UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CLEVERSON FULBER

E-MARKETPLACE PARA COMERCIALIZAÇÃO DE MATERIAIS ESPORTIVOS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2020

CLEVERSON FULBER

E-MARKETPLACE PARA COMERCIALIZAÇÃO DE MATERIAIS ESPORTIVOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Câmpus* Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Andreia Scariot Beulke

PATO BRANCO
2020

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
UTFPR - CAMPUS PATO BRANCO
DIRETORIA GERAL - CÂMPUS PATO BRANCO
DIR. DE GRAD. E EDUCAÇÃO PROFISSIONAL-PB
DEP. ACADÊMICO DE INFORMÁTICA DAINF-PB

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

E-Marketplace para Comercialização de Materiais Esportivos

POR

Cleverson Fulber

Este trabalho de conclusão de curso foi apresentado no dia 26 de novembro de 2020, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Profa. MSc Andreia Scariot Beulke	Prof. MSc Vinicius Pegorini	Profa. Dra. Rúbia Eliza de Oliveira Schultz Ascari
Professora orientadora	Professor convidado	Professora convidada

Assinam também:

Prof. Dr. Edilson Pontarolo	Profa. Dra. Mariza Miola Dosciatti
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas	Responsável pela Atividade de Trabalho de Conclusão de Curso

Documento assinado eletronicamente por (Document electronically signed by) **ANDREIA SCARIOT BEULKE, PROFESSOR(A) ORIENTADOR(A)**, em (at) 03/12/2020, às 19:58, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **RUBIA ELIZA DE OLIVEIRA SCHULTZ ASCARI, PROFESSOR DO MAGISTERIO SUPERIOR**, em (at) 03/12/2020, às 20:30, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **VINICIUS PEGORINI, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em (at) 04/12/2020, às 08:20, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **MARIZA MIOLA DOSCIATTI, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em (at) 04/12/2020, às 09:33, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **EDILSON PONTAROLO, COORDENADOR(A) DE CURSO/PROGRAMA**, em (at) 07/12/2020, às 10:28, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site (The authenticity of this document can be checked on the website) https://sei.utfpr.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador (informing the verification code) **1771546** e o código CRC (and the CRC code) **821272A7**.

RESUMO

O *e-marketplace* é um modelo de negócio que está se expandindo devido às vantagens que oferece para o consumidor, porque dispõe diversas marcas e modelos de mercadorias oferecidas por múltiplas lojas em um só ambiente e de forma *on-line*. Diversos sites vêm adotando essa estratégia, pois oferece maior comodidade e segurança para os clientes que adquirem um produto por meio de um site de sua confiança. Além disso, o *e-marketplace* beneficia também os pequenos negócios, pois estes têm a vantagem de anunciar em um grande site e tornar seus produtos conhecidos e, posteriormente, obter maior lucratividade do que teriam ao abrir um site próprio. Os sites de *e-marketplace* oferecem aos clientes produtos de todos os tipos, marcas e categorias, onde os clientes podem comparar ofertas de lojas diferentes e escolher a mais vantajosa. Diante do exposto, neste trabalho foi desenvolvido um sistema do tipo *e-marketplace* de materiais esportivos que visa atender clientes e lojistas desse segmento. O parceiro poderá ofertar seus produtos e os clientes poderão buscá-los acessando as páginas de produtos e também por filtros. Para o desenvolvimento do trabalho as principais tecnologias utilizadas foram Java, Thymeleaf e Spring.

Palavras-chave: E-marketplace. Materiais esportivos. Sistema web.

ABSTRACT

The e-marketplace is a business model that is expanding due to the advantages it offers to the consumer because it has several brands and models of goods offered by multiple stores in a single environment and online. Several websites have adopted this strategy, as it offers greater convenience and security for customers who purchase a product through a website they trust. In addition, the e-marketplace also benefits small businesses, as they have the advantage of advertising on a large website and making their products known and, later, obtain greater profitability than they would have when opening a website. The e-marketplace sites offer customers products of all types, brands, and categories, where customers can compare offers from different stores and choose the most advantageous. In view of the above, this work proposes an e-marketplace type system for sporting goods that aims to serve customers and retailers in this segment. The partner will be able to offer their products and customers will be able to search for them by accessing the product pages and also by filters. For the development of the work, the main technologies used were Java, Thymeleaf, and Spring.

