvimento baseada em nuvem, utilizada para hospedar, compartilhar e colaborar em códigos de *software*. De acordo com o GitHub (2025), a plataforma atua como um *hub* completo para o ciclo de vida de desenvolvimento, integrando ferramentas de automação de fluxo de trabalho, segurança e revisão de código. O serviço utiliza o Git como sistema de versão subjacente, facilitando a colaboração em equipe através de funcionalidades como *pull requests* e rastreamento de problemas, promovendo uma gestão eficiente de projetos de *software* (GITHUB, 2025).

4 MODELAGEM DO PROJETO

Nesta seção, são apresentados os detalhes fundamentais sobre o planejamento e a estruturação da solução proposta. Serão descritos os requisitos funcionais e não funcionais levantados, a arquitetura de *software* adotada, bem como os diagramas que ilustram a organização dos dados e a interação dos usuários com o sistema.

4.1 LEVANTAMENTO DE REQUISITOS

A definição de requisitos é uma etapa fundamental no desenvolvimento de software, pois estabelece as bases sobre as quais todo o sistema será construído. A partir do levantamento de dados, foram identificadas e documentadas as necessidades essenciais para que a aplicação atenda ao objetivo de auxiliar pequenos comerciantes.

4.1.1 Requisitos funcionais

Os requisitos funcionais descrevem as funcionalidades específicas e os comportamentos esperados da aplicação. Eles detalham o que o sistema deve fazer, cobrindo as interações do usuário desde o gerenciamento de produtos até a finalização de compras, conforme apresentado na tabela a seguir.

Tabela 1 - Requisitos funcionais

ID	Requisito
RF001	O sistema deve permitir que o usuário crie sua conta
RF002	O sistema deve permitir que o usuário edite sua conta
RF003	O sistema deve permitir que o usuário visualize os dados de sua conta
RF004	O sistema deve permitir que o usuário valide seu email
RF005	O sistema deve permitir que o usuário cancele sua conta
RF006	O sistema deve permitir que o usuário recupere sua senha
RF007	O sistema deve permitir que o usuário se autentique
RF008	O sistema deve permitir que o usuário cadastre endereços

RF009	O sistema deve permitir que o usuário edite um endereço
RF010	O sistema deve permitir que o usuário liste os endereços cadastrados
RF011	O sistema deve permitir que o usuário remova endereços cadastrados
RF012	O sistema deve permitir que o usuário veja a lista de produtos
RF013	O sistema deve permitir que o usuário busque por produtos
RF014	O sistema deve permitir que o usuário veja detalhes de produtos cadastrados
RF015	O sistema deve permitir que o usuário adicione produtos cadastrados no carrinho
RF016	O sistema deve permitir que o usuário edite a quantidade de produtos no carrinho
RF017	O sistema deve permitir que o usuário remova produtos do carrinho
RF018	O sistema deve permitir que o usuário veja seu carrinho
RF019	O sistema deve permitir que o usuário veja o total da compra, discriminando o subtotal dos produtos e o valor da taxa de entrega
RF020	O sistema deve permitir que o usuário faça o pagamento dos itens do carrinho
RF021	O sistema deve exibir uma confirmação de pedido realizado com sucesso para o usuário após o pagamento
RF022	O sistema deve permitir que o usuário veja seus pedidos feitos
RF023	O sistema deve permitir que o usuário veja os detalhes de um pedido
RF024	O sistema deve permitir que o administrador veja os pedidos feitos
RF025	O sistema deve permitir que o administrador atualize os pedidos feitos
RF026	O sistema deve permitir que o administrador filtre os pedidos feitos
RF027	O sistema deve permitir que o administrador faça a configuração de customizações da loja
RF028	O sistema deve permitir que o administrador defina o valor para a taxa de entrega
RF029	O sistema deve permitir que o administrador edite sua conta

RF030	O sistema deve permitir que o administrador visualize os dados de sua conta
RF031	O sistema deve permitir que o administrador recupere sua senha
RF032	O sistema deve permitir que o administrador se autentique
RF033	O sistema deve permitir que o administrador remova outros administradores
RF034	O sistema deve permitir que o administrador veja a lista de administradores existentes
RF035	O sistema deve permitir que o administrador cadastre produtos
RF036	O sistema deve permitir que o administrador edite produtos
RF037	O sistema deve permitir que o administrador delete produtos
RF038	O sistema deve permitir que o administrador busque os produtos cadastrados
RF039	O sistema deve permitir que o administrador busque por produtos cadastrados
RF040	O sistema deve permitir que o administrador faça o upload de uma imagem do produto
RF041	O sistema deve permitir que o administrador veja o total faturado e a quantidade de pedidos realizados no mês vigente

4.1.2 Requisitos não funcionais

Já os requisitos não funcionais definem os atributos de qualidade e as restrições técnicas da solução. Estes requisitos determinam como o sistema deve operar, estabelecendo critérios críticos como desempenho, segurança e padrões de usabilidade que garantem a eficiência da plataforma, listados na sequência.

Tabela 2 - Requisitos não funcionais

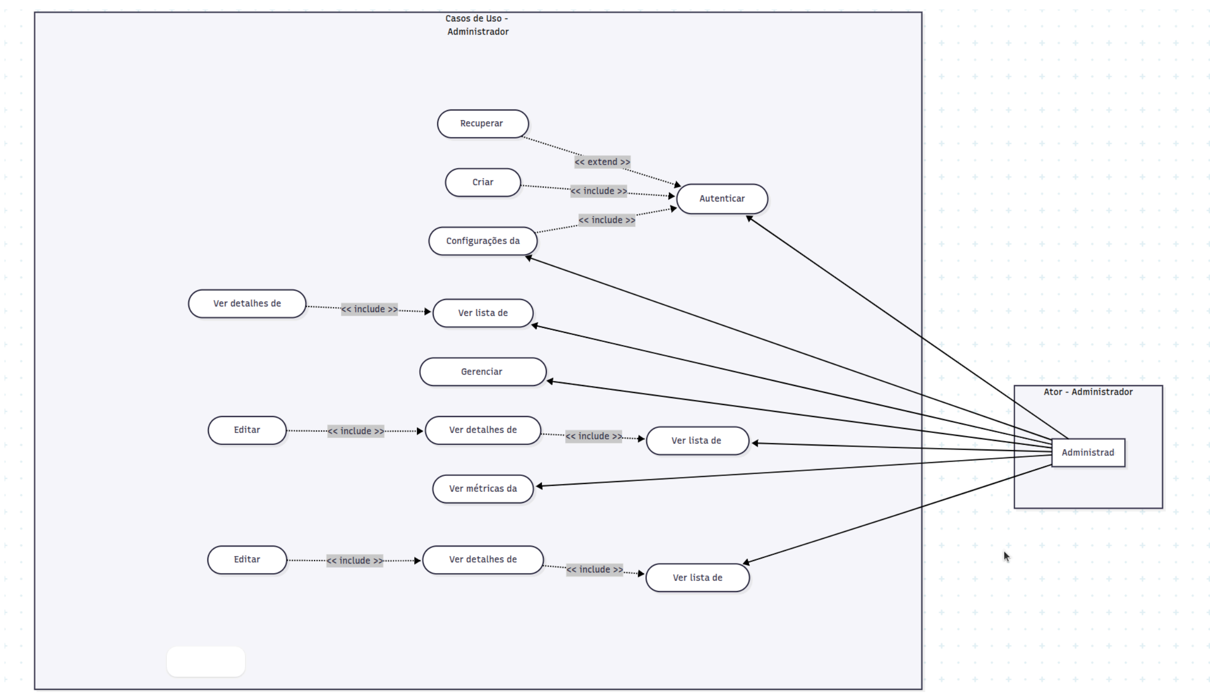
ID	Requisito
RNF001	Suportar ao menos 10000 produtos cadastrados no estoque
RNF002	Suportar ao menos 10000 usuários cadastrados
RNF003	Seguir as regulamentações locais de proteção de dados e segurança da informação

RNF004	Garantir um tempo de resposta para ações do usuário no sistema de, no máximo, 2 segundos
RNF005	Estar disponível 24 horas por dia, 7 dias por semana
RNF006	Atualizações devem manter o software interrompido no máximo por 30 minutos

