INSTITUTO FEDERAL DE RONDÔNIA
CAMPUS PORTO VELHO CALAMA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DANIEL VINÍCIUS TORRES VIANA, LUKAS PINHEIRO DA SILVA, LUIS MARCELO FABRICIO GUIMARÃES

DESENVOLVIMENTO DE UMA PLATAFORMA DE E-COMMERCE MULTITENANT COM GATEWAYS DE PAGAMENTO INTEGRADOS

Porto Velho/RO
2025

DANIEL VINÍCIUS TORRES VIANA, LUKAS PINHEIRO DA SILVA, LUIS MARCELO FABRICIO GUIMARÃES

DESENVOLVIMENTO DE UMA PLATAFORMA DE E-COMMERCE MULTITENANT COM GATEWAYS DE PAGAMENTO INTEGRADOS

Artigo entregue como Trabalho de Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), *Campus* Porto Velho Calama, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Leandro Ferrarezi Valiante

Porto Velho/RO
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Viana, Daniel Vinícius Torres.
Desenvolvimento de uma plataforma de e-commerce multitenant com gateways de pagamento integrados / Daniel Vinícius Torres Viana, Lukas Pinheiro da Silva, Luis Marcelo Fabricio Guimarães. - Porto Velho, 2025.
29 f. : il.

Orientador(a): Prof. Leandro Ferrarezi Valiante.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Porto Velho, 2025.

1. Software como serviço. 2. Arquitetura multicliente. 3. Pequenos e médios varejistas. 4. Segurança da informação. I. Silva, Lukas Pinheiro da. II. Valiante, Leandro Ferrarezi (orient.). III. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. IV. Título.

Bibliotecário(a) Responsável: Miria Santana Veiga, CRB-11/898

DANIEL VINÍCIUS TORRES VIANA, LUKAS PINHEIRO DA SILVA, LUIS MARCELO FABRICIO GUIMARÃES

DESENVOLVIMENTO DE UMA PLATAFORMA DE E-COMMERCE MULTITENANT COM GATEWAYS DE PAGAMENTO INTEGRADOS

A banca examinadora, abaixo listada, aprova o Trabalho de Conclusão de Curso “DESENVOLVIMENTO DE UMA PLATAFORMA DE E-COMMERCE MULTITENANT COM GATEWAYS DE PAGAMENTO INTEGRADOS” elaborado por “DANIEL VINÍCIUS TORRES VIANA, LUKAS PINHEIRO DA SILVA, LUIS MARCELO FABRICIO GUIMARÃES” como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas, pelo Instituto Federal de Educação, Ciência e Tecnologia de Rondônia.

Porto Velho/RO, 01/12/2025

Comissão Examinadora

gov.br
Documento assinado digitalmente
LEANDRO FERRAREZI VALIANTE
Data: 08/12/2025 17:47:56-0300
Verifique em https://validar.iti.gov.br

Prof. Leandro Ferrarezi Valiante - IFRO
(Orientador)

gov.br
Documento assinado digitalmente
WESLLEY MICHEL SILVA BOLSONI
Data: 08/12/2025 12:39:26-0300
Verifique em https://validar.iti.gov.br

Prof. Weslley Bolsoni - IFRO

gov.br
Documento assinado digitalmente
ROSANGELA SILVA CUNHA
Data: 08/12/2025 17:36:13-0300
Verifique em https://validar.iti.gov.br

Prof. Rosângela Silva Cunha - SAPIENS

DESENVOLVIMENTO DE UMA PLATAFORMA DE E-COMMERCE MULTITENANT COM GATEWAYS DE PAGAMENTO INTEGRADOS

RESUMO: No contexto de expansão do comércio eletrônico, pequenos e médios negócios ainda dependem de soluções fragmentadas ou de plataformas genéricas, com baixa flexibilidade e pouco controle sobre seus dados. Este trabalho apresenta o desenvolvimento do BROS, sistema web em modelo *Software as a Service (SaaS)* que opera múltiplas lojas virtuais em ambiente compartilhado, com isolamento lógico entre clientes e integração a serviços externos de pagamento. A solução foi construída com tecnologias da plataforma .NET e princípios de arquitetura modular, combinando *Domain-Driven Design* e *Clean Architecture*. A metodologia envolveu levantamento de requisitos, prototipação de interfaces e desenvolvimento iterativo com práticas ágeis. Como resultado, o sistema centraliza o cadastro e o controle de produtos, estoques e transações de diferentes lojas, reduzindo retrabalho operacional e oferecendo uma base estruturada para futuras extensões funcionais da plataforma.

PALAVRAS-CHAVE: software como serviço. arquitetura multicliente. pequenos e médios varejistas. segurança da informação. transações financeiras online. gestão de lojas virtuais.

ABSTRACT: In the context of expanding electronic commerce, small and medium-sized businesses still rely on fragmented solutions or generic platforms, with low flexibility and limited data control. This work presents the development of BROS, a web system in the *Software as a Service (SaaS)* model that operates multiple virtual stores in a shared environment, ensuring logical isolation between clients and integration with external online payment services. The solution was built with .NET technologies and modular architectural principles, combining *Domain-Driven Design* and *Clean Architecture*. The methodology involved requirements elicitation, interface prototyping and iterative development with agile practices. As a result, the system centralizes the registration and control of products, inventory and transactions across different stores, reducing operational rework and providing a structured basis for future functional extensions of the platform.

KEYWORDS: software as a service. multi-tenant architecture. small and medium-sized retailers. information security. online financial transactions. online store management.

1 INTRODUÇÃO

O comércio eletrônico tem experimentado crescimento exponencial nas últimas décadas, transformando profundamente a forma como empresas e consumidores interagem no ambiente de negócios (Albertin; Moura, 1998). No Brasil, esse avanço foi fortemente impulsionado após 2020, devido à digitalização acelerada, ao isolamento social durante a pandemia de COVID-19 e às novas demandas por conveniência e agilidade. Segundo a Associação Brasileira de Comércio Eletrônico (ABCOMM, 2021), a base de usuários do *e-commerce* no país cresceu mais de 70% em 2020, um patamar que em condições normais levaria cerca de dez anos para ser atingido. Esse cenário revelou novas oportunidades, mas também desafios para pequenas e médias empresas (PMEs), especialmente relacionados aos altos custos de infraestrutura tecnológica e à complexidade de se adotar sistemas robustos e seguros (Stallings, 2017).

A arquitetura *multitenant* surge, portanto, como solução estratégica para democratizar o acesso a plataformas de *e-commerce* modernas. Conforme Bezemer e Zaidman (2010), esse modelo permite que múltiplos clientes (*tenants*) compartilhem uma mesma instância de aplicação e banco de dados, mantendo um alto grau de configurabilidade individual, como se operassem em ambientes dedicados. Os principais benefícios desse modelo incluem melhor utilização de recursos de *hardware* e facilidade de manutenção, resultando em redução de custos operacionais.

Além da questão de escala e custos, pequenos e médios lojistas também enfrentam dificuldades na gestão integrada de múltiplas lojas virtuais, catálogos e estoques. Em muitos casos, cada canal utiliza uma solução distinta, com pouca padronização e baixa visibilidade sobre pedidos e transações, o que aumenta retrabalho e o risco de inconsistências nos registros.

A segurança das transações eletrônicas representa outro pilar essencial para a confiabilidade no comércio *online*. Albertin e Moura (1998) destacam que fatores como segurança, privacidade e autenticação são pré-requisitos críticos para o pleno funcionamento do comércio eletrônico. Nesse contexto, a integração de *gateways* de pagamento seguindo normas como o Padrão de Segurança de Dados da Indústria de Cartões de Pagamento (PCI-SSC, 2025) se torna vital, pois esse padrão estabelece um conjunto internacional de requisitos criados pelas principais bandeiras de cartão para garantir a proteção de dados sensíveis durante o processamento, transmissão e armazenamento. A adoção dessas práticas fortalece a segurança das transações, utilizando recursos como criptografia, tokenização e autenticação de múltiplos fatores.

Diante desse panorama, este trabalho apresenta o desenvolvimento da plataforma “BROS”, um sistema de *e-commerce multitenant* cuja proposta se estrutura em três pilares tecnológicos interdependentes: (i) arquitetura escalável *multitenant* com

isolamento lógico por locatário; (ii) integração segura com *gateways* de pagamento em conformidade com boas práticas de segurança; e (iii) um painel administrativo multiloja para gestão de produtos, estoques, categorias, imagens, usuários e personalização visual das lojas. O sistema foi construído com tecnologias modernas do ecossistema .NET, aplicando os princípios de *Domain-Driven Design* (DDD) e *Clean Architecture*, o que favorece manutenibilidade, modularidade e evolução contínua da aplicação. A problemática central abordada consiste em como desenvolver uma plataforma de *e-commerce* que seja simultaneamente escalável, segura, adaptável e economicamente viável para pequenos e médios varejistas no contexto brasileiro. A hipótese considerada é que a combinação entre *multitenancy*, um painel administrativo multiloja e a integração segura com meios de pagamento pode resultar em uma solução eficaz, competitiva e alinhada às reais necessidades desses lojistas.

1.1 Objetivos

1.1.1 Objetivo geral

O presente trabalho tem como objetivo geral desenvolver uma plataforma de *e-commerce multitenant* escalável, integrando *gateways* de pagamento seguros e disponibilizando um painel administrativo multiloja para gestão de produtos, estoques, categorias, imagens, usuários e personalização visual, a fim de criar uma solução robusta, modular, manutenível e adaptável às demandas de múltiplos lojistas virtuais.