Keywords: E-marketplace. Sports materials. Web system.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

AWS	*Amazon Web Services*
B2B	*Business to Business*
B2G	*Business to Government*
C2C	*Consumer to Consumer*
CEP	Código de Endereçamento Postal
CNPJ	Cadastro Nacional da Pessoa Jurídica
G2C	*Government to Citizen*
HTML	*Hipertext Markup Language*
JSP	*JavaServer Pages*
RF	Requisito Funcional
RNF	Requisito Não-Funcional
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa da realização deste trabalho. No final do capítulo é apresentada a estrutura do texto que apresenta a organização dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

O varejo eletrônico é uma área que se expandiu e ganhou visibilidade no comércio que envolve serviços como compra e venda de mercadorias. Isso porque possibilita que as empresas possam divulgar e comercializar seus produtos de forma mais abrangente, atingindo uma gama maior de clientes, já que as transações comerciais são realizadas por meio da Internet.

Comprar pela Internet é uma tarefa, atualmente, comum entre os usuários. De acordo com E-bit (2015), cerca de 51,5 milhões de consumidores fizeram compra em 2014, sendo que 10,2 milhões são novos usuários de *e-commerce*. Nessa perspectiva, Sasse e Braz Junior (2015) afirmam que ter uma loja *on-line* não é mais uma possibilidade apenas de grandes empresas, mas que pequenas e médias empresas vêm buscando espaço nesse segmento. Esses autores afirmam, ainda, que como essa atividade tornou-se comum, a demanda por criação de lojas virtuais aumentou potencialmente, sendo necessário encontrar novos canais para a comercialização de produtos.

Nesse contexto, de expansão da Internet e de oferta de produtos e serviços, surgem modelos de sistemas de comércio eletrônico para agregar a oferta dos varejistas. Entre os vários tipos de comércio existente estão os sistemas denominados *e-marketplace*. Esses sistemas funcionam como intermediador entre os compradores e os vendedores das lojas *e-commerce*, ou seja, permite que vários lojistas disponibilizem seus produtos de forma *on-line* a um único agregador, centralizando, assim, a oferta e a comercialização dos produtos inseridos no sistema.

Assim, um sistema *e-marketplace,* além de vender seus próprios produtos, permite, também, comercializar produtos de outras lojas, ou seja, esse tipo de sistema funciona como um *shopping* virtual, onde vários lojistas se reúnem para divulgar e comercializar seus produtos em um mesmo espaço (HONG; CHO, 2011).

Nas lojas de *marketplace* é possível encontrar todo tipo de produto. Contudo, há os que comercializam produtos de uma única categoria e são especialistas em determinado segmento. Gilbertoni (2014) destaca que a transação comercial dos produtos oferecidos nesses sistemas é processada pelo próprio operador do *e-marketplace*.

Diante disso, neste trabalho, foi proposto o desenvolvimento de um sistema no modelo de *e-marketplace* para comercialização de produtos esportivos. O sistema visa reunir ofertas de diferentes fornecedores, proporcionando, assim, um espaço para o cliente encontrar os produtos desejados.

1.2 OBJETIVOS

O objetivo geral está relacionado com o resultado principal da realização deste trabalho e os objetivos específicos complementam o geral em termos de funcionalidades do sistema.

1.2.1 Objetivo Geral

Desenvolver um sistema no modelo *e-marketplace* para divulgação e comercialização de materiais esportivos.

1.2.2 Objetivos Específicos

- Permitir que lojistas do segmento de materiais esportivos possam divulgar e comercializar seus produtos por meio de um único ambiente.
- Permitir que os clientes possam localizar os produtos desejados utilizando um ambiente *on-line*.
- Permitir que os anúncios sejam validados antes de serem anunciados no sistema.

1.3 JUSTIFICATIVA

E-marketplace são ferramentas que auxiliam lojistas a comercializarem seus produtos e serviços por meio da Internet. Esse conceito iniciou nos Estados Unidos em 1995, em site de leilões, mas evoluiu para sistemas de ofertas de diversas marcas e produtos.