4.2 DIAGRAMAS DE CASOS DE USO

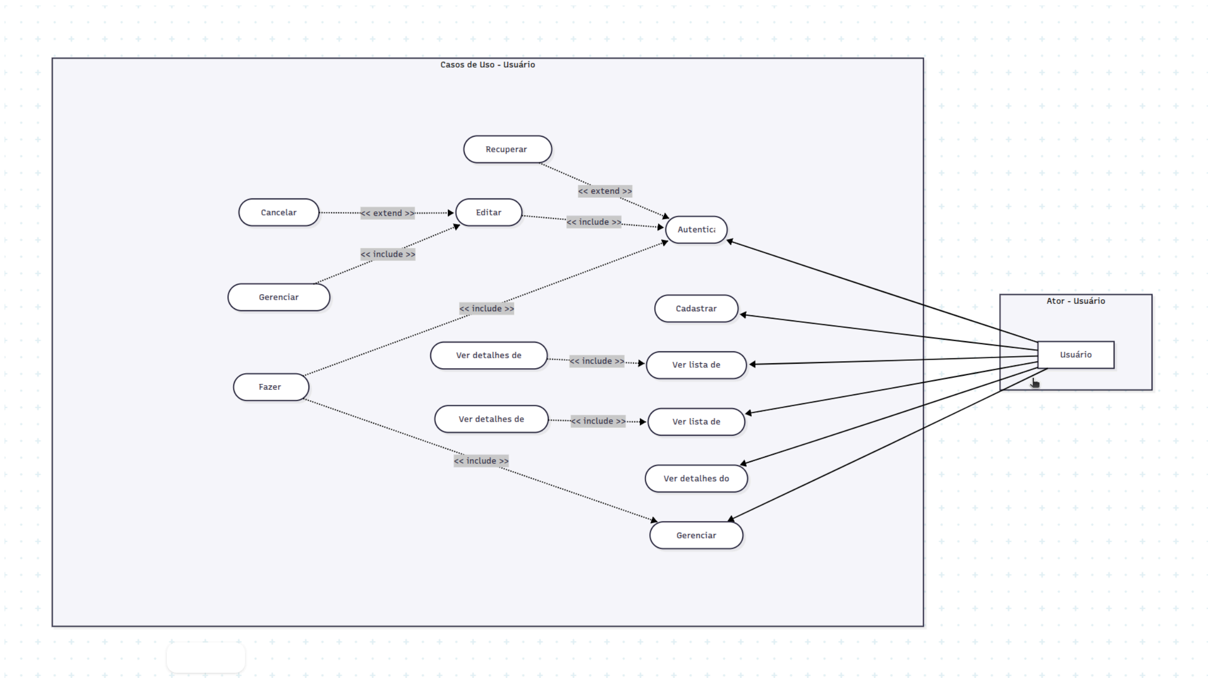
A elaboração dos diagramas de caso de uso constitui uma etapa essencial para a visualização das interações entre os usuários e o sistema. A partir dessa representação gráfica, é possível mapear as funcionalidades da plataforma sob a perspectiva dos atores envolvidos, detalhando como o administrador e o cliente final interagem com os módulos de venda e gestão. Esta ferramenta serve como um guia visual que traduz os requisitos levantados em ações práticas, assegurando que o fluxo de operações da solução esteja alinhado às necessidades do negócio.

Figura 2 - Diagrama de caso de uso do administrador



Fonte: Próprio Autor

Figura 3 - Diagrama de caso de uso do usuário final



Fonte: Próprio Autor

4.3 ARQUITETURA DO SISTEMA

A arquitetura da solução foi projetada seguindo um modelo distribuído de repositórios, visando o desacoplamento entre as *interfaces* de usuário e os serviços de processamento. O ecossistema é composto por três repositórios distintos: um dedicado ao *frontend* da loja virtual, outro exclusivo para o painel administrativo e um terceiro repositório centralizado que abriga os serviços de *backend*.

No *frontend*, ambas as aplicações foram desenvolvidas utilizando Next.js e Tailwind CSS. Elas operam como clientes independentes, responsáveis pela renderização da *interface* e pela gestão de estado visual. Para assegurar o cumprimento do requisito RNF004, que estabelece um tempo de resposta máximo de 2 segundos, as requisições assíncronas são configuradas com políticas de *timeout* rigorosas. Essa abordagem garante que a camada de apresentação permaneça isolada da lógica de persistência e forneça um *feedback* imediato ao usuário caso o processamento exceda o limiar de desempenho esperado.

No *backend*, optou-se por uma estratégia de repositório único para abrigar os serviços da API. Embora sirvam a propósitos distintos, os serviços compartilham o mesmo domínio de dados. Internamente, a aplicação segue uma arquitetura em camadas gerenciada pelo container de injeção de dependência. Esta estrutura organiza o código em:

- **Controllers**: responsáveis por receber as requisições REST e validar os dados de entrada;
- **Services**: onde reside a lógica de negócio e as regras de validação do sistema;
- **Repositories**: camada responsável pela abstração do acesso ao banco de dados.

4.3.1 Padrões de projeto

Na estruturação do código, foram aplicados os padrões de projeto listados abaixo:

- **Singleton**: assegurar que classes críticas, como o gerenciador de conexões com o banco de dados, possuam uma única instância compartilhada por toda a aplicação, otimizando o uso de recursos. No projeto, este padrão é aplicado no cliente de conexão com o banco de dados, assegurando que as requisições compartilhem o mesmo *pool* de conexões e evitando o desperdício de recursos de memória do servidor;
- **Adapter**: compatibiliza *interfaces* de comunicação, permitindo que o sistema interaja com serviços externos ou bibliotecas distintas sem comprometer a lógica interna. No projeto, este padrão é aplicado no módulo de segurança para a proteção de dados sensíveis em conformidade com a Lei Geral de Proteção de Dados (LGPD). O *Adapter* encapsula bibliotecas de criptografia, garantindo que informações pessoais sejam transformadas em *hashes* antes da persistência no banco de dados. Dessa forma, além de assegurar a privacidade dos dados do usuário, o padrão permite que a biblioteca criptográfica seja substituída futuramente sem a necessidade de alterar as regras de negócio do sistema;
- **Princípio da Inversão de Dependência**: assegura que os módulos de alto nível com regras de negócio não dependam de módulos de baixo nível, onde

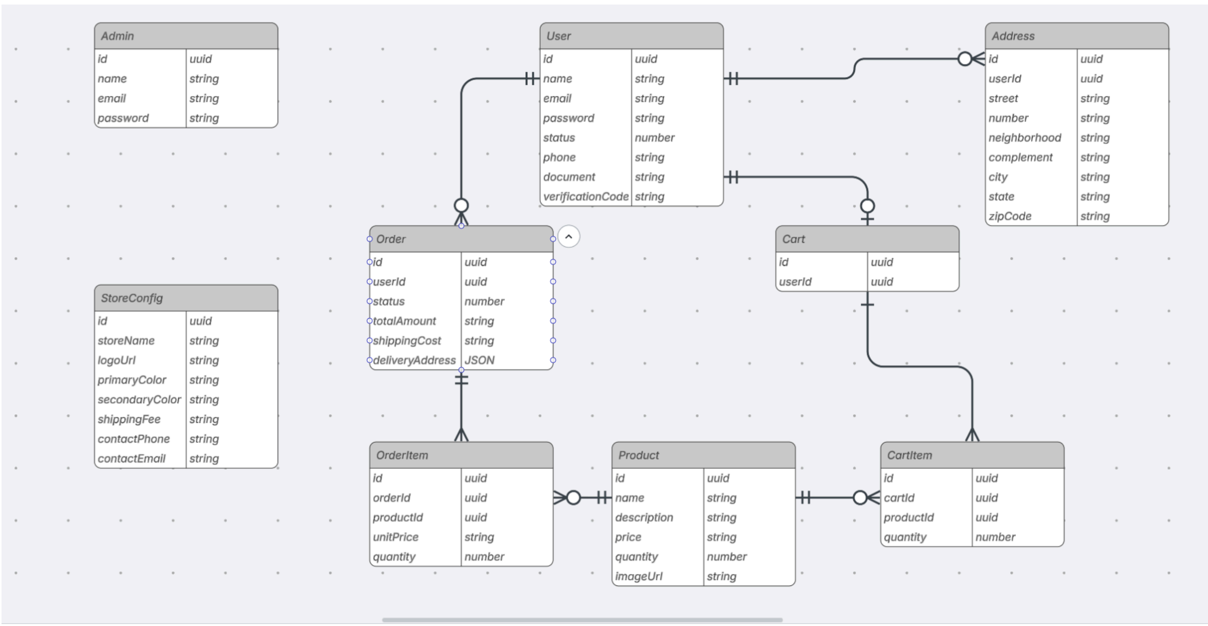
há detalhes de implementação, mas sim de abstrações. Na arquitetura desta solução, isso é visível quando as classes de serviço solicitam uma interface de repositório em vez de uma classe específica do TypeORM, garantindo que o sistema seja testável e flexível a mudanças;

- **Injeção de Dependência**: transfere a responsabilidade de instanciar as dependências para um contêiner externo, promovendo maior modularidade. No desenvolvimento deste projeto, utiliza-se a biblioteca InversifyJS como container de inversão de controle, que gerencia automaticamente a estrutura de dependências, promovendo um código mais modular e facilitando a substituição de componentes durante os testes.

4.4 DIAGRAMAS DE ENTIDADES-RELACIONAMENTOS

Para estruturar o armazenamento e a persistência dos dados da aplicação, foi elaborado o Modelo Entidades-Relacionamentos (MER). Este diagrama detalha a estrutura lógica do banco de dados, definindo as principais entidades, bem como seus atributos e os relacionamentos entre elas.