1.1.2 Objetivos específicos

Em busca de atingir o objetivo geral, foram elencados estes objetivos específicos:

(i) Implementar arquitetura *multitenant* com banco de dados compartilhado e isolamento lógico por *TenantId*, assegurando segurança dos dados, escalabilidade e redução de custos operacionais;

(ii) Estruturar o painel administrativo com módulos para gerenciamento de produtos, categorias, estoques, imagens e usuários, bem como recursos de personalização visual das lojas, garantindo autonomia operacional aos locatários;

(iii) Integrar *gateways* de pagamento em conformidade com o padrão PCI-DSS, empregando criptografia, tokenização e autenticação para proteger dados sensíveis durante as transações;

(iv) Aplicar os princípios de *Domain-Driven Design* (DDD) e *Clean Architecture* para estruturação modular do código, com separação entre camadas de domínio, aplicação, infraestrutura e apresentação;

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos teóricos, conceitos e tecnologias que embasam o desenvolvimento do sistema “BROS”. Organizam-se aqui os elementos relacionados ao processo de engenharia de software, levantamento e modelagem de requisitos, prototipação de interfaces, arquitetura adotada, padrões de projeto e tecnologias utilizadas.

2.1 Engenharia de Software

A engenharia de software fornece métodos e práticas sistemáticas para planejar, organizar e controlar o desenvolvimento de sistemas. Um processo bem definido abrange desde o levantamento de requisitos até a manutenção, passando por projeto e implementação. Conforme Pressman e Maxim (2021), processos estruturados reduzem retrabalho, melhoram a comunicação entre os envolvidos e contribuem para a qualidade do produto final. Nesse contexto, abordagens iterativas e metodologias ágeis, como Scrum e Kanban, permitem entregas incrementais de software e adaptação contínua às mudanças de necessidades e de escopo.

O levantamento de requisitos busca identificar as necessidades dos usuários e as funcionalidades que o sistema deve fornecer, por meio de técnicas de elicitação como entrevistas com *stakeholders*, análise de documentos e definição de *user stories*. Requisitos bem definidos orientam as etapas seguintes e evitam ambiguidades no produto final Pressman e Maxim (2021), incluindo tanto requisitos funcionais (por exemplo, operações de venda, cadastro e acompanhamento de pedidos) quanto requisitos não funcionais, como desempenho, segurança e usabilidade. A prototipação de interfaces complementa essa etapa ao materializar fluxos e telas em artefatos visuais, permitindo validar antecipadamente a experiência do usuário e ajustar o desenho da solução antes da implementação Pressman e Maxim (2021).

No contexto de comércio eletrônico, a definição de requisitos de negócio envolve aspectos como gestão de catálogo, controle de estoque, políticas de frete, meios de pagamento e atendimento ao cliente. Neste trabalho, o foco recai especialmente sobre requisitos relacionados à arquitetura *multitenant*, à segurança das transações financeiras e à gestão operacional das lojas virtuais, que orientam diretamente as decisões de projeto da plataforma BROS.

2.2 Arquitetura de Software

A arquitetura de software define a estrutura geral de um sistema e organiza como seus componentes se relacionam. Uma boa arquitetura favorece modularidade, baixo acoplamento e facilidade de manutenção. Entre as abordagens utilizadas na literatura destacam-se o *Domain-Driven Design* (DDD), também conhecido como design orientado ao domínio, e a Arquitetura Limpa (*Clean Architecture*), que oferecem princípios para construção de sistemas consistentes e escaláveis Martin (2019).

O DDD enfatiza que o software deve ser estruturado a partir do domínio do problema, priorizando regras de negócio e uma linguagem comum entre equipe técnica e especialistas. Segundo Evans (2004), essa abordagem aproxima modelagem e implementação e contribui para maior clareza e consistência das regras do sistema.

A Arquitetura Limpa organiza o software em camadas, mantendo as regras de negócio no centro da aplicação e posicionando elementos de infraestrutura em níveis externos. Para Martin (2019), essa separação aumenta a independência tecnológica e facilita futuras adaptações, uma vez que o núcleo da aplicação permanece protegido de mudanças externas.

Em ambientes de computação em nuvem e aplicações no modelo *Software as a Service* (*SaaS*), esses princípios arquiteturais são frequentemente combinados com o paradigma *multitenant*, no qual múltiplos clientes compartilham uma mesma aplicação preservando o isolamento lógico de seus dados Bezemer e Zaidman (2010). Esse modelo reduz custos de infraestrutura e simplifica a manutenção centralizada, ao mesmo tempo em que impõe desafios adicionais relacionados à segurança, à configuração específica de cada locatário e ao desenho de mecanismos de separação de contexto em nível de dados e de negócio.

Em conjunto, essas abordagens fornecem fundamentos teóricos que favorecem a construção de sistemas modulares e preparados para evoluir de forma contínua.

2.3 Tecnologias utilizadas

Esta seção apresenta as principais tecnologias empregadas no desenvolvimento da plataforma. Em aplicações web modernas, costuma-se organizar a solução em duas camadas principais: *front-end*, responsável pela interface e interação com o usuário, e *back-end*, responsável pelo processamento das regras de negócio, acesso a dados e integração com serviços externos. Dessa forma, as tecnologias descritas a seguir são apresentadas de acordo com o papel que desempenham dentro da arquitetura do sistema.

No *front-end*, utilizou-se o HTML (*HyperText Markup Language*), que constitui a

base estrutural das páginas web, organizando os elementos que compõem a interface conforme a especificação mantida pelo WHATWG (W3C, 2025). O CSS (*Cascading Style Sheets*) é empregado para definir estilos visuais, permitindo controle sobre cores, espaçamentos, tipografias e layouts, conforme definido pelo CSS – W3C (2025). Para aprimorar a responsividade e padronização visual, adotou-se o *Bootstrap 5*, biblioteca que fornece componentes reutilizáveis construídos sobre HTML, CSS e JavaScript. O JavaScript, por sua vez, é uma linguagem amplamente utilizada para adicionar dinamismo às interfaces e controlar comportamentos interativos (ECMA International, 2025). Complementarmente, o mecanismo *Razor*, disponibilizado pelo *framework* ASP.NET Core, permite integrar código C# à marcação HTML, possibilitando a renderização dinâmica de conteúdos enviados pelo servidor, atuando como elo entre o *front-end* e o *back-end*.

Já no *back-end*, empregou-se a plataforma .NET, um ecossistema de desenvolvimento mantido pela Microsoft e composto por ferramentas, bibliotecas e *runtimes* para construção de aplicações web, desktop e serviços distribuídos (Microsoft, 2025a). Dentro desse ecossistema, utiliza-se a linguagem C#, tipada e orientada a objetos, amplamente empregada em soluções corporativas devido ao desempenho e à vasta coleção de bibliotecas disponíveis (Microsoft, 2025b). Sobre essa base tecnológica, o ASP.NET Core MVC funciona como *framework* principal da aplicação, seguindo o padrão arquitetural *Model–View–Controller* e oferecendo suporte nativo para recursos como injeção de dependências, autenticação, segurança, roteamento e criação de APIs (Microsoft, 2025a). Uma API (*Application Programming Interface*) consiste em um conjunto de regras que define como diferentes sistemas se comunicam, permitindo a troca estruturada de informações (Microsoft, 2025d).

Para acesso e manipulação dos dados, utilizou-se o Entity Framework Core, um mapeador objeto-relacional (*Object–Relational Mapper*) que possibilita a interação com bancos de dados relacionais por meio de classes e consultas tipadas, reduzindo a necessidade de comandos SQL explícitos (Microsoft, 2025). Entre seus recursos destacam-se o uso do padrão *Code First*, migrações automáticas e filtros globais de consulta, aspectos úteis para a organização do domínio e para o isolamento lógico dos dados em contextos *multitenant*.

O armazenamento das informações é realizado no SQL Server, um Sistema Gerenciador de Banco de Dados (SGBD) relacional que oferece suporte a transações ACID, integridade referencial, controle de concorrência e mecanismos de otimização (Microsoft, 2025c). No contexto acadêmico deste trabalho, utiliza-se a edição SQL Server Express, disponibilizada gratuitamente pela Microsoft e suficiente para o volume de dados e carga de uso do protótipo. Em um cenário de produção, a mesma arquitetura será implantada sobre edições licenciadas do SQL Server ou serviços gerenciados

equivalentes, dimensionados de acordo com os requisitos de escala, custo e infraestrutura de cada implantação.

Por fim, a plataforma integra o VLibras, ferramenta gratuita e de código aberto desenvolvida pelo Governo Federal que realiza tradução automática de conteúdos digitais para a Língua Brasileira de Sinais (LIBRAS). A utilização desse recurso amplia a acessibilidade da aplicação e a alinha às diretrizes nacionais de inclusão digital (VLibras, 2025).

3 METODOLOGIA

Este trabalho caracteriza-se como uma pesquisa aplicada, pois envolve o desenvolvimento de um sistema web de comércio eletrônico voltado ao atendimento de necessidades concretas de pequenos e médios lojistas. Quanto aos objetivos, enquadra-se como pesquisa exploratória e descritiva, uma vez que analisa referências existentes, identifica requisitos e descreve de forma sistemática o processo de construção da solução.

A abordagem adotada é predominantemente qualitativa, com elementos quantitativos pontuais. A dimensão qualitativa está presente na interpretação das demandas de negócio, na análise das interfaces e na definição de requisitos. A dimensão quantitativa aparece em observações simples do comportamento do sistema, como registro do fluxo em cenários de navegação típicos e confirmação do comportamento dos principais fluxos sob diferentes volumes de dados. (Pressman; Maxim, 2021; Schwaber; Sutherland, 2020).

A pesquisa foi desenvolvida em ambiente acadêmico, no contexto do curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Rondônia (IFRO). Os requisitos da plataforma foram definidos a partir da evolução do sistema acadêmico GymBros, um *e-commerce monotenant* de suplementos alimentares, para a visão de produto *SaaS* multiloja, da análise de lojas virtuais de referência no segmento de suplementos, como Growth Supplements e Max Titanium, e das discussões com o orientador. As demandas de negócio consideradas decorrem, portanto, da combinação entre o estudo desses sistemas de referência e a definição conjunta de requisitos pela equipe de desenvolvimento e pelo orientador, não havendo participação direta de lojistas reais na etapa de levantamento.