O Sebrae (2017) afirma que os sistemas de *e-marketplace* diferem de um *e-commerce* porque oferecem vantagens para todas as pessoas envolvidas (cliente, lojista e administrador). A vantagem para o lojista está nas possibilidades de vendas oferecidas pelo sistema, pois aumentam a visibilidade dos produtos em menos tempo e atingem uma maior gama de possíveis consumidores. Os clientes podem encontrar produtos de diversas marcas e modelos em um único ambiente em menor tempo. Além disso, podem comparar os valores e fornecedores

proporcionando maior agilidade na escolha do produto. Para o administrador (ou operador) do sistema, esse modelo de negócio é uma estratégia para aumentar suas receitas por meio das comissões sobre os produtos vendidos.

Sasse e Braz Junior (2015) afirmam que o *marketplace* não direciona os clientes para sites dos lojistas, mas concentra toda sua experiência (localização, compra, pagamento) em um único ambiente, contudo, geralmente, a entrega do produto é de responsabilidade do lojista.

Nesse sentido, esse trabalho se justifica pela necessidade de disponibilizar um ambiente único para atender as demandas dos clientes de diversas lojas de materiais esportivos. Para o desenvolvimento do trabalho, as principais tecnologias utilizadas foram a linguagem Java, o banco de dados PostgreSQL, o *framework* Spring.

Salienta-se que para sistemas *web* é importante que as interfaces atendam aos padrões de desenvolvimento, atendendo a critérios de navegação, organização, interação e apresentação. Além disso, é necessário que o sistema seja responsivo, pois muitos usuários utilizam dispositivos móveis para o acesso a esses sistemas. Para isso, foi utilizado o *framework* Bootstrap que permite o desenvolvimento de aplicações responsivas.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos dos quais este é o primeiro e apresenta as considerações iniciais, objetivos e a justificativa do trabalho. O Capítulo 2 apresenta o referencial teórico. No Capítulo 3 são apresentados os materiais e o método utilizados para o desenvolvimento do trabalho. No Capítulo 4 está o resultado da realização do trabalho, que é a modelagem, desenvolvimento das interfaces e da implementação do sistema. Por fim, está a conclusão e as referências utilizadas no trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a fundamentação teórica deste trabalho, cujo conteúdo explana sobre o conceito, características e tipos de *e-commerce.*

2.1 E-COMMERCE

As lojas de *e-commerce* surgiram com o intuito de facilitar as ações de vendedores e clientes em suas transações, pois proporcionam maior agilidade em suas negociações, visto que tudo é realizado em tempo real. Para Laudon (2001) o comércio e os negócios, quando realizados eletronicamente, fornecem capacidades para comunicação, coordenação e aceleração do fluxo das transações de compras e vendas. Segundo Kotler e Keller (2006, p. 489) com a utilização de um *e-commerce* "[...] a empresa ou site realiza ou facilita a venda de produtos e serviços *on-line*".

> "Comércio eletrônico (ou *e-commerce*) refere-se ao uso da Internet e da Web para conduzir negócios. Mais formalmente, diz respeito às transações comerciais realizadas digitalmente entre organizações e indivíduos ou entre duas ou mais organizações. O conceito de transações realizadas digitalmente abarca todas as transações mediadas pela tecnologia digital. Na maioria dos casos, isso significa transações que ocorrem pela Internet e pela Web. Transações comerciais envolvem a saída de valores (por exemplo, dinheiro) das fronteiras individuais ou organizacionais em troca de produtos e serviços." (LAUDON, 2007, p. 271).

Assim, o *e-commerce* se caracteriza como uma loja virtual, na qual um vendedor disponibiliza suas mercadorias para que sejam adquiridas por seus consumidores, sem precisar sair de casa para comprar, pagar ou buscar a mercadoria adquirida. Turban (2004, p.3) conceitua *e-commerce* como um "processo de compra, venda e troca de produtos, serviços e informações por redes de computadores ou pela Internet".

Devido a essa facilidade e comodidade nos negócios, os consumidores e comerciantes tem cada vez mais procurado esse tipo de comércio. Segundo Cobra (2009), o comércio eletrônico trouxe impactos significativos no modo de vida das pessoas. Isso porque esse tipo de comércio permite que os usuários tenham, facilmente, acesso aos mais variados produtos e serviços de compra e venda. Além disso, atividades como o acesso a pesquisas de preços e

características de produtos e a realização de pagamento e de troca ou devolução, podem ser realizadas sem necessitar da presença física do consumidor na loja.