Figura 4 - Diagrama Entidades-Relacionamentos



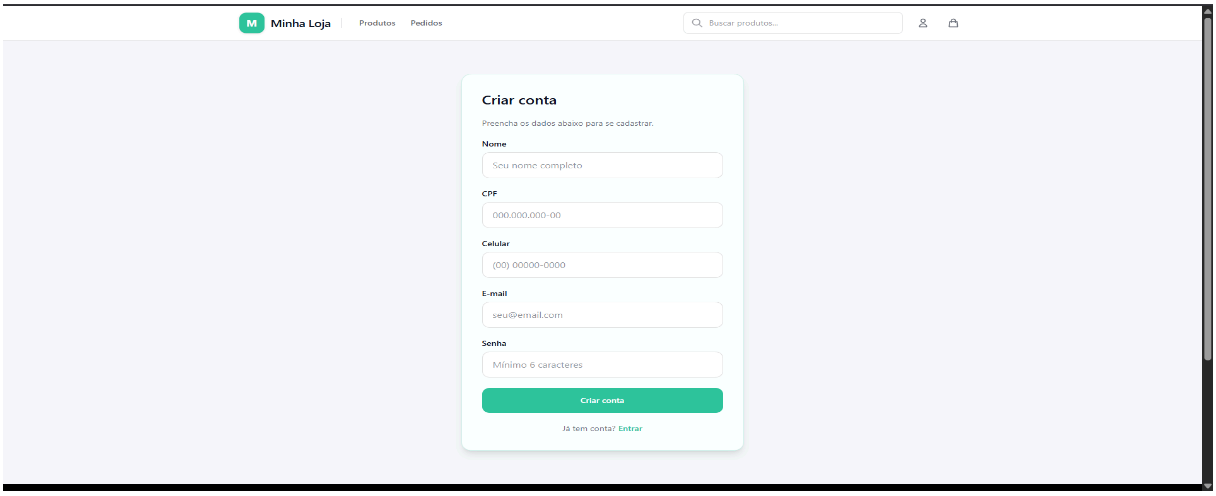
Fonte: Próprio Autor

4.5 INTERFACE

A *interface* da aplicação foi construída sobre a base do *framework* Next.js, que utiliza a biblioteca React.js como motor de renderização. Essa arquitetura foi selecionada por permitir a criação de componentes reutilizáveis e modulares, essenciais para a manutenção de um sistema *white label*. Para a estilização e definição da identidade visual, adotou-se o Tailwind CSS.

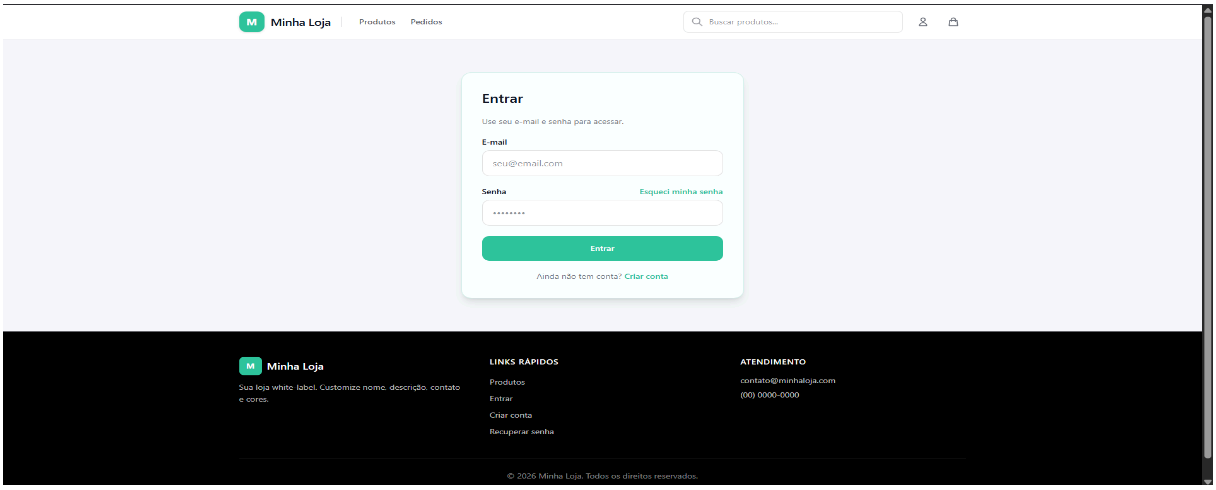
O acesso inicial à plataforma ocorre através das telas de autenticação e criação de conta. Nestas *interfaces*, o usuário encontra de forma centralizada os formulários necessários para realizar o *login* ou criar uma nova conta. A estrutura destas páginas podem ser visualizadas nas figuras 3 e 4.

Figura 5 - Tela de cadastro de conta



Fonte: Próprio Autor

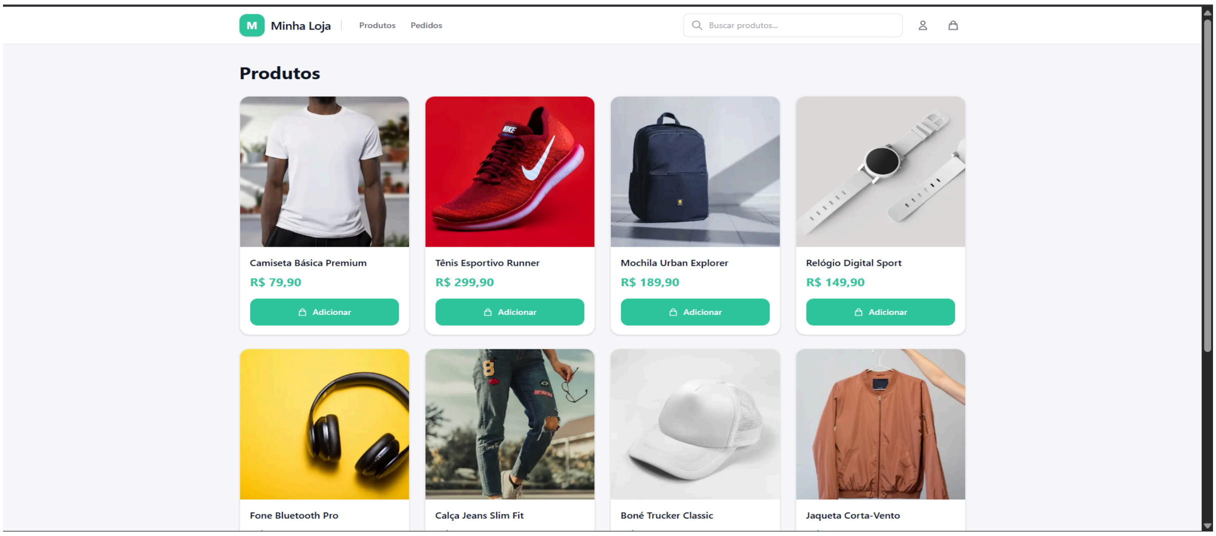
Figura 6 - Tela de login



Fonte: Próprio Autor

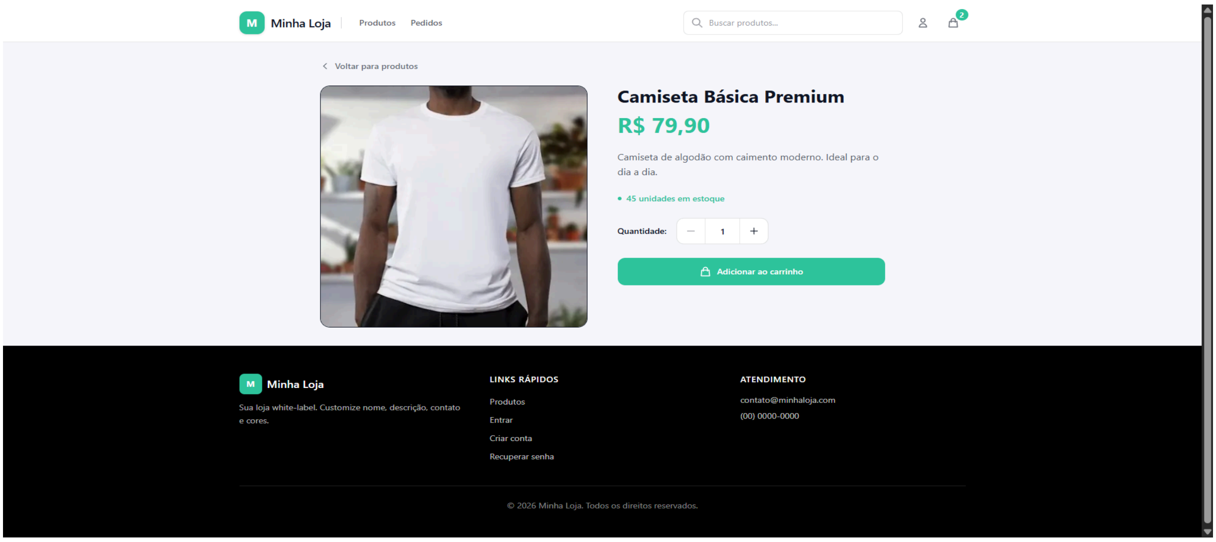
A experiência de compra centraliza-se na navegação pelo catálogo digital. A *interface* de listagem, apresentada na figura 5, organiza os produtos em uma disposição visual responsiva. Ao selecionar um item de interesse, o cliente é direcionado à tela de detalhes, ilustrada na figura 6, esta seção destaca informações como preço, descrição técnica e imagens.