Nesse processo foram identificadas funcionalidades essenciais de catálogo, carrinho e *checkout*, bem como necessidades específicas de arquitetura *multitenant*, gestão centralizada de múltiplas lojas e integração com meios de pagamento. Esses elementos orientaram as fases de desenvolvimento descritas a seguir e foram tomados como base para os procedimentos de avaliação utilizados ao longo do projeto.

3.1 Processo de desenvolvimento

O desenvolvimento da plataforma seguiu um processo iterativo e incremental, organizado em ciclos de planejamento, implementação e revisão, com iterações curtas e entregas frequentes ao longo do projeto. Em cada ciclo eram definidas as funcionalidades a serem implementadas, estimadas as tarefas no Jira e planejadas entregas parciais para a equipe.

Na etapa inicial, foi definida a arquitetura da solução com base nas abordagens *Domain-Driven Design* (DDD) (Evans, 2004) e *Clean Architecture* (Martin, 2019). Nessa fase foram especificadas as camadas de domínio, aplicação, infraestrutura e apresentação (Web), além da separação em projetos dentro da solução. Em seguida, configuraram-se o repositório no GitHub, o ambiente de desenvolvimento e o fluxo de versionamento, com uso de *branches* específicas para cada incremento.

Com a arquitetura estabelecida, os ciclos iniciais concentraram-se na implementação de um conjunto básico de funcionalidades, incluindo autenticação de usuários, cadastro de produtos e fluxo de compra em um único contexto de loja. A partir dessa base, o projeto evoluiu em fases sucessivas para incorporar suporte a múltiplas lojas de forma isolada, integração com *gateways* de pagamento e ampliação do painel administrativo (módulos de produtos, estoque, categorias, imagens, usuários, lojas e personalização), mantendo o mesmo processo incremental de planejamento das alterações, implementação em *branches* dedicadas, revisão de código entre os integrantes e integração ao ramo principal após validação.

Ao final de cada ciclo, as entregas eram revisadas pela equipe com apoio dos protótipos elaborados no Figma e do quadro de tarefas no Jira, o que permitiu ajustar prioridades, registrar pendências e documentar a evolução do sistema. As fases de evolução da plataforma foram planejadas de modo a contemplar os objetivos específicos definidos na introdução, que incluem a definição da arquitetura *multitenant*, a consolidação do painel administrativo multiloja e a integração com *gateways* de pagamento em conformidade com requisitos de segurança.

3.2 Procedimentos de avaliação

A solução envolveu diferentes procedimentos, alinhados às funcionalidades implementadas em cada etapa.

Inicialmente, foram realizados procedimentos funcionais manuais contemplando cenários de sucesso e de erro nos fluxos de cadastro, autenticação, gerenciamento de produtos, carrinho de compras e finalização de pedidos. Esses procedimentos buscavam confirmar se cada funcionalidade atendia aos requisitos definidos e se os fluxos

principais de navegação se comportavam de forma consistente.

As interfaces desenvolvidas foram comparadas sistematicamente com os protótipos construídos no Figma, analisando aderência visual, organização das informações e coerência dos caminhos de navegação. Sempre que identificadas divergências relevantes entre o protótipo e a implementação, eram registrados ajustes para os ciclos seguintes.

Nos módulos administrativos (produtos, estoque, categorias, imagens, usuários e gestão de lojas), a análise concentrou-se em confirmar se as operações de criação, atualização e exclusão eram refletidas corretamente tanto na base de dados quanto na interface pública da loja. Para isso, foram executados casos de uso em que se alteravam preços, descrições, imagens e quantidades em estoque, observando-se o impacto imediato na vitrine da loja e no fluxo de compra. Esses procedimentos permitiram identificar inconsistências de interface e ajustar mensagens, rótulos de campos e regras de negócio associadas ao cadastro.

Por fim, na integração com gateways de pagamento, foram executados procedimentos de ponta a ponta em ambientes de homologação dos provedores, incluindo a iniciação do pagamento a partir do checkout da plataforma, o redirecionamento para as páginas externas de cobrança, o retorno à aplicação após a conclusão e o registro do status do pedido. Esses procedimentos consideraram também o tratamento de erros e falhas de comunicação, buscando garantir um fluxo de compra consistente e alinhado às recomendações de segurança estabelecidas pelo padrão PCI-DSS (PCI-SSC, 2025).

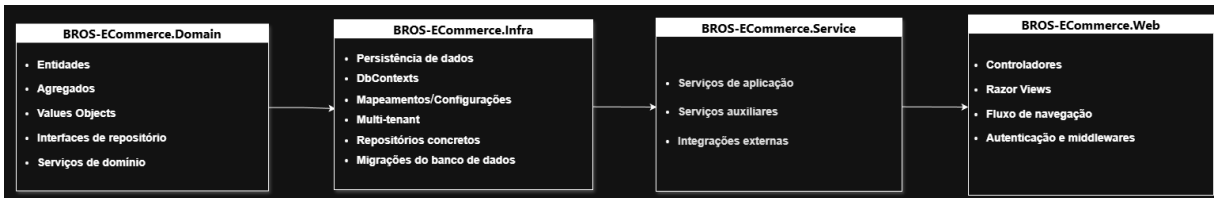
Os resultados dessas análises foram registrados e utilizados para orientar correções, refinos de interface e ajustes nas regras de negócio ao longo dos ciclos de desenvolvimento.

4 DESENVOLVIMENTO

O desenvolvimento da plataforma BROS evoluiu a partir de um e-commerce acadêmico de suplementação (GymBros) para uma solução multiloja em modelo Software as a Service (SaaS). Essa transição exigiu a reestruturação da arquitetura, do modelo de dados e dos principais fluxos do sistema. A implementação seguiu os requisitos definidos e os conceitos da fundamentação teórica, adotando organização em camadas baseada em Clean Architecture e Domain-Driven Design, com separação explícita por locatário. O trabalho foi conduzido de forma iterativa, com ciclos de planejamento, codificação e revisão, assegurando escalabilidade, segurança e suporte a pequenos e médios lojistas.

A estrutura da plataforma foi organizada utilizando princípios de Clean Architecture e Domain-Driven Design, distribuída em quatro camadas principais: Domínio, Infraestrutura, Serviços de Aplicação e Apresentação (Web). Essa organização visa reduzir acoplamento, aumentar coesão e permitir evolução independente dos módulos do sistema, mantendo as regras de negócio protegidas de detalhes de tecnologia. A Figura 1 apresenta o diagrama de camadas adotado na plataforma BROS, evidenciando a separação entre responsabilidades e o fluxo de dependências entre os componentes.

Figura 1 – Diagrama de camadas

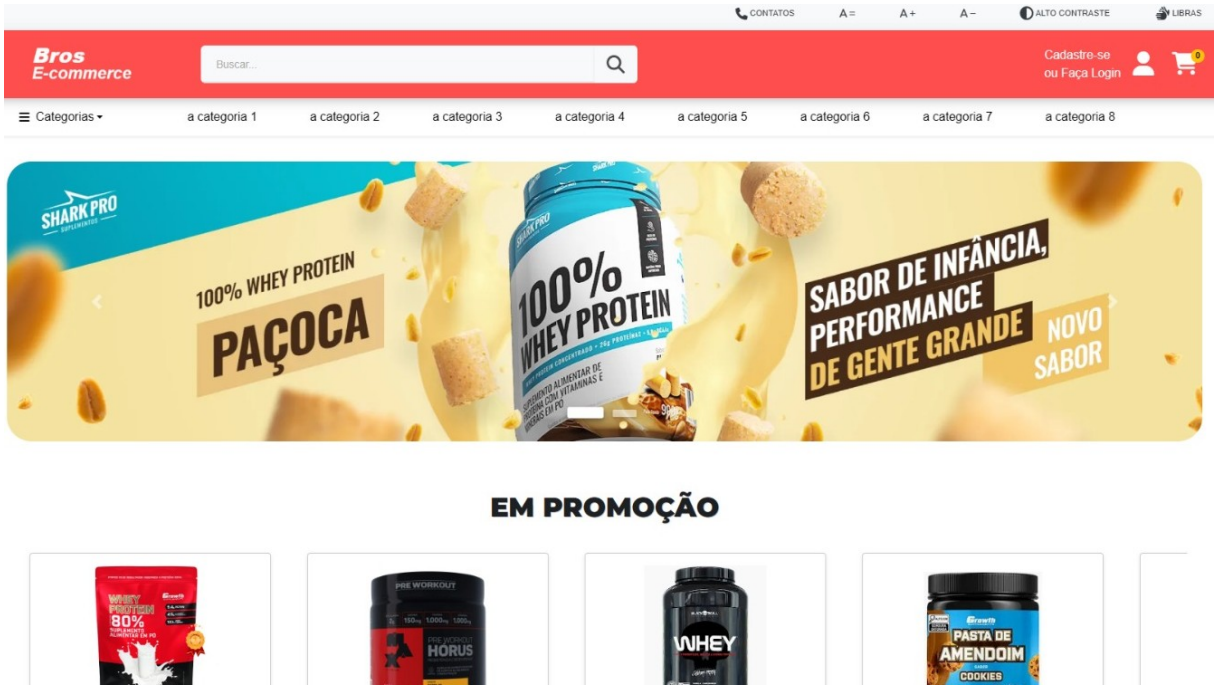


Fonte: Acervo dos autores

4.1 Página inicial

A página inicial da plataforma, apresentada na Figura 2, corresponde ao primeiro ponto de contato do usuário com a loja virtual de cada locatário. Nessa tela, o sistema exibe o cabeçalho com a identidade visual configurada para a loja (nome, logotipo e elementos de navegação) e o acesso ao fluxo de autenticação por meio do botão “Cadastre-se ou Faça Login”, que direciona o usuário para a tela de identificação. A partir dessa página o visitante visualiza os produtos disponibilizados pelo lojista, organizados em seções que facilitam a descoberta de itens, com destaque para imagem, nome, preço e ações de navegação para continuidade da jornada de compra, como acessar mais detalhes do produto ou seguir para o carrinho. Essa estrutura cumpre o papel de apresentar rapidamente o propósito da loja e oferecer caminhos claros tanto para exploração do catálogo quanto para o início do processo de compra.