Para Turban (2004), os tipos de *e-commerce* podem ser classificados de acordo com a função que ele oferece ou, ainda, com a natureza de sua transação. Esses autores afirmam que os tipos mais conhecidos são: Empresas para Consumidores (B2C - *Business to Consumer*); Empresas para Empresas (B2B - *Business to Business*); Consumidores e Consumidores (C2C - *Consumer to Consumer*); Empresas e Governo (B2G - *Business to Government*) e; Governo e Cidadão (G2C - *Government to Citizen*).

Para Torres (2013) cada tipo de *e-commerce* possibilita que sejam implementados modelos de negócios, como, por exemplo, *marketing*, *e-marketplace*, leilões, corretora de informações, customização de produtos ou serviços, provedores de serviços on-line e compras em grupo. Um *e-marketplace* possibilita a junção de diversos vendedores e produtos em um único ambiente, garantindo, assim, maior variedade de produtos para seus consumidores.

2.2 E-MARKETPLACE

Um *e-markeplace*, assim como o *e-commerce*, é uma loja virtual na qual vende-se bens e serviços ou, ainda, é um modelo de negócios capaz de conectar diretamente cliente e fornecedor sem participação de intermediários (SEBRAE, 2017). Por meio desse sistema, fornecedores de um ou diversos segmentos se reúnem para disponibilizarem suas mercadorias em um único ambiente. Isso facilita a escolha dos produtos pelos consumidores, pois não há necessidade de dirigir-se a diversas lojas físicas ou virtuais, pois as mercadorias podem ser localizadas em um único sistema.

Nessa perspectiva, Morinishi e Guerrini (2011) afirmam que os sistemas de *e-marketplace* apresentam vantagens tanto para os consumidores quanto para os fornecedores. Para os fornecedores esses sistemas funcionam como um meio eficiente na publicidade, além de reduzir custos de operações comerciais e financeiras e para os consumidores as vantagens podem ser visualizadas pela redução de tempo na seleção de fornecedores e maior variedade de produtos e suas informações.

Corroborando com Morinishi e Guerrini (2011), o Sebrae (2017) afirma que por meio desses sistemas os consumidores encontram uma variedade de produtos, marcas e modelos em um único lugar permitindo, assim, o acesso a diferentes lojas e preços mais competitivos podendo pagar os produtos em uma única transação.

No entanto, para os lojistas, essa praticidade de comercialização e publicidade pode ser comprometida, pois não há uma identidade visual definida da loja que oferece os produtos, uma vez que em um sistema de *e-marketplace* os produtos são oferecidos por vários lojistas de um ou diversos segmentos.

Morinishi e Guerrini (2011), explicam que sistemas de *e-marketplace* que oferecem produtos de um ramo da cadeia produtiva são conhecidos como verticais e, quando não há restrição a uma atividade específica, ou seja, os produtos disponibilizados são de vários segmentos, como, por exemplo, roupas, calçados, acessórios, materiais esportivos, brinquedos, entre outros, esses sistemas são conhecidos como horizontais.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para o desenvolvimento deste trabalho. Os materiais elencam as ferramentas e as tecnologias utilizadas na modelagem, banco de dados, linguagem e *frameworks* para programação *front e back-end.* O método está relacionado a um modelo de processo que visa ordenar e estruturar o desenvolvimento do aplicativo.

3.1 MATERIAIS

O Quadro 1 apresenta a lista de ferramentas e tecnologias que foram utilizadas para o desenvolvimento deste trabalho.

Quadro 1 – Lista de ferramentas e tecnologias

Ferramenta / Tecnologia	Versão	Site	Finalidade
Amazon Web Services (AWS)	1.11.106	https://aws.amazon.com/pt/	Armazenamento em nuvem
Apache Maven	4.0.0	http://maven.apache.org/ref/3.0.3/maven-model/maven.html	Automação de compilação
Astah Community	7.0.0	http://astah.net/editions/community	Modelagem de casos de uso e digrama de classes
Bootstrap	4.0.0	http://getbootstrap.com/	Framework *front-end*
DataTables	1.10.12	https://frontbackend.com/maven/artifact/org.webjars/datatables/1.10.12	Framework *back-end*
IntelliJ IDEA	2020.2.3	https://www.jetbrains.com/pt-br/idea/	Plataforma de desenvolvimento
Java	1.8.0_221	http://www.oracle.com/	Linguagem de programação
JQuery	3.1.1-1	https://jquery.com/	Biblioteca JavaScript
Lombok	0.33-2020.2	https://plugins.jetbrains.com/plugin/6317-lombok	Biblioteca Java
MySQL Workbench	8.0.22	https://www.mysql.com/products/workbench/	Design de banco de dados Visual
PostgreSQL	11.5-1	https://www.enterprisedb.com/	Banco de dados
Spring Framework, Spring Data JPA e Spring Security	2.1.3	http://www.spring.io/	*Framework back-end*