Figura 7 - Tela de listagem de produtos



Fonte: Próprio Autor

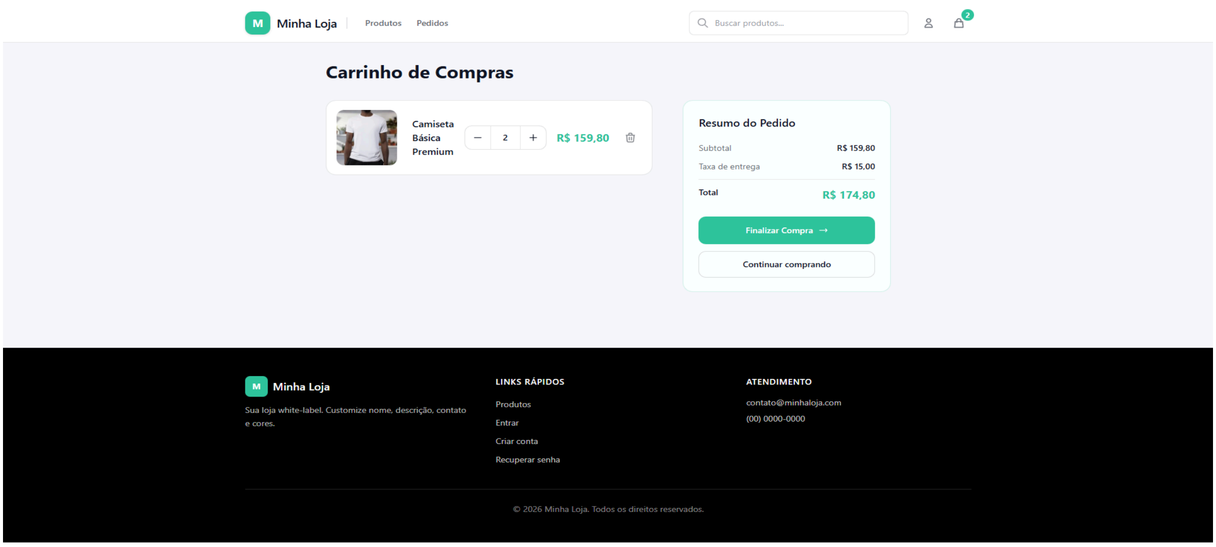
Figura 8 - Tela de detalhes do produto



Fonte: Próprio Autor

A *interface* do carrinho de compras, ilustrada na figura 7, permite a revisão dos itens selecionados antes da conclusão do pedido. Nesta tela, o usuário pode visualizar o valor total calculado automaticamente, alterar a quantidade de produtos ou removê-los da lista.

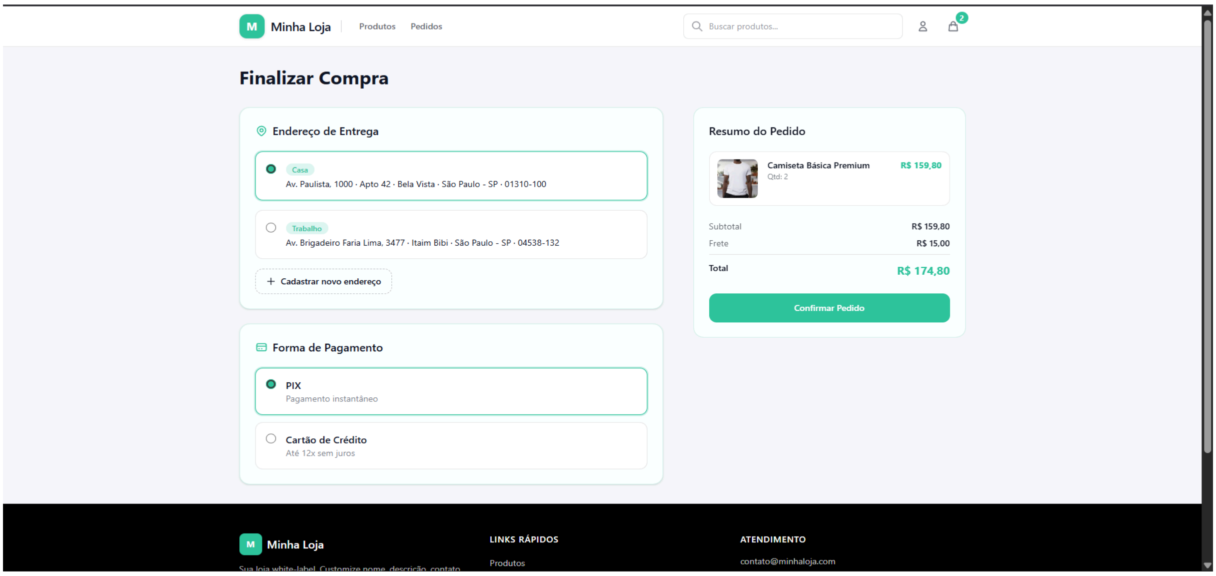
Figura 9 - Tela do carrinho



Fonte: Próprio Autor

Na figura 8 está presente a *interface* de pagamento, ela constitui a etapa final do processo de compra. Nesta seção, o usuário confirma o endereço de entrega e seleciona o método de pagamento desejado entre as opções integradas.

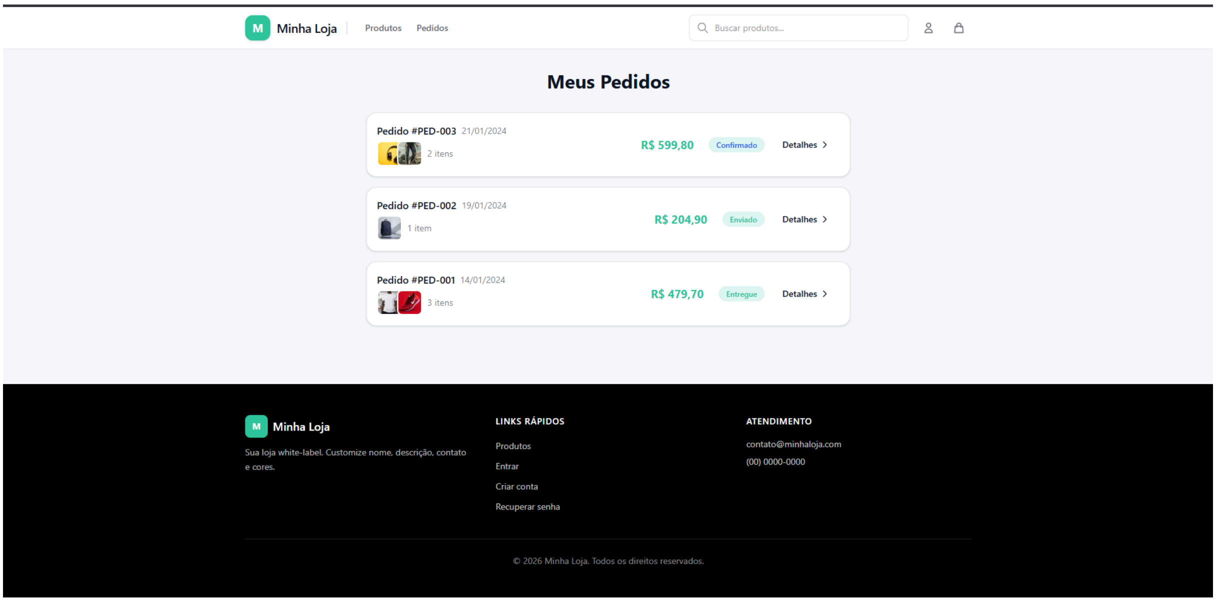
Figura 10 - Tela de pagamento



Fonte: Próprio Autor

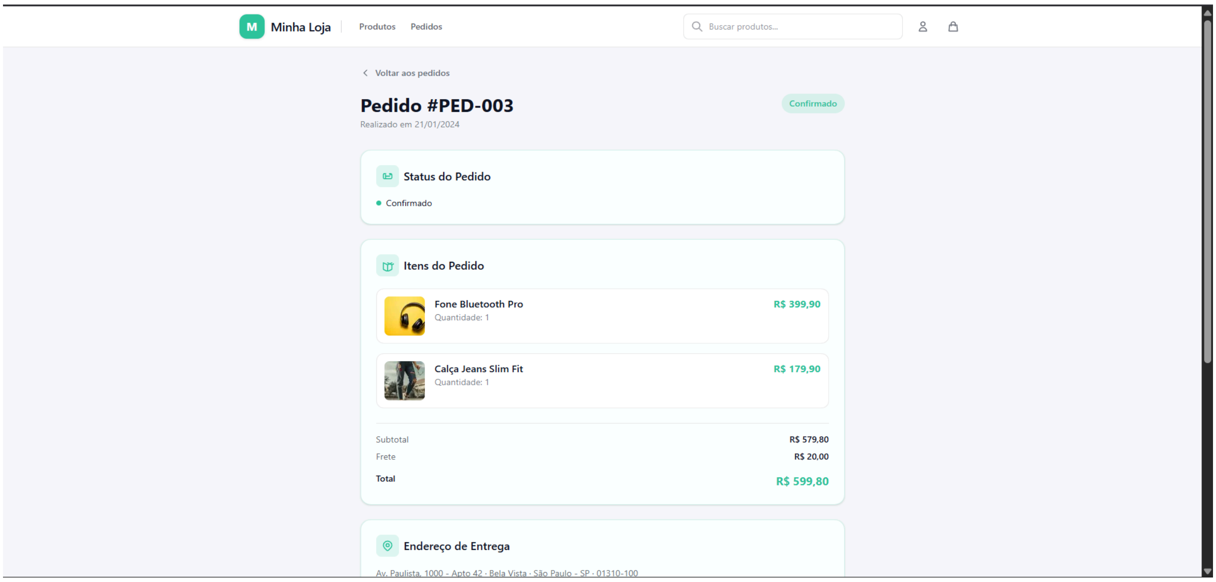
A gestão dos pedidos é realizada através da *interface* de listagem de pedidos, demonstrada na figura 9. Esta tela oferece uma visão geral das transações, permitindo o monitoramento do *status* de cada compra. Ao selecionar um registro específico, o sistema exibe a visão detalhada, *interface* presente na figura 10.