Figura 2 – Página inicial



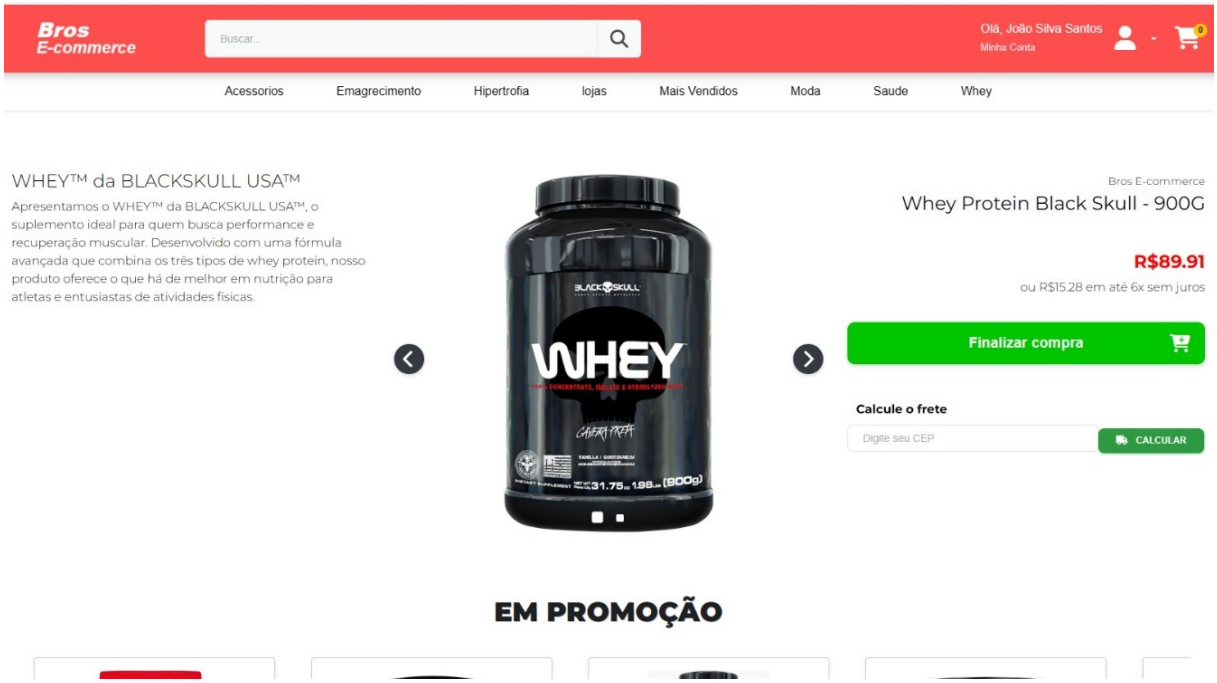
Fonte: Acervo dos autores

4.2 Tela de produto

A tela de produto, apresentada na Figura 3, corresponde ao ambiente em que o usuário obtém informações detalhadas sobre o item selecionado no catálogo. Essa página é acessada a partir da página inicial ou de qualquer listagem de produtos por meio do botão “Comprar”. Ao ser redirecionado, o usuário visualiza a composição completa do produto, incluindo descrição, imagem em destaque, preço, condições de pagamento, variações quando existentes e elementos de persuasão utilizados em páginas de venda.

No centro da tela é exibida a imagem principal, acompanhada de controles que permitem navegar entre outras imagens cadastradas para o mesmo item, facilitando a avaliação visual antes da compra. No lado direito, o sistema apresenta o nome do produto, o preço atual e, quando aplicável, informações de parcelamento. Nessa mesma área encontra-se o botão “Finalizar compra”, que inicia diretamente o fluxo de carrinho e *checkout*, permitindo ao usuário avançar para a etapa de pagamento de forma imediata.

Figura 3 – Tela de produto.



Fonte: Acervo dos autores

A página também disponibiliza um campo para cálculo de frete, no qual o usuário insere o CEP para obter estimativas de prazo e custo de entrega, reforçando a transparência das condições logísticas antes da conclusão da compra. Abaixo da seção principal, o sistema exibe outros produtos em promoção, contribuindo para a navegação contínua dentro do catálogo do lojista.

Assim, a tela de produto desempenha o papel de consolidar todas as informações necessárias para a decisão de compra e estabelece a transição entre a navegação exploratória e a finalização do pedido, mantendo coerência com o fluxo geral da plataforma.

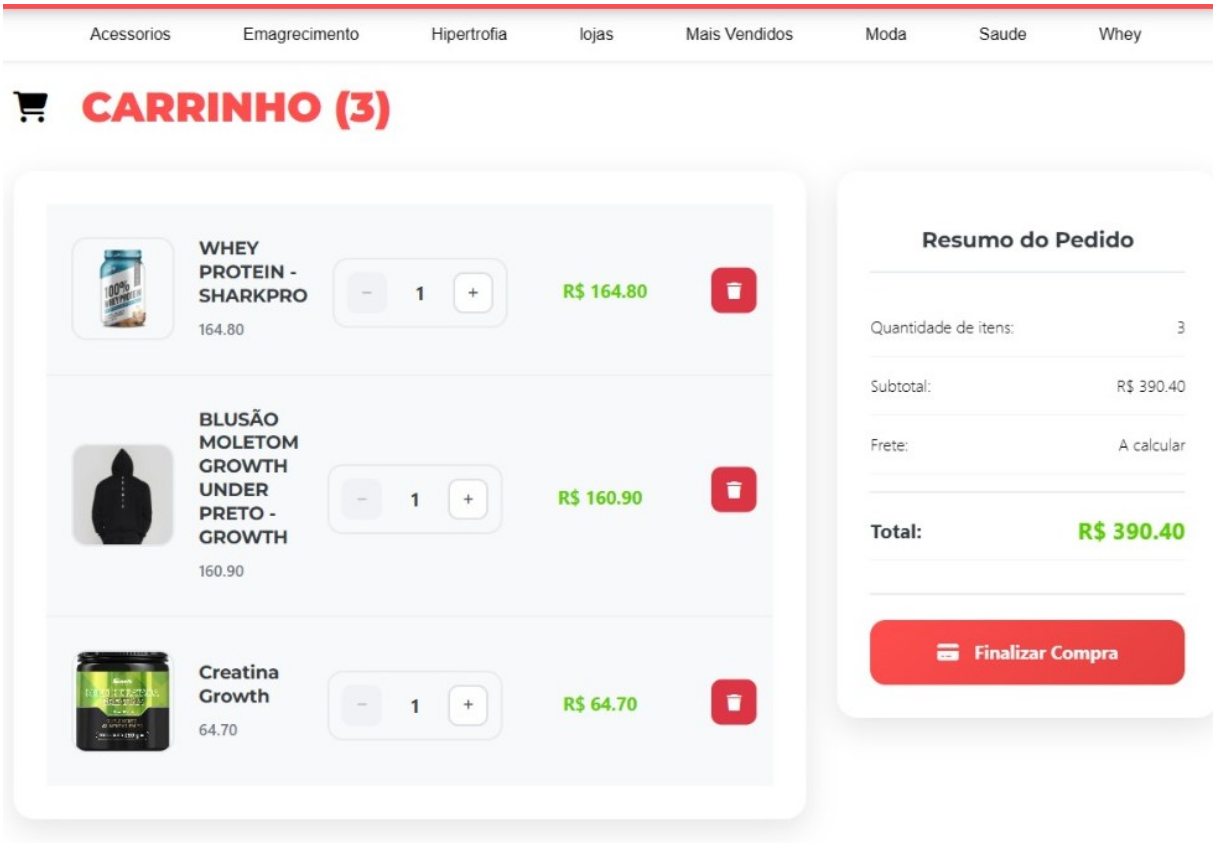
4.3 Carrinho de compras e Módulo de pagamento

A tela de confirmação do pedido, apresentada na Figura 4, é exibida imediatamente após o usuário acionar o botão “Finalizar compra” ao lado do produto. Essa página funciona como uma etapa intermediária entre a revisão dos itens adicionados e o início do fluxo de identificação e preenchimento de dados pessoais.

Nessa interface, o sistema organiza os produtos selecionados em uma estrutura limpa e objetiva, exibindo imagem, nome, quantidade e valor individual de cada item. No painel lateral é apresentado o resumo do pedido, incluindo subtotal, frete a calcular e valor total estimado, mantendo todas as informações essenciais visíveis antes do avanço para as próximas etapas do *checkout*.

A partir dessa tela, o usuário confirma que os itens e valores estão corretos e, então, aciona o botão “Finalizar compra”. Somente após essa confirmação é que o sistema redireciona para o preenchimento dos dados de contato, endereço e, por fim, pagamento.