Thymeleaf	3.0.4	https://www.thymeleaf.org/	Mecanismo de modelo Java / HTML5
Visual Paradigm	16.2	https://www.visual-paradigm.com/	Modelagem do banco de dados

Fonte: Autoria própria.

3.2 MÉTODO

O método utilizado neste trabalho foi o processo sequencial linear de Pressman e Maxim (2016) que define as etapas de comunicação, modelagem, construção e entrega. Neste trabalho são apresentadas as duas primeiras etapas.

A etapa de comunicação consiste no levantamento de requisitos que foram obtidos por meio da experiência do autor do trabalho na utilização de sistemas de *e-commerce* e *e-marketplace* (como o mercado livre, por exemplo) e, também, foram realizadas pesquisas sobre sistemas semelhantes ao desenvolvido nesse trabalho. Dentre esses sistemas, destacam-se: Mercado Livre; B2W, gestora das bandeiras Americanas, Submarino e ShopTime; Walmart; Via Varejo; NetShoes; Maganize Luiza e; Dafiti. Nesses sistemas foram consultadas as informações necessárias para cada produto para definir as tabelas do banco de dados, cadastros e as funcionalidades do sistema.

Na etapa de modelagem foi realizada a análise e o projeto e foram documentados os requisitos do sistema proposto. A modelagem dos requisitos foi realizada com o objetivo de fornecer suporte para a implementação subsequente, que será realizada em trabalho futuro. A modelagem está relacionada com a construção do diagrama de casos de uso que representa os atores do sistema e as funcionalidades direcionadas a cada ator, diagrama de atividades que exibem o fluxo de atividades dos processos e com a construção do modelo de entidade e relacionamento do banco de dados.

Na etapa da implementação foram desenvolvidas as interfaces do sistema e os códigos *back-end*. Os testes foram realizados informalmente à medida que os requisitos eram implementados.

4 RESULTADOS

Este capítulo apresenta o resultado deste trabalho que é o desenvolvimento de um sistema de *e-marketplace* para divulgação e comercialização de materiais esportivos.

4.1 ESCOPO DO SISTEMA

Os sistemas do tipo *e-marketplace* são utilizados para divulgar e comercializar produtos de uma área específica ou em conjunto. Assim, neste trabalho foi proposto o desenvolvimento de um sistema no modelo *e-marketplace* com produtos do segmento de materiais esportivos. Basicamente são identificados três atores para o sistema: administrador, lojista e cliente.

O cliente terá a opção de realizar buscas dos produtos por palavra-chave e por filtros, podendo ser por marcas, categorias e tipos. O cliente poderá acessar o sistema por meio de computador, *tablet* ou celular, pois o sistema é para a *web*. Os clientes serão responsáveis por manter seu cadastro com *login*, senha e as informações pessoais e poderão gerenciar o carrinho de compras, inserindo e excluindo itens e alterando as quantidades desejadas para compra. Para finalizar a compra o cliente deverá autenticar-se no sistema e selecionar um endereço de entrega informado no seu cadastro ou adicionar um novo endereço.

O lojista é o indivíduo responsável por cadastrar-se como pessoa jurídica do sistema e, marcar a opção informando que deseja ser um parceiro. Após o administrador validar o cadastro do parceiro, ele poderá cadastrar os produtos a serem comercializados por meio do sistema. Somente poderão cadastrar-se como parceiros, empresas, cujos campos a serem informados, são: Cadastro Nacional da Pessoa Jurídica (CNPJ), razão social, nome fantasia, telefones e *e-mail*. Os anúncios registrados serão validados pelo administrador. Isso para verificar a legalidade da empresa e dos anúncios registrados.