Figura 11 - Tela de listagem de pedidos



Fonte: Próprio Autor

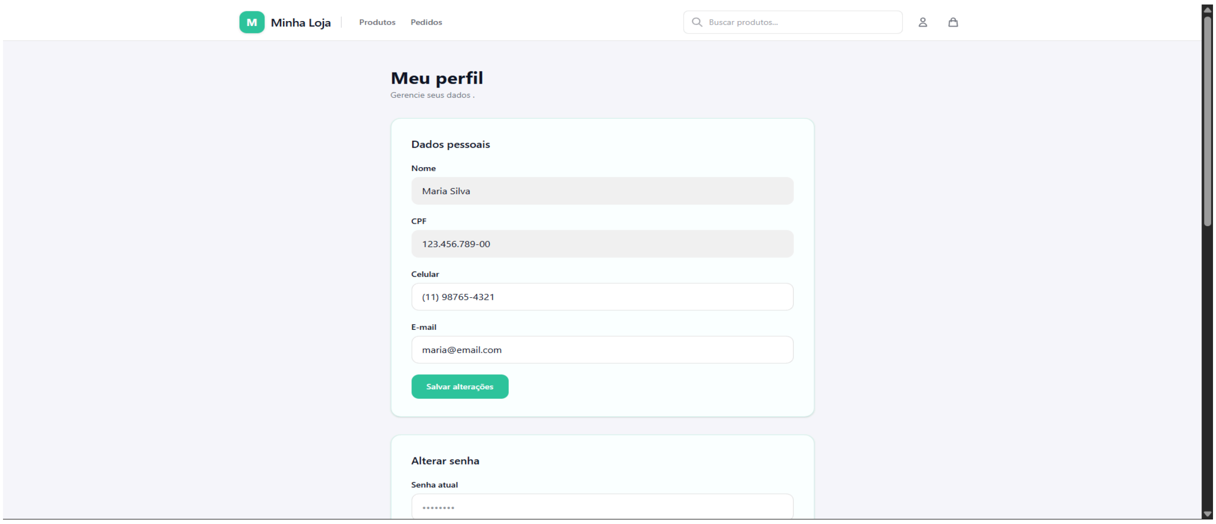
Figura 12 - Tela de detalhes do pedido



Fonte: Próprio Autor

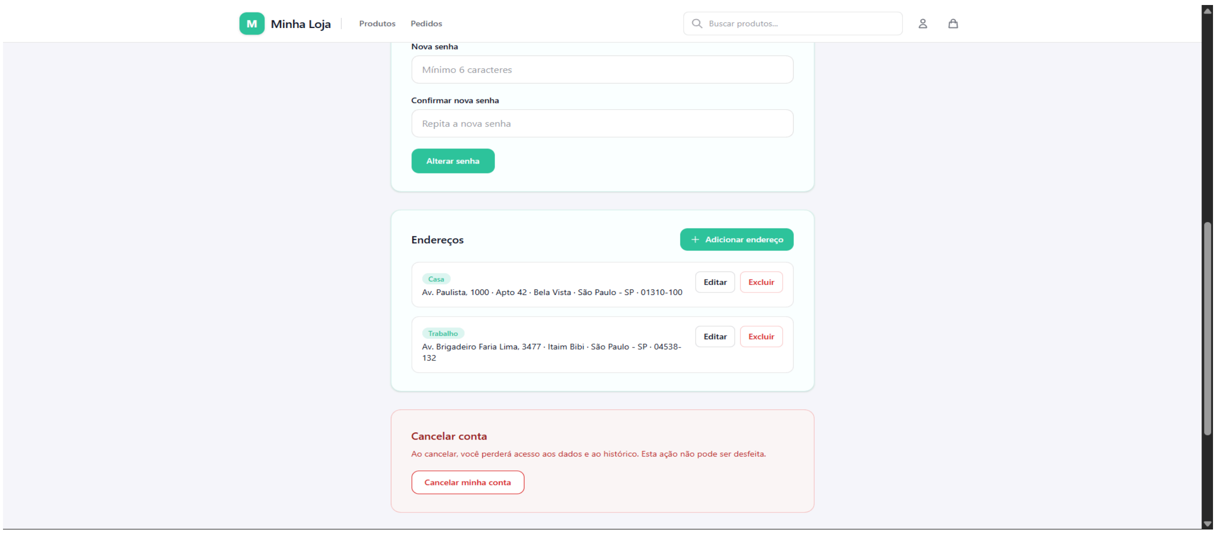
A manutenção dos dados cadastrais é realizada na *interface* de perfil do usuário, demonstrada nas figuras 11 e 12. Nesta seção, o cliente possui autonomia para visualizar e editar suas informações pessoais, além de realizar procedimentos de segurança, como a alteração de senha e o cancelamento definitivo da conta. A tela contempla ainda um módulo específico para a gestão de endereços, permitindo o cadastro e a atualização de múltiplos locais de entrega para agilizar o processo de finalização de futuras compras.

Figura 13 - Tela de perfil do usuário



Fonte: Próprio Autor

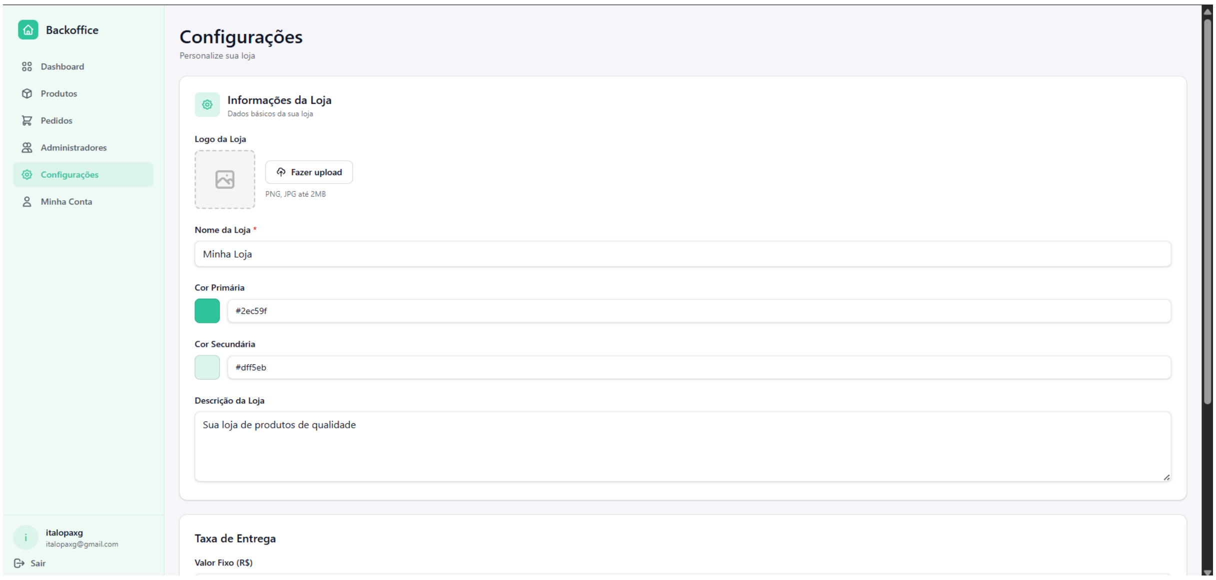
Figura 14 - Tela de perfil do usuário



Fonte: Próprio Autor

A personalização da identidade visual é realizada através da seção de configurações do sistema de administração, demonstrada na figura 13. Esta *interface* oferece autonomia ao comerciante para definir a aparência da loja, disponibilizando controles para o upload de logomarca, seleção da paleta de cores e atualização das informações de contato.

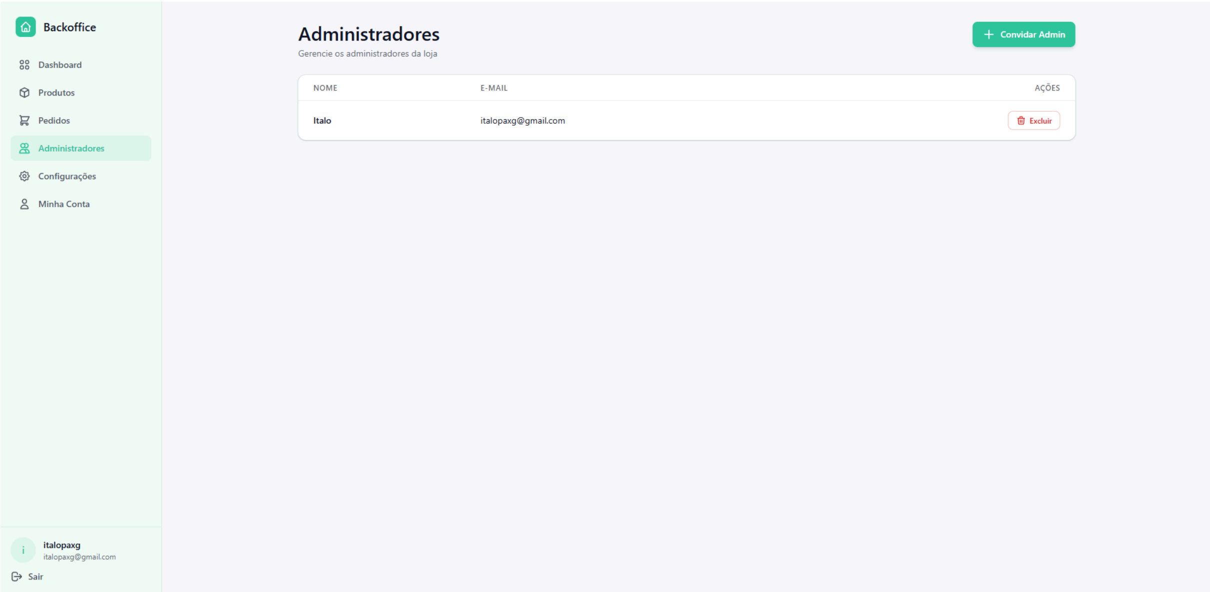
Figura 15 - Tela de configurações de personalização



Fonte: Próprio Autor

O controle de acesso ao painel administrativo é centralizado no módulo de gestão de administradores, apresentado na figura 14. Esta *interface* disponibiliza as operações fundamentais para a manutenção da equipe, permitindo criar novos perfis de administrador e listar os registros existentes.