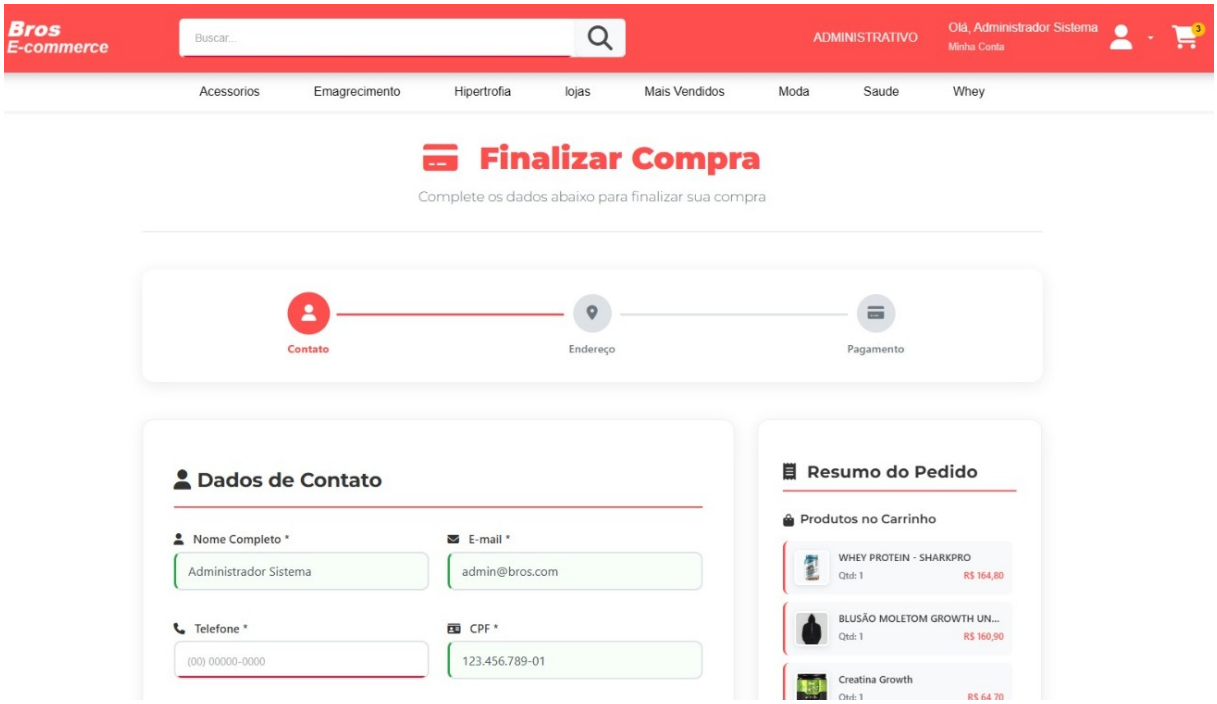
Figura 4 – Etapa de finalização de compra.



Fonte: Acervo dos autores

Ao avançar, o usuário é direcionado à tela de finalização de compra, apresentada na Figura 5. Essa etapa organiza o processo em duas fases: dados de contato e dados de endereço. Inicialmente são solicitadas informações pessoais como nome completo, e-mail, telefone e CPF. Na sequência, o usuário informa os dados de entrega, incluindo CEP, logradouro, número, bairro, cidade e estado. Durante todo o processo, o resumo do pedido permanece visível no painel lateral, facilitando a conferência dos itens antes da etapa de pagamento.

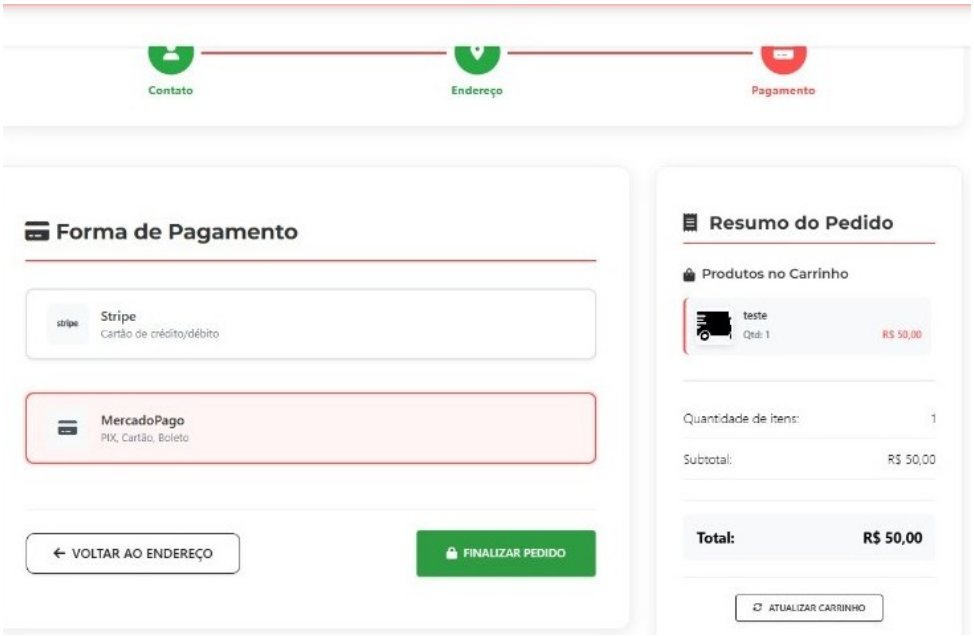
Figura 5 – Tela dos dados de contato



Fonte: Acervo dos autores

A terceira etapa do fluxo de finalização corresponde à escolha da forma de pagamento, apresentada na Figura 6. Nessa tela, o sistema exibe as opções de *gateways* disponíveis, atualmente Stripe e Mercado Pago, cada uma acompanhada de breve descrição dos meios aceitos (cartão, Pix, boleto, entre outros). O usuário pode retornar à etapa anterior pelo botão “Voltar ao endereço” ou concluir a seleção pelo botão “Finalizar pedido”, que efetivamente inicia o processo de cobrança.

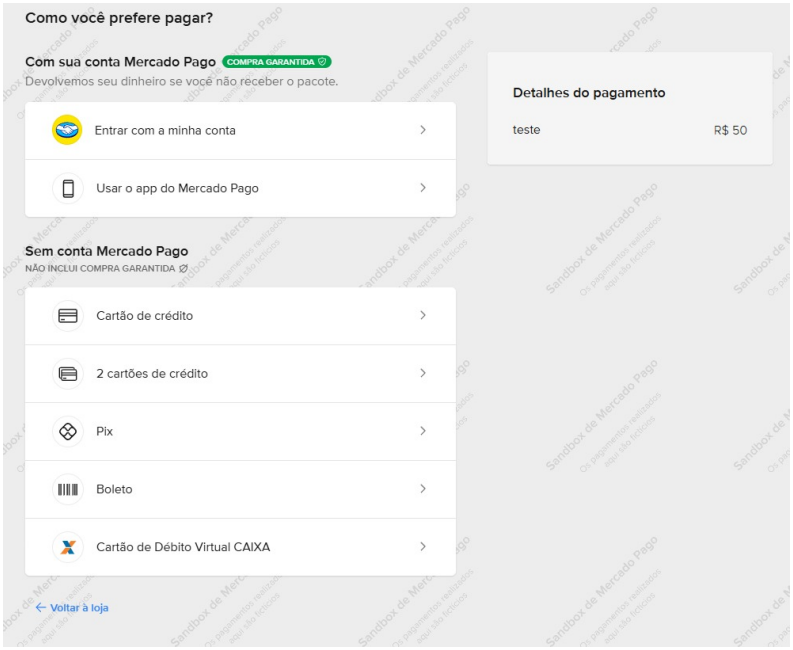
Figura 6 – Tela de escolha da forma de pagamento.



Fonte: Acervo dos autores

Após escolher o *gateway*, o comprador é redirecionado para o ambiente seguro do provedor selecionado. A Figura 7 apresenta a interface do Mercado Pago, na qual são exibidas opções como pagamento com conta da plataforma, cartão de crédito, Pix e boleto. Nessa etapa, todo o tratamento dos dados sensíveis de pagamento ocorre diretamente nos provedores externos, mantendo a aplicação BROS responsável apenas pela orquestração do fluxo e pelo registro do status da transação.

Figura 7 – Interface de pagamento no ambiente do Mercado Pago.

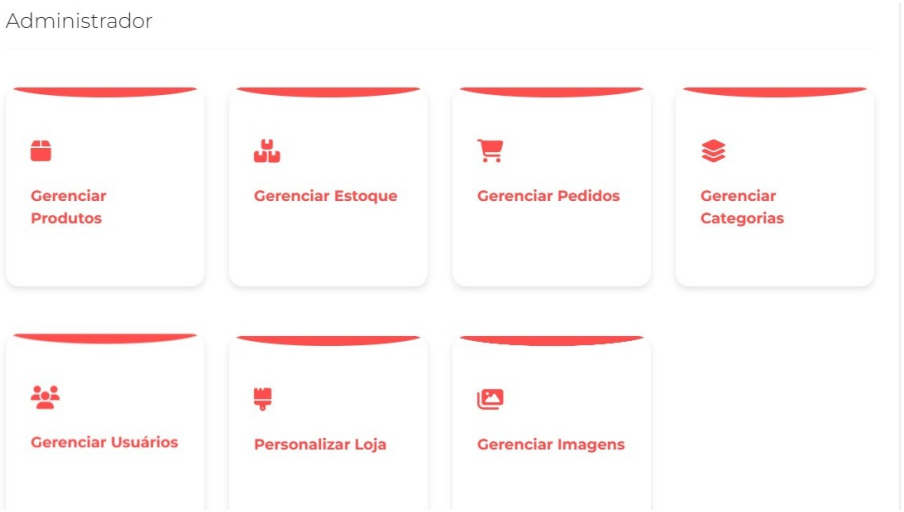


Fonte: Acervo dos autores

4.4 Painel Administrativo

O Painel Administrativo reúne todos os módulos responsáveis pelo gerenciamento interno da plataforma, permitindo ao administrador manter controle sobre produtos, estoque, imagens, categorias, usuários e personalização da loja. A Figura 8 apresenta a visão geral do painel.