O administrador será a pessoa responsável pelo gerenciamento do sistema. Ele cadastrará as categorias, marcas e tipos dos produtos e validará os anúncios inseridos pelos lojistas.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta a lista dos requisitos funcionais e o Quadro 3 a lista dos requisitos não-funcionais do sistema, cujas siglas são, respectivamente, RF e RNF.

Os requisitos funcionais são todas as funcionalidades esperados do sistema.

Requisitos não funcionais, não interferem em funcionalidades básicas, pois estabelece como o sistema se comportará em determinadas situações que interferem no resultado final e se relacionam com a qualidade do sistema.

Quadro 2 – Requisitos Funcionais

Identificação	Nome	Descrição
RF 1	Manter usuário	Refere-se ao cadastro dos usuários do sistema, contendo os dados necessários para identificação e *login*. Basicamente, os usuários são: administrador, lojistas e cliente.
RF 2	Manter categorias	As categorias referem-se aos grupos de produtos. como: calçados, roupas, acessórios, por exemplo, e são mantidas pelo administrador do sistema.
RF 3	Manter tipos	Refere-se aos tipos de produtos, como: botas, jaquetas, camisetas, por exemplo, e são mantidos pelo administrador do sistema.
RF 4	Manter marcas	Refere-se às marcas dos produtos disponíveis no sistema e são mantidas pelo administrador do sistema.
RF 5	Manter anúncio	Corresponde a um conjunto de um ou mais produtos anunciados por uma empresa por um determinado período de tempo e é mantido pelo parceiro. O anúncio será disponibilizado após ser validado pelo administrador.
RF 6	Manter produtos	Cadastro de produtos que deve estar associado a uma categoria, marca e tipo. Além disso, deve ser informado dados de nome, quantidade, valor, tamanho, cor, descrição, características e imagem do produto. O produto é mantido pelo parceiro.
RF 7	Manter endereços de entrega	Ao finalizar a compra o cliente poderá selecionar um endereço já cadastrado ou incluir um novo endereço para entrega do pedido.
RF 8	Manter carrinho	O cliente poderá manter seu carrinho de compras por meio das operações de inclusão, exclusão, alteração e consulta dos itens do carrinho.
RF 9	Validar parceiro	O parceiro poderá cadastrar seus produtos somente se ele for validado pelo administrador do sistema.
RF 10	Validar anúncio	Os anúncios poderão ser visualizados somente após a validação pelo administrador do sistema. Um anúncio corresponde a um ou vários produtos cadastrados e que ficarão disponíveis por um determinado período de tempo.
RF 11	Finalizar compra	O cliente poderá concluir uma compra ao clicar na opção “Finalizar Compra”. O sistema exibe o resumo da compra e o endereço de entrega podendo ser alterado para outro já cadastrado ou um novo.

Fonte: Autoria própria.