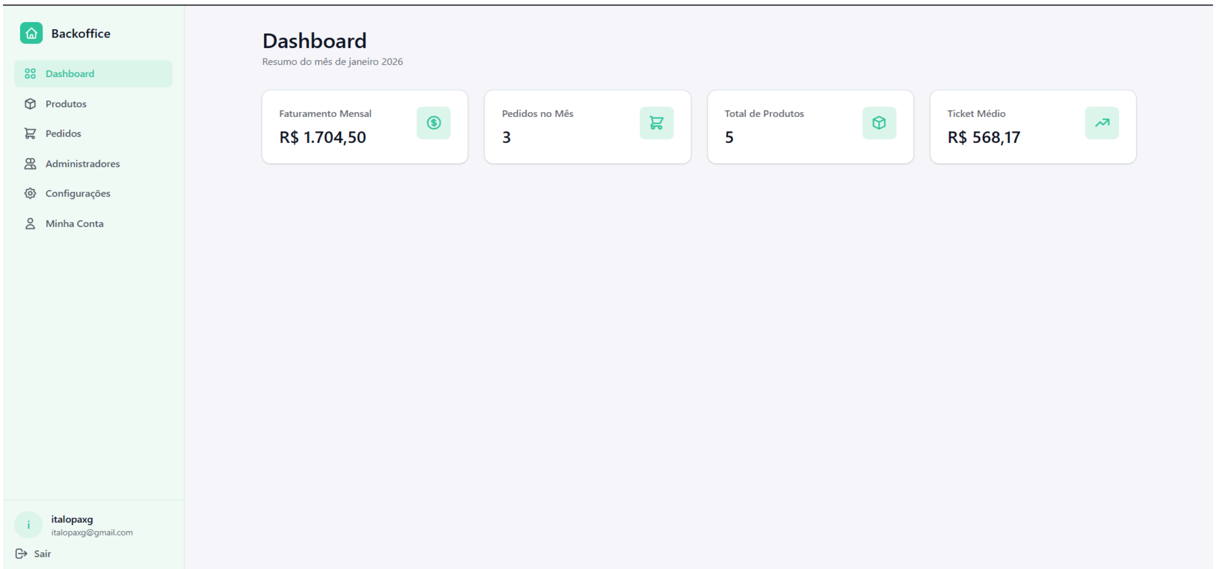
Figura 16 - Tela de gestão de administradores



Fonte: Próprio Autor

O painel de controle, apresentado na figura 15, atua como a tela inicial da área administrativa, fornecendo uma visão panorâmica do negócio. A *interface* consolida indicadores de desempenho em tempo real do mês vigente, com volume de vendas, receita acumulada, quantidade de itens vendidos e valor médio por venda.

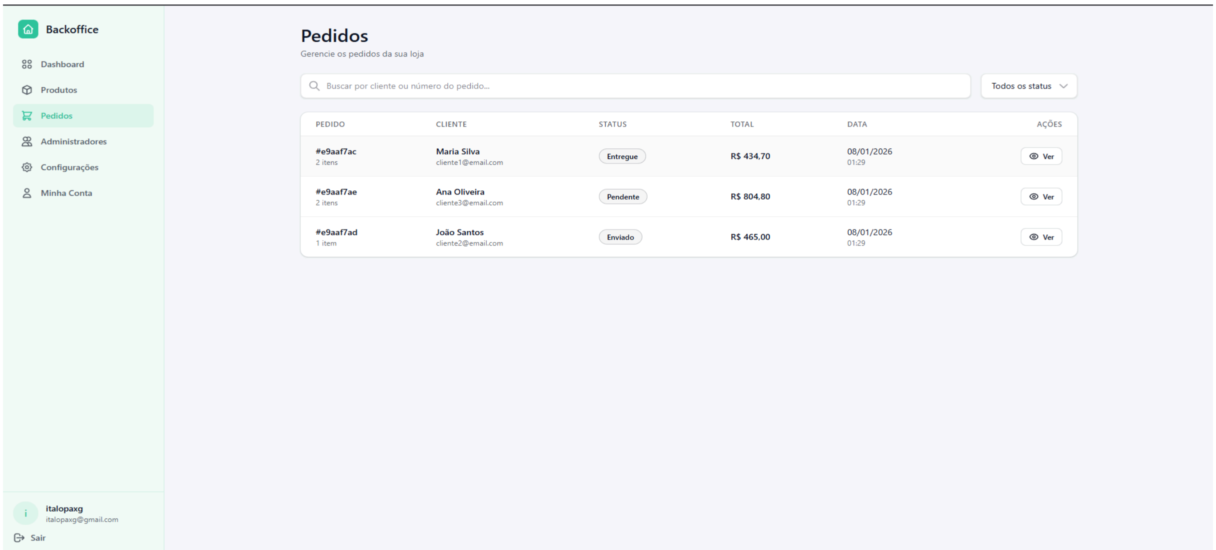
Figura 17 - Dashboard



Fonte: Próprio Autor

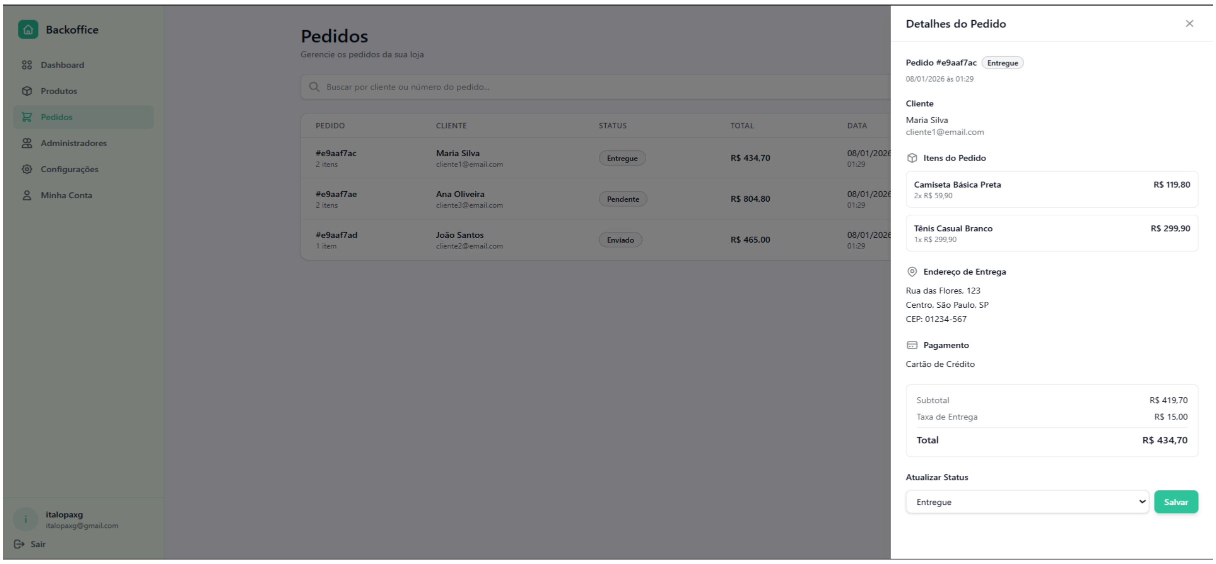
A gestão operacional das vendas concentra-se na aba de pedidos do painel administrativo, ilustrada nas figuras 16 e 17. Esta *interface* centraliza todos os pedidos recebidos, permitindo a filtragem dos registros. A partir desta listagem, o administrador tem acesso aos detalhes de cada compra para realizar o processamento logístico e atualizar a situação do pedido, garantindo o controle efetivo do fluxo de entregas.