Figura 8 – Painel Administrativo do sistema

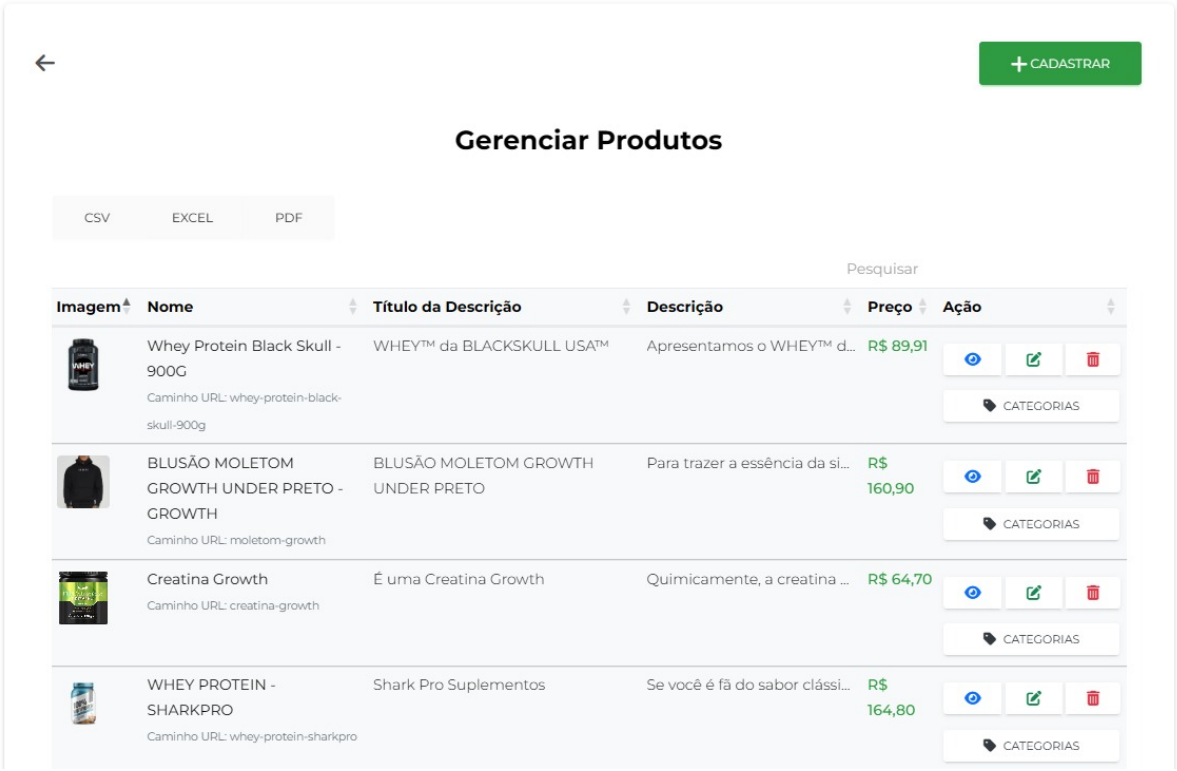


Fonte: Acervo dos autores

4.4.1 Gerenciar Produtos

O módulo “Gerenciar Produtos”, apresentado na Figura 9, centraliza a administração do catálogo de itens da loja para o locatário autenticado. A interface lista, em uma única grade, informações como imagem ilustrativa, nome do produto, título da descrição, preço e conjunto de ações disponíveis (visualizar, editar, excluir e gerenciar categorias vinculadas). A partir desse módulo é possível manter a coerência entre o que aparece na vitrine da loja e os registros gravados no banco de dados, evitando itens desatualizados ou inconsistentes. O sistema ainda oferece opções de exportação em CSV, Excel e PDF, o que facilita auditorias, elaboração de relatórios gerenciais e integração com controles externos de estoque ou financeiro.

Figura 9 – Tela do módulo Gerenciar Produtos

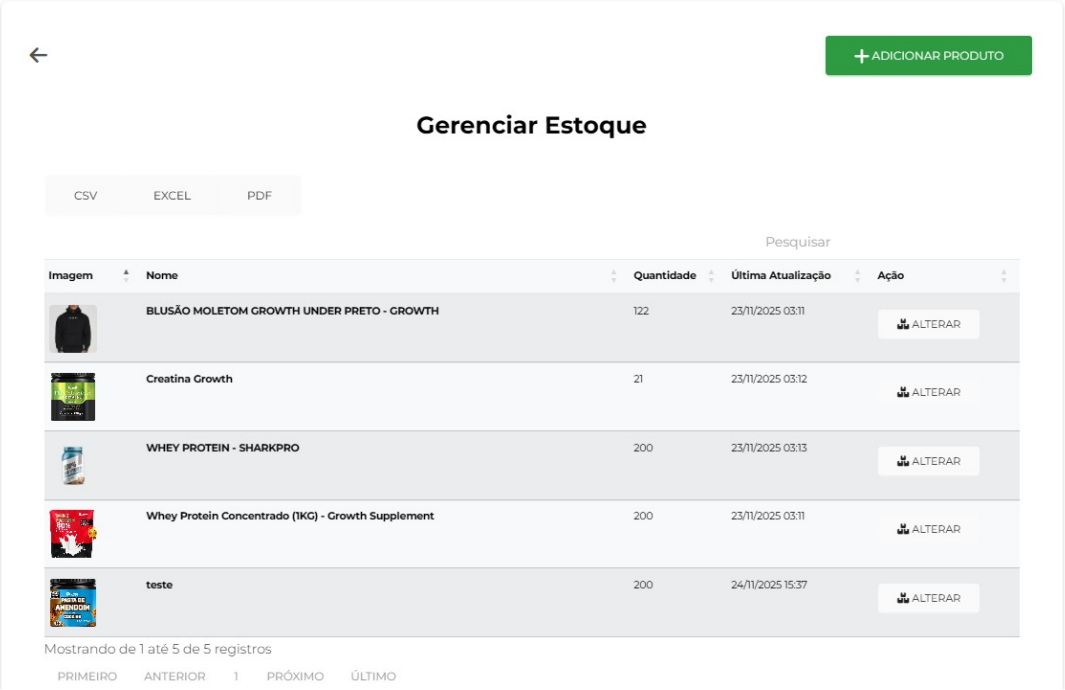


Fonte: Acervo dos autores

4.4.2 Gerenciar Estoque

O módulo “Gerenciar Estoque”, apresentado na Figura 10, exibe a relação de produtos com suas respectivas quantidades disponíveis e a data da última atualização registrada. A partir do botão “Alterar”, o administrador ajusta o saldo de cada item, registrando entradas ou saídas para alinhar o sistema à situação física do estoque. Esse módulo reduz o risco de vendas de produtos esgotados e de divergências operacionais, pois o controle passa a ser feito em um ponto único, diretamente integrado ao fluxo de pedidos do *e-commerce*.

Figura 10 – Tela do módulo Gerenciar Estoque

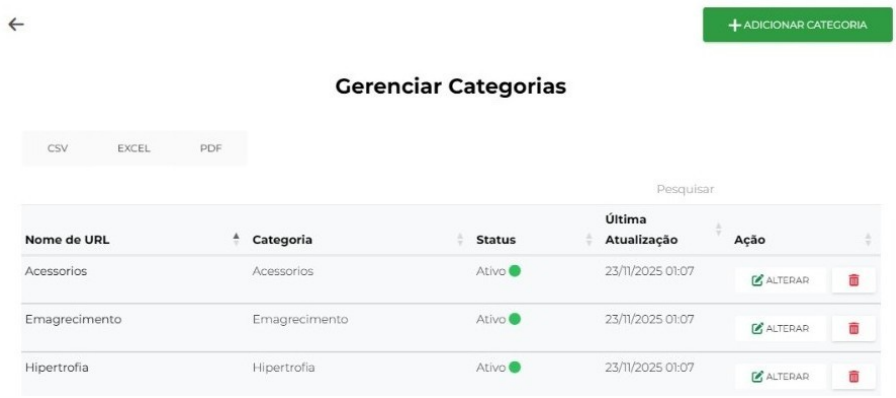


Fonte: Acervo dos autores

4.4.3 Gerenciar Categorias

O módulo “Gerenciar Categorias”, apresentado na Figura 11, organiza a estrutura hierárquica do catálogo de produtos. A tela exibe, para cada categoria, o nome amigável, o identificador utilizado na URL, o status de ativação, a data da última atualização e as ações de edição e exclusão. Como as categorias alimentam diretamente o menu principal e os filtros de navegação da loja, esse módulo é responsável por definir como o catálogo é apresentado ao usuário final, facilitando a localização de produtos e a criação de estruturas específicas para cada locatário.

Figura 11 – Tela do módulo Gerenciar Categorias

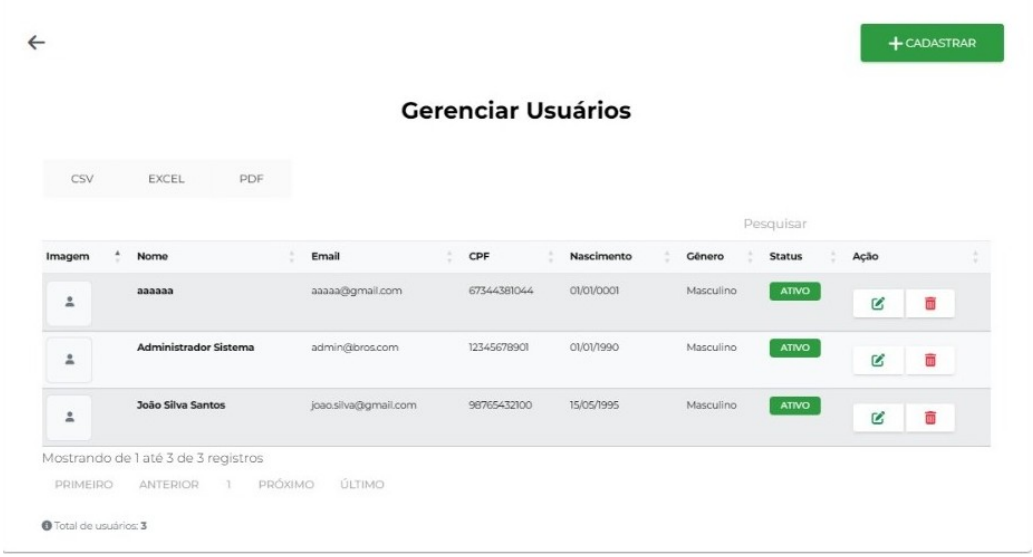


Fonte: Acervo dos autores

4.4.4 Gerenciar Usuários

O módulo “Gerenciar Usuários”, apresentado na Figura 12, permite ao administrador visualizar e administrar as contas cadastradas na plataforma. A listagem inclui dados como nome, e-mail, CPF, data de nascimento, gênero e status da conta, além de ações para cadastro, edição e exclusão de registros. O módulo também disponibiliza exportação em formatos como CSV, Excel e PDF, o que apoia atividades de auditoria, controle de acesso e integração com processos internos do lojista, garantindo maior governança sobre o ciclo de vida dos usuários do sistema.