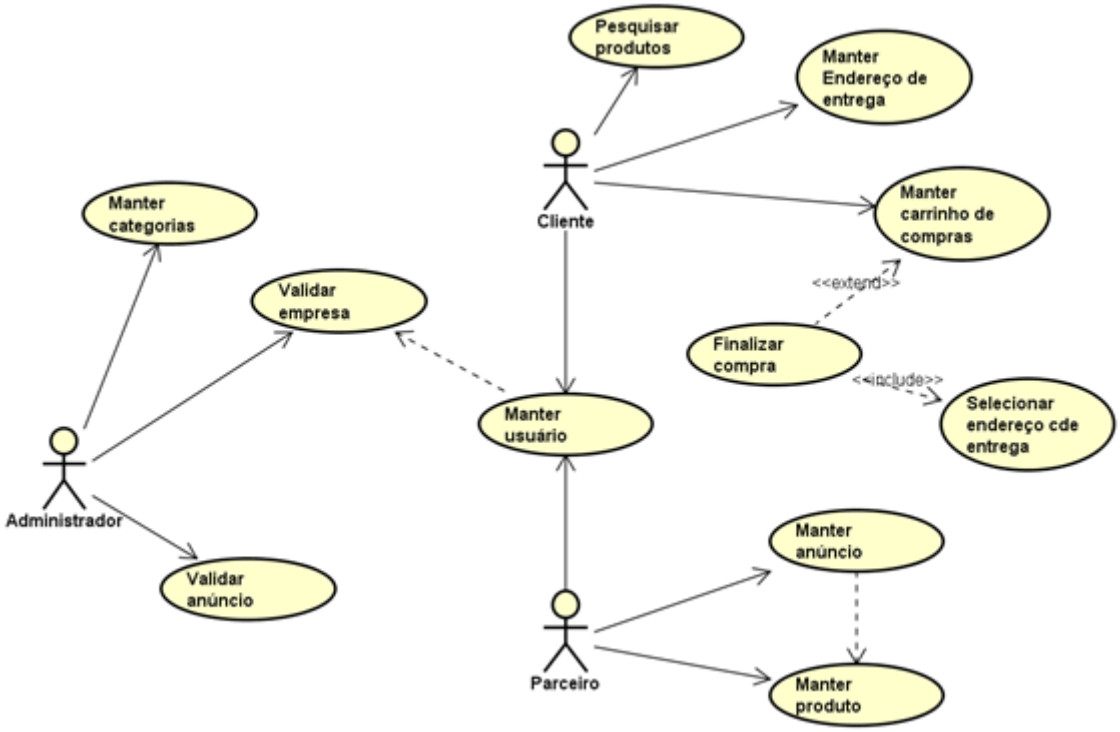
Quadro 3 – Requisitos não-funcionais

Identificação	Nome	Descrição
RNF01	Acesso ao sistema	O acesso ao sistema será realizado por meio de *login* e senha.
RNF02	Validação de campos para o cadastro de usuários	O campo *e-mail* deverá ser validado com restrições de formato. O campo senha deverá ser validado com restrições de quantidade mínima e máxima de caracteres.
RNF02	Campos de preenchimento obrigatórios	Os campos que são de preenchimento obrigatório serão validados por meio de uma função do sistema.
RNF03	Campos com máscaras de entrada	Os campos que possuem caracteres especiais e, que não serão armazenados no banco de dados, como, por exemplo, "()", ".", "-", para os campos RG, CPF, telefone, entre outros, serão validados por meio de máscaras de entrada.
RNF04	Finalizar compra	O cliente poderá finalizar o pedido somente se selecionar o endereço de entrega.

Fonte: Autoria própria.

A Figura 1 apresenta o diagrama de casos de uso do sistema. Esse diagrama representa as funcionalidades essenciais do sistema que são realizadas por seus atores: administrador, lojista e cliente. O administrador tem acesso às funções principais do sistema. O lojista é quem deseja anunciar seus produtos no sistema. Para isso, ele deverá realizar seu cadastro que será validado pelo administrador e, somente após a validação, o lojista poderá cadastrar os produtos e anúncios desejados. Um anúncio refere-se a um ou mais produtos que serão disponibilizados para venda no sistema atendendo a características, como, por exemplo, quantidade e tempo. O cliente é quem poderá consultar os produtos anunciados e realizar a compra *on-line.*

Figura 1 – Diagrama de Casos de uso



Fonte: Autoria própria.

Os Quadros numerados de 4 a 9 apresentam a expansão dos casos de uso. Os casos de uso dos quadros 4 a 7 apresentam a descrição dos casos de uso do tipo “Manter” e se referem às operações de inclusão, alteração, exclusão e consulta dos dados cadastrados no sistema. Essas operações apresentam, basicamente, o mesmo comportamento para todos os casos de uso que incluem essas atividades.

Quadro 4 – Operação incluir dos casos de uso “manter”

Caso de uso:
Incluir (refere-se à operação de inclusão nos casos de uso “manter”).

Descrição:
Ator inclui dados no sistema.

Atores:
Administrador, cliente ou lojista de acordo com suas funções definidas no caso de uso.

Pré-condição:
Estar autenticado no sistema, com exceção dos casos de uso cadastrar um usuário ou inserir um item no carrinho.

Sequência de Eventos:

1. Ator acessa a página inicial do sistema para realizar o cadastro.
2. Ator preenche os campos e clica no botão “Salvar”.
3. O sistema insere as informações no banco de dados e mostra mensagem de confirmação da operação.

Pós-Condição:
Registro inserido no banco de dados.