Figura 18 - Tela de gestão de pedidos



Fonte: Próprio Autor

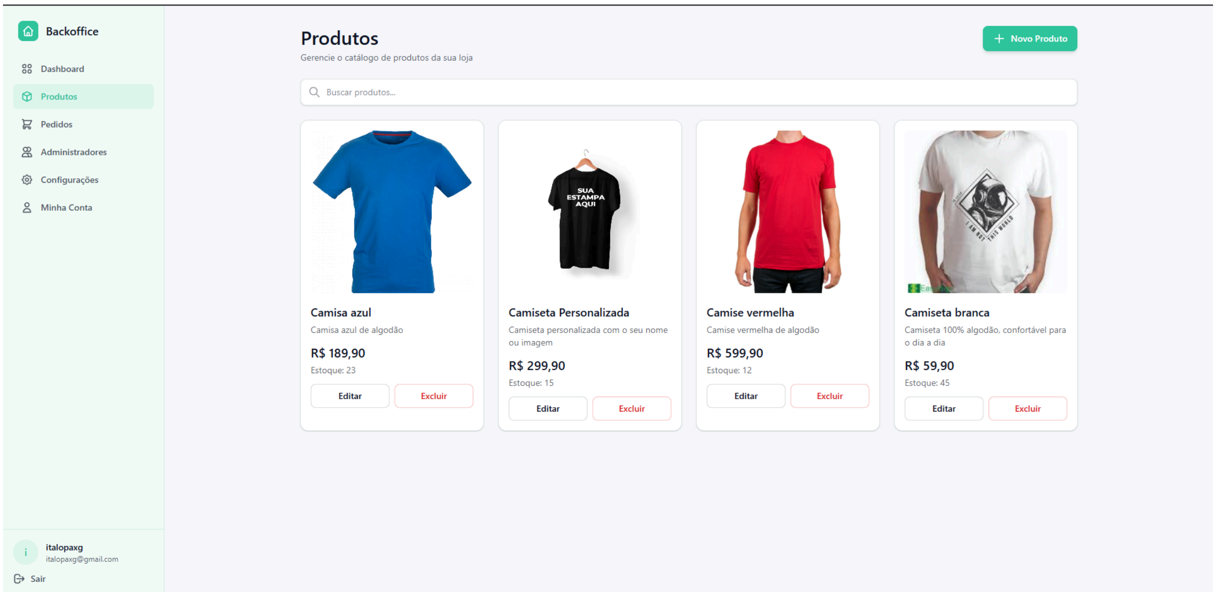
Figura 19 - Tela de detalhes de um pedido



Fonte: Próprio Autor

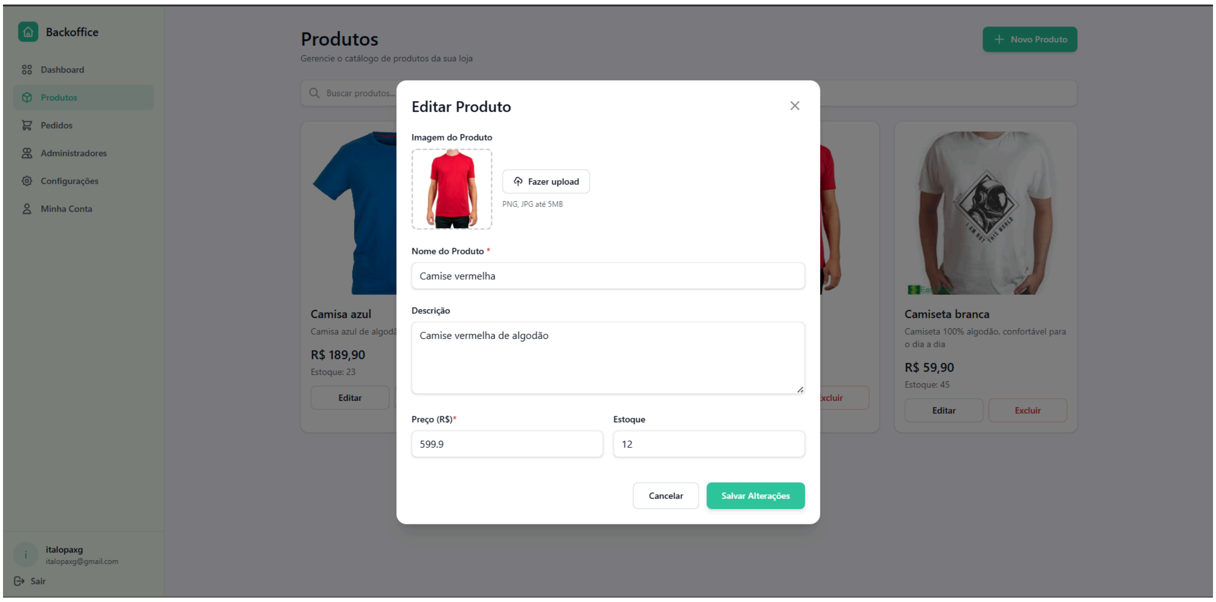
O gerenciamento do inventário ocorre na aba de produtos, ilustrada nas figuras 18 e 19. Nesta seção, o administrador possui controle total sobre o catálogo, dispondo de funcionalidades para cadastrar ou remover itens, atualizar informações e ajustar os níveis de estoque.

Figura 20 - Tela de gestão de produtos



Fonte: Próprio Autor

Figura 21 - Tela de detalhes de um produto



Fonte: Próprio Autor

5 SOFTWARE

O produto de *software* resultante deste trabalho consiste em uma plataforma de *e-commerce white label*. O código-fonte foi organizado de forma modular, utilizando repositórios distintos para as *interfaces* de usuário e uma estrutura centralizada para os serviços de *backend*, todos versionados através da plataforma GitHub. A aplicação consolida as funcionalidades de gestão administrativa e a vitrine de vendas, integrando-se através de APIs RESTful.

5.1 IMPLANTAÇÃO

A estratégia de implantação adotou uma arquitetura de infraestrutura distribuída em nuvem, selecionando ambientes otimizados para a natureza de cada componente do sistema. As aplicações de *frontend*, desenvolvidas em Next.js, foram implantadas na plataforma Vercel, aproveitando-se de sua rede global de distribuição de conteúdo e otimização nativa para o *framework*. Simultaneamente, os serviços de *backend* e o banco de dados foram hospedados em uma instância de computação virtual na Oracle Cloud Infrastructure. Essa configuração híbrida garante a alta performance e disponibilidade das *interfaces* visuais, ao mesmo tempo em que assegura o controle total sobre o ambiente de execução e a persistência dos dados no servidor de aplicação.

5.2 TESTES

Neste capítulo, são apresentados os procedimentos e resultados da submissão da plataforma de *e-commerce* à avaliação de usuários. O objetivo central desta etapa foi colher *feedbacks* estruturados para metrificar a efetividade da solução, identificando se a solução atende às necessidades dos pequenos comerciantes e onde são necessárias melhorias para garantir uma melhor experiência de uso.

5.2.1 Metodologia de teste

A seleção de um método de avaliação de usabilidade deve considerar a natureza do artefato e o perfil dos participantes envolvidos. Dentre as alternativas consolidadas na literatura, destacam-se a Avaliação Heurística, fundamentada na inspeção de interfaces por especialistas a partir de princípios de design e usabilidade (NIELSEN, 1994), e o Caminhamento Cognitivo, que foca na facilidade de aprendizado do sistema por meio da simulação de tarefas específicas (WHARTON et al., 1994). Contudo, a aplicação desses métodos pode apresentar limitações em cenários de desenvolvimento ágil ou com escassez de avaliadores experientes, uma vez que a eficácia da identificação de problemas de interface depende fortemente da expertise prévia do profissional (NIELSEN, 1994).

Nesse contexto, optou-se pela aplicação do System Usability Scale (SUS). A escolha fundamenta-se na evidência de que o SUS apresenta uma alta correlação com métricas de desempenho de tarefas, mesmo quando aplicado em amostras pequenas, variando de 8 a 12 usuários (BROOKE, 1996). Diferente de instrumentos mais extensos como o Questionnaire for User Interaction Satisfaction (QUIS), reconhecido por sua estrutura detalhada, porém longa (CHIN; DIEHL; NORMAN, 1988), o SUS fornece resultados confiáveis sobre a percepção global de usabilidade com uma estrutura objetiva de dez itens.

Além da eficiência estatística, o SUS permite a transformação de respostas subjetivas em um escore numérico padronizado, facilitando a classificação da interface através de escalas de adjetivos validadas, como a proposta por Bangor, Kortum e Miller (2009). Esta abordagem quantitativa assegura que os resultados obtidos com o grupo heterogêneo de usuários possam ser comparados diretamente com médias de mercado estabelecidas em estudos de larga escala, conferindo validade estatística à interpretação do desempenho do sistema.