Figura 12 – Tela do módulo Gerenciar Usuários

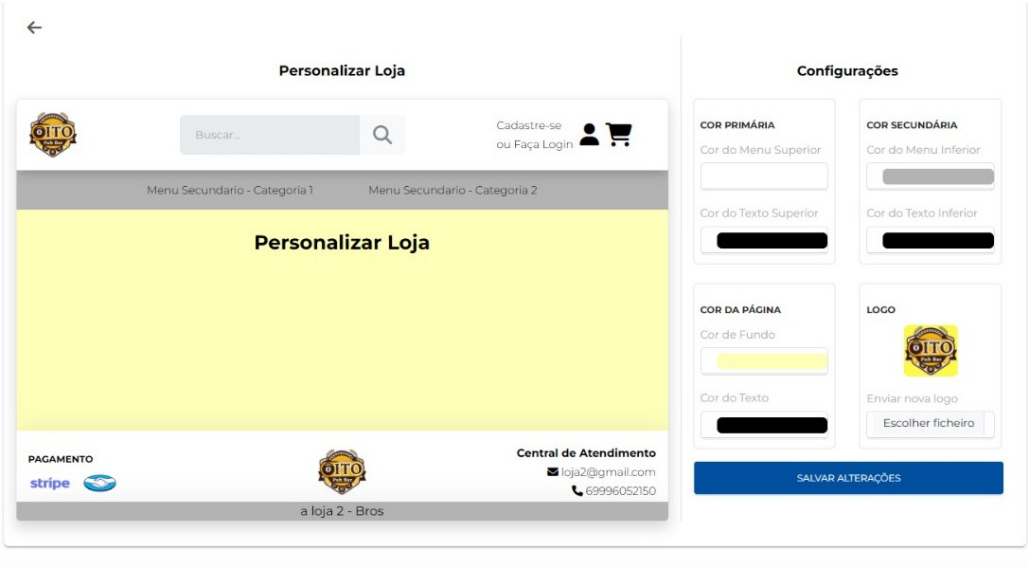


Fonte: Acervo dos autores

4.4.5 Personalizar Loja

O módulo “Personalizar Loja”, apresentado na Figura 13, permite que cada locatário ajuste a identidade visual da sua loja dentro da plataforma. A tela oferece campos para configurar cores de cabeçalho, rodapé e textos, além do envio de logotipo, exibindo uma prévia da página inicial com as alterações aplicadas. Dessa forma, o módulo viabiliza que várias lojas compartilhem a mesma infraestrutura técnica mantendo aparência e marca próprias, reforçando o caráter multiloja da solução sem necessidade de customizações de código.

Figura 13 – Tela do módulo Personalizar Loja

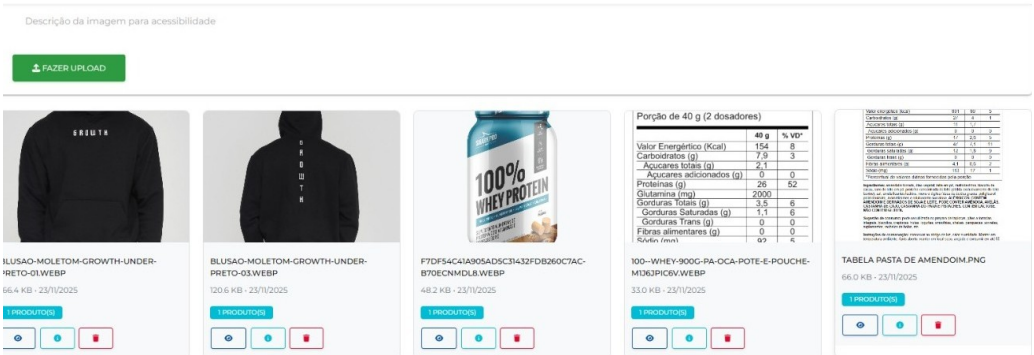


Fonte: Acervo dos autores

4.4.6 Gerenciar Imagens

O módulo “Gerenciar Imagens”, apresentado na Figura 14, reúne todas as mídias utilizadas na plataforma, permitindo o envio, organização e remoção de arquivos de forma centralizada. A interface suporta o envio de arquivos por arrastar e soltar, exibe a descrição alternativa associada a cada imagem e indica em quantos produtos aquele recurso visual está sendo utilizado, o que evita exclusões acidentais de imagens ainda referenciadas no catálogo.

Figura 14 – Tela do módulo Gerenciar Imagens



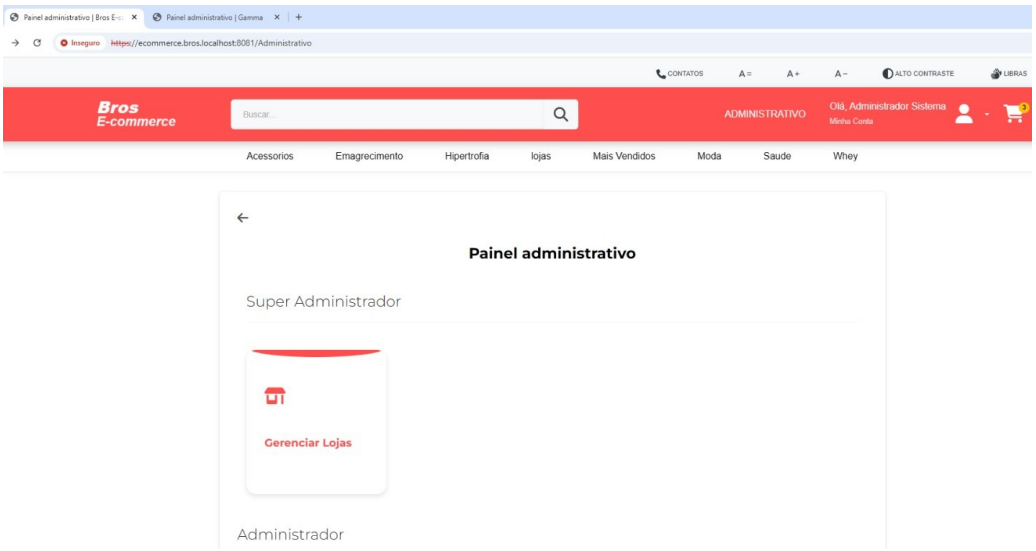
Fonte: Acervo dos autores

4.5 Gestão de locatários

A gestão de locatários (*tenants*) na plataforma BROS é responsabilidade exclusiva do perfil de superadministrador, que controla todas as lojas vinculadas ao ambi-

ente *multitenant*. No painel apresentado na Figura 15, esse perfil dispõe de um atalho dedicado para acessar a listagem de locatários e demais funções de administração global da solução.

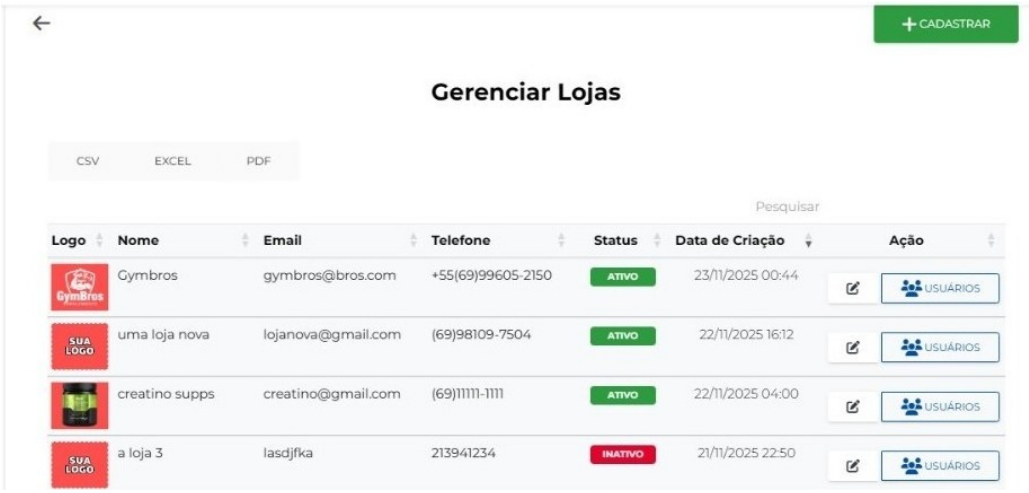
Figura 15 – Painel administrativo do superadministrador.



Fonte: Acervo dos autores

A tela ilustrada na Figura 16 exibe, em formato de tabela, informações como logotipo, nome da loja, e-mail, telefone, status (ativo ou inativo) e data de criação, além de ações para edição dos dados, controle de usuários vinculados e exportação dos registros.

Figura 16 – Tela de gestão de tenants e lojas cadastradas.