Nome do fluxo alternativo (extensão)	**Descrição**
1. Campos obrigatórios não preenchidos.	1.1. O ator não preenche os campos obrigatórios e clica no botão “Salvar”. 1.2 O sistema valida as informações e exibe mensagem informando que os campos obrigatórios não foram preenchidos corretamente. 1.3. O sistema retorna à tela de inclusão, com os campos que haviam sido preenchidos e destacando os campos obrigatórios sem preenchimento.
2. Campos preenchidos com formato inválido	2.1. O ator preenche os campos de forma incorreta e clica no botão “Salvar”. 2.2. O sistema valida as informações e exibe uma mensagem informando que os dados foram preenchidos de forma incorreta. 2.3. O sistema retorna à tela de inclusão, com os campos que já haviam sido preenchidos e destacando os campos com formato inválido.

Fonte: Autoria própria.

Quadro 5 – Operação alterar dos casos de uso “manter”

Caso de uso:
Alterar (refere-se à operação de alteração nos casos de uso “manter”).

Descrição:
Ator altera dados no sistema.

Atores:
Administrador, lojista ou cliente de acordo com suas funções definidas no caso de uso.

Pré-condição:
Dados cadastrados no sistema.

Sequência de Eventos:
1. O ator acessa a tela para visualização de dados já cadastrados.
2. O sistema apresenta o registro selecionado para a alteração.
3. Ator altera os dados do registro e clica em salvar.
4. O sistema valida as informações e salva no mesmo registro.

Pós-Condição:
Registro alterado no banco de dados.

Nome do fluxo alternativo (extensão)	**Descrição**
1. Campos obrigatórios não preenchidos.	1.1. O ator não preenche campos obrigatórios e clica no botão “Salvar”. 1.2. O sistema valida as informações e exibe uma mensagem informando que campos obrigatórios não foram preenchidos e não salva as alterações no banco de dados. 1.3. O sistema retorna a tela de alteração, com os campos que já haviam sido preenchidos e destacando os campos sem preenchimento.

2. Campos preenchidos com formato inválido.	2.1. O ator preenche os campos de forma incorreta e clica no botão "Salvar". 2.2. O sistema exibe mensagem informando que os dados não estão no formato e não salva o registro. 2.3. O sistema retorna à tela de inclusão, com os campos que já haviam sido preenchidos corretamente e destaca os campos com formato inválido.

Fonte: Autoria própria.

Quadro 6 – Operação excluir dos casos de uso "manter"

Caso de uso: Excluir (refere-se à operação de exclusão nos casos de uso "manter"). **Descrição:** Ator solicita a exclusão de dados no sistema. **Atores:** Administrador, lojista ou cliente de acordo com suas funções definidas no caso de uso. **Pré-condição:** Dados cadastrados no sistema. **Sequência de Eventos:** 1. O ator acessa a tela para visualização de dados já cadastrados. 2. O sistema apresenta o registro selecionado para a exclusão. 3. Ator clica em excluir registro. 4. O sistema exclui as informações do banco de dados e exibe as informações do *status* do procedimento. **Pós-Condição:** Registro excluído no banco de dados.	
Nome do fluxo alternativo (extensão)	**Descrição**
1. Exclusão de registro que possui vínculos no sistema.	1.1. Ator solicita a exclusão do registro que possui vínculos no sistema. 1.2 O sistema verifica se o registro possui vínculos e, em caso positivo exibe uma mensagem informando que o registro não pode ser excluído.

Fonte: Autoria própria.

Quadro 7 – Operação consultar dos casos de uso "manter"

Caso de uso: Consultar (refere-se à operação de consulta nos casos de uso "manter"). **Descrição:** Ator solicita a consulta de dados cadastrados no sistema. **Atores:** Cliente, administrador ou lojista de acordo com suas funções definidas no caso de uso. **Pré-condição:** Dados cadastrados no sistema. **Sequência de Eventos:** 1. Ator acessa a tela para visualização de dados já cadastrados.

2. Ator indica quais dados pretende consultar por meio de filtros. 3. O sistema exibe os dados da consulta ao usuário. **Pós-Condição:** Dados são exibidos aos usuários.

Fonte: Autoria própria.

O Quadro 8 apresenta o caso de uso "Finalizar Compra". Essa funcionalidade permite que o cliente possa concluir uma compra ao clicar na opção "Finalizar compra". O sistema exibe o resumo da compra e o endereço de entrega cadastrado no momento da inserção do usuário no sistema. Esse endereço poderá ser alterado para outro cadastrado ou um novo.

Quadro 8 – Caso de uso "Finalizar compra"

Caso de