O instrumento de coleta de dados constitui-se de um formulário com dez afirmações, alternando entre aspectos positivos e negativos do sistema. Os participantes expressam sua opinião através da escala de Likert, variando de "Discordo Completamente" (1) a "Concordo Completamente" (5). As questões aplicadas foram:

1. Eu acho que gostaria de usar esse sistema com frequência;
2. Eu acho o sistema desnecessariamente complexo;
3. Eu achei o sistema fácil de usar;
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema;

5. Eu acho que as várias funções do sistema estão muito bem integradas;
6. Eu acho que o sistema apresenta muita inconsistência;
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente;
8. Eu achei o sistema complicado de usar;
9. Eu me senti confiante ao usar o sistema;
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

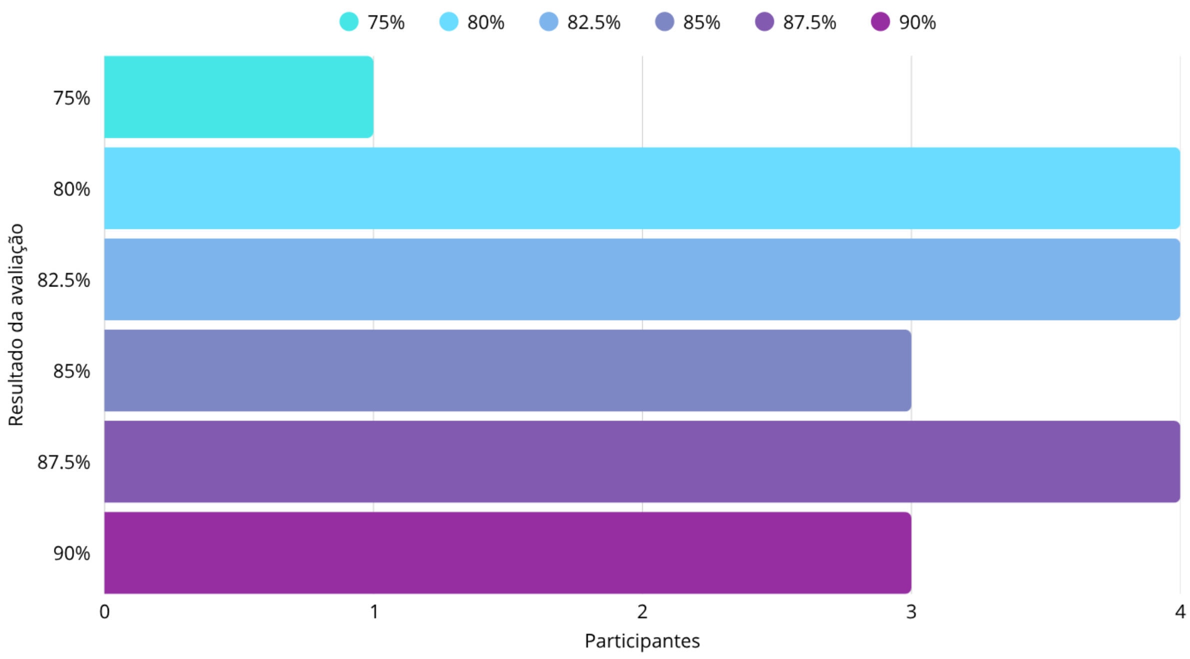
A análise dos dados segue o algoritmo padrão do SUS: subtrai-se 1 da pontuação dada nas questões ímpares, enquanto nas questões pares, subtrai-se a resposta do valor 5. O somatório final é multiplicado por 2,5, resultando em um *score* de usabilidade que varia de 0 a 100. Este valor quantitativo permite classificar a qualidade da experiência do usuário e comparar o desempenho da plataforma com padrões de mercado.

5.2.2 Resultados

A pesquisa contou com a colaboração voluntária de 19 participantes, compondo um grupo heterogêneo formado por estudantes da área de tecnologia e comerciantes de pequeno porte. Essa amostragem permitiu submeter a aplicação a diferentes perfis de navegação, enriquecendo os dados coletados sobre a interação com a *interface*.

Os resultados detalhados das avaliações estão apresentados na figura 20, que ilustra os resultados da pesquisa. Após a tabulação dessas respostas e a aplicação do algoritmo de cálculo do SUS, o sistema obteve uma pontuação média final de, aproximadamente, 84 pontos na escala de 0 a 100. Ao analisar este desempenho à luz da escala de adjetivos proposta por Bangor et al. (2009), o resultado classifica a usabilidade da plataforma como boa, posicionando o *software* no terceiro quartil de aceitação. Este índice, superior à média de mercado de 68 pontos, indica que a solução foi bem recebida pelos usuários, que não encontraram barreiras significativas para a conclusão das tarefas propostas, validando a efetividade da *interface* desenvolvida para o público-alvo.

Figura 22 - Gráfico do resultado da pesquisa SUS



Fonte: Próprio Autor

Uma análise qualitativa por item revela que o sistema se destacou positivamente em termos de integridade. Os participantes indicaram, de forma expressiva, que as diversas funções estão bem integradas e que a *interface* apresenta baixa inconsistência. Por outro lado, aproximadamente 20% a 25% das respostas apresentaram caráter neutro em questões relacionadas à complexidade percebida e à necessidade de aprendizado prévio. Esse comportamento sugere que, embora o sistema seja funcional, a curva de aprendizado inicial ainda pode ser otimizada para usuários menos familiarizados com ferramentas de e-commerce. Além disso, a frequência de uso pretendida também apresentou oscilação nessa mesma faixa percentual, o que pode estar atrelado ao estágio da aplicação, que foca no essencial e ainda não contempla todas as funcionalidades avançadas de retenção de um produto maduro.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento deste trabalho permitiu a concretização de uma plataforma de *e-commerce*, validando a hipótese de que é possível oferecer uma solução tecnológica robusta e acessível para pequenos comerciantes. A utilização de tecnologias modernas como TypeScript, Next.js e Node.js, aliada a uma arquitetura modular, resultou em um sistema que atende aos requisitos fundamentais de gestão de vendas e personalização da identidade visual da loja. Conclui-se, portanto, que a aplicação entrega valor ao simplificar a inserção digital de empreendedores, oferecendo uma ferramenta funcional que remove barreiras técnicas iniciais.

Como sugestão para trabalhos futuros, visando a evolução da plataforma, propõe-se a implementação de uma suíte de testes automatizados unitários e de integração para assegurar a estabilidade do código a longo prazo. Além disso, identifica-se a necessidade de enriquecer o módulo de produtos através da inclusão de campos dinâmicos para fichas técnicas detalhadas. No que tange à expansão das funcionalidades de venda e interação, recomenda-se a integração direta com *gateways* de pagamento para automatizar a liquidação financeira, além do desenvolvimento de um sistema de comentários e avaliações. Estas implementações permitirão que o cliente final forneça *feedbacks* sobre os produtos, aumentando a prova social e a confiabilidade da loja virtual.

REFERÊNCIAS

ABCOMM. **Crescimento do e-commerce brasileiro.** [S. l.]: ABCOMM, [202?]. Disponível em: https://dados.abcomm.org/crescimento-do-ecommerce-brasileiro. Acesso em: 12 jul. 2025.

BANGOR, A.; KORTUM, P.; MILLER, J.; **Determining what individual SUS scores mean: adding an adjective rating scale**. J. Usability Studies 4, 3, 114–123. 2009.

BROOKE, J. SUS: **A 'Quick and Dirty' Usability Scale**. In: JORDAN, P. W. et al. (eds.). Usability Evaluation in Industry. London: Taylor & Francis, 1996.

CHIN, J. P.; DIEHL, V. A.; NORMAN, K. L. **Development of an instrument measuring user satisfaction of the human-computer interface**. In: SIGCHI Conference on Human Factors in Computing Systems. ACM, 1988. p. 213-218.

CHISNALL, David. **The JavaScript Engine Family Tree**. Queue, v. 16, n. 3, p. 10-31, 2018.

EKPOBIMI, Harrison Oke. **Building high-performance web applications with NextJS.** *Computer Science & IT Research Journal*, [S. l.], v. 5, n. 8, p. 1963-1977, ago. 2024. DOI: 10.51594/csitrj.v5i8.1459. Disponível em: http://dx.doi.org/10.51594/csitrj.v5i8.1459. Acesso em: 22 jun. 2025.

EMMANNI, Phani Sekhar. **The Role of TypeScript in Enhancing Development with Modern JavaScript Frameworks.** *International Journal of Science and Research (IJSR)*, [S. l.], v. 10, n. 2, p. 1738-1741, fev. 2021. Disponível em: https://www.researchgate.net/publication/380361058. Acesso em: 22 jun. 2025.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 4. ed. São Paulo: Atlas, 2002.

GIT. About Git. [S. l.]: Git, 2025. Disponível em: https://git-scm.com/about. Acesso em: 30 jun. 2025.

GITHUB. About GitHub. [S. l.]: GitHub, 2025. Disponível em: https://github.com/about. Acesso em: 30 jun. 2025.

KARIMI, Jahangir; WALTER, Zhiping. **Corporate Entrepreneurship, Disruptive Business Model Innovation Adoption, and Its Performance: The Case of the Newspaper Industry**. *Long Range Planning*, Oxford, v. 49, n. 3, p. 342-360, jun. 2016.

MICROSOFT. **Documentation for Visual Studio Code**. [S. l.]: Microsoft, 2025. Disponível em: https://code.visualstudio.com/docs. Acesso em: 30 jun. 2025.

NIELSEN, J. **Usability Engineering**. San Diego: Academic Press, 1994.

ROGERS, David L. **Transformação digital:** a estratégia para a era digital. São Paulo: Autêntica Business, 2017.

SOUSA, Aislan Rafael Rodrigues de. **Processo Projeto Integrador III**. Material de aula. Picos: Instituto Federal do Piauí, 2023.

TURBAN, Efraim et al. **Electronic Commerce 2018:** a managerial and social networks perspective. 9. ed. Cham: Springer, 2017.

WHARTON**,** C. et al. **The cognitive walkthrough method: A practitioner's guide.** In: NIELSEN, J.; MACK, R. L. (eds.). Usability Inspection Methods. New York: John Wiley & Sons, 1994.

WODYK, Rafał; SKUBLEWSKA-PASZKOWSKA, Maria. **Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework.** *Journal of Computer Sciences Institute*, [S. l.], v. 17, p. 358-364, 2020. DOI: 10.47750/eibg.2021.27.03.090. Acesso em: 22 jun. 2025.