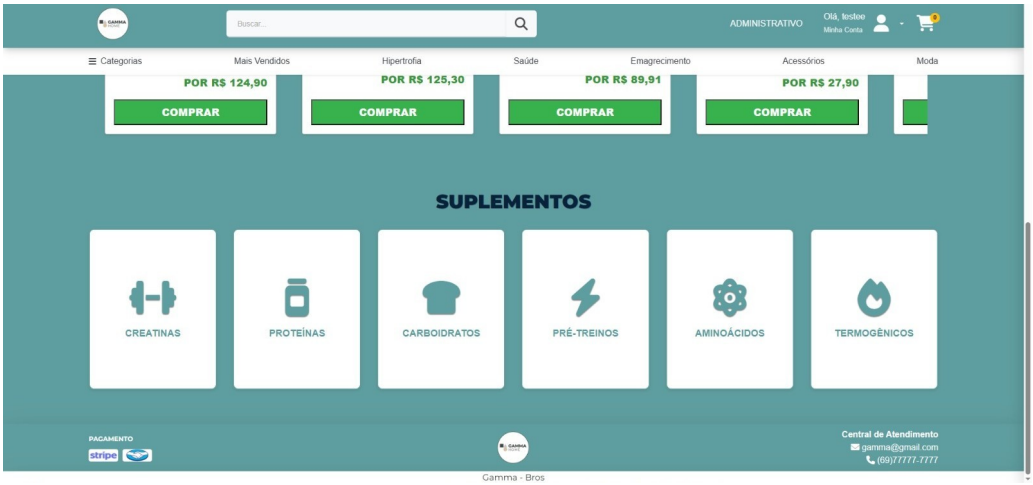


Fonte: Acervo dos autores

Ao concentrar as operações citadas anteriormente em um único módulo, o superadministrador consegue criar, ativar ou desativar lojas e acompanhar seu ciclo de

vida, materializando na prática o modelo multiloja e o isolamento lógico entre *tenants*.

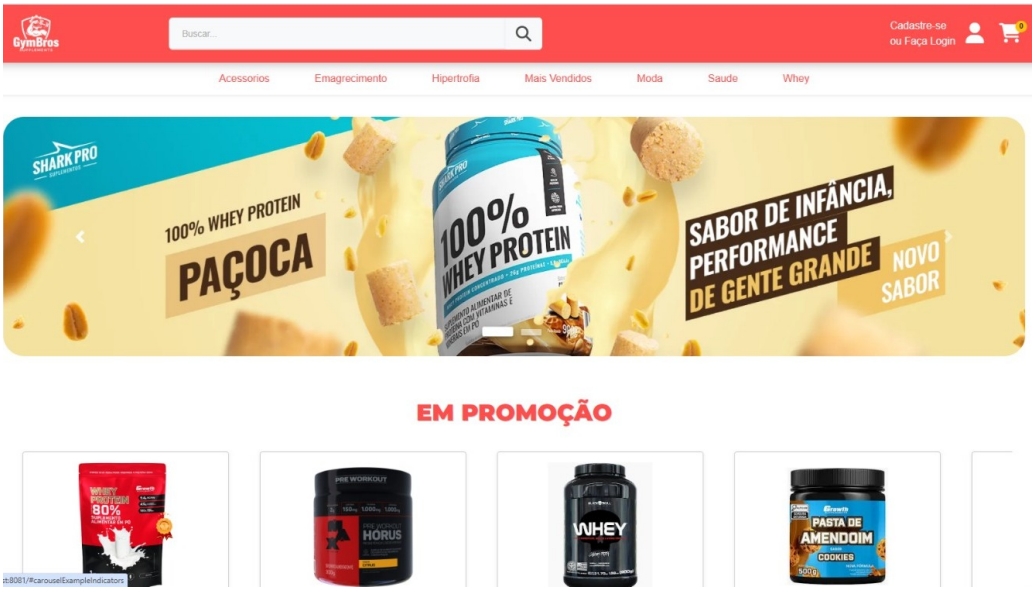
Figura 17 – Exemplo de tenant configurado com identidade visual Gamma.



Fonte: Acervo dos autores

Para evidenciar visualmente o funcionamento desse modelo, as Figuras 17 e 18 apresentam dois exemplos de lojas hospedadas na mesma instância da plataforma: uma configurada com identidade visual “Gamma” e outra com a identidade da antiga “GymBros”.

Figura 18 – Exemplo de tenant configurado com identidade visual GymBros.



Fonte: Acervo dos autores

Ambas as telas dispostas nas Figuras 17 e 18 compartilham a mesma base de código e infraestrutura, porém exibem cores, logotipo, cabeçalho e organização de categorias adaptadas às preferências de cada gestor. Essa personalização é viabilizada

pela combinação entre o módulo de gestão de *tenants* e o módulo “Personalizar Loja”, permitindo que diferentes empresas utilizem o mesmo *SaaS* com aparência e parâmetros de negócio próprios, sem necessidade de ramificações de código ou instalações isoladas.

5 CONCLUSÃO

O trabalho apresentou o desenvolvimento da plataforma BROS, um sistema de *e-commerce multitenant* voltado a pequenos e médios lojistas, estruturado sobre os princípios de *Domain-Driven Design* e *Clean Architecture*. A partir da evolução do protótipo acadêmico GymBros, foi possível conceber e implementar uma solução multiloja com autenticação, catálogo de produtos, carrinho de compras, fluxo completo de *checkout*, integração com *gateways* de pagamento e painel administrativo dividido entre perfis de administrador e superadministrador, com módulos para gerenciamento de lojas, produtos, categorias, usuários, estoque, imagens e personalização visual.

Metodologicamente, adotou-se um processo iterativo e incremental, apoiado em ferramentas como Figma e Jira, o que permitiu alinhar arquitetura, implementação e protótipos de interface ao longo dos ciclos. O resultado é um protótipo funcional que materializa os objetivos gerais e específicos: a arquitetura *multitenant* foi concretizada com isolamento lógico por *TenantId*, os fluxos de venda foram consolidados em um ambiente único e os pagamentos passaram a seguir práticas de segurança alinhadas ao PCI-DSS por meio da integração com provedores especializados.

Como continuidade, pretende-se desenvolver módulos de relatórios gerenciais e *dashboards* com indicadores de vendas, comportamento de clientes e desempenho por loja, aproveitando os dados já centralizados na plataforma. Também estão previstos o aprofundamento da integração com múltiplos *gateways* de pagamento, a incorporação de mecanismos de observabilidade (métricas, *logging* e monitoramento) e a realização de estudos de usabilidade com usuários finais e lojistas. Essa agenda de evolução busca conduzir o BROS da condição de prova de conceito acadêmica para uma solução *SaaS* madura, passível de adoção em cenários reais de comércio eletrônico multiloja.

REFERÊNCIAS

ABCOMM. **E-commerce brasileiro cresce 72% em 2020, aponta ABComm**. 2021. Disponível em: https://abcomm.org/dados-do-setor/ecommerce-cresce-72-em-2020/. Acesso em: 12 nov. 2025.

ALBERTIN, Alberto Luiz; MOURA, Rosa Maria de. Comércio eletrônico: seus aspectos de segurança e privacidade. **RAE — Revista de Administração de Empresas**, São Paulo, v. 38, n. 2, p. 49–61, 1998. Disponível em: https://rae.fgv.br/. Acesso em: 12 nov. 2025.

BEZEMER, Cor-Paul; ZAIDMAN, Andy. Multi-Tenant SaaS Applications: Maintenance Dream or Nightmare? *In:* 20–21 set. 2010. IWPSE-EVOL'10 — Proceedings of the 2010 International Workshop on Principles of Software Evolution and the 2010 European Workshop on Software Evolution. Antwerp, Bélgica: ACM, 2010. DOI: 10.1145/1862372.1862389. Acesso em: 12 nov. 2025.

CSS – W3C. **Cascading Style Sheets (CSS)**. 2025. Disponível em: https://www.w3.org/Style/CSS. Acesso em: 12 nov. 2025.

ECMA INTERNATIONAL. **ECMAScript Language Specification (JavaScript)**. 2025. Disponível em: https://262.ecma-international.org/. Acesso em: 12 nov. 2025.

EVANS, Eric. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. Boston: Addison-Wesley Professional, 2004.

MARTIN, Robert C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. Prentice Hall, 2019.

MICROSOFT. **.NET 9 SDK and ASP.NET Core MVC Documentation**. 2025. Disponível em: https://learn.microsoft.com/aspnet/core/. Acesso em: 12 nov. 2025.

MICROSOFT. **C# Language Specification**. 2025. Disponível em: https://learn.microsoft.com/dotnet/csharp/. Acesso em: 12 nov. 2025.

MICROSOFT. **SQL Server Documentation**. 2025. Disponível em: https://learn.microsoft.com/sql/. Acesso em: 12 nov. 2025.

MICROSOFT. **What is an API?** 2025. Disponível em: https://learn.microsoft.com/aspnet/core/fundamentals/apis. Acesso em: 12 nov. 2025.

MICROSOFT. **Entity Framework Core Documentation**. 2025. Disponível em: https://learn.microsoft.com/ef/core/. Acesso em: 12 nov. 2025.

PCI-SSC. **PCI DSS Quick Reference Guide**. 2025. Disponível em: https://www.pcisecuritystandards.org/standards/pci-dss. Acesso em: 12 nov. 2025.

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

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**. Scrum.org, 2020. Disponível em: https://scrumguides.org/.

STALLINGS, William. **Cryptography and Network Security: Principles and Practice**. 8. ed.: Pearson, 2017.

VLIBRAS. **Plugin de acessibilidade VLibras**. 2025. Disponível em: https://www.vlibras.gov.br/. Acesso em: 12 nov. 2025.

W3C. **HTML Standard**. 2025. Disponível em: https://html.spec.whatwg.org/. Acesso em: 12 nov. 2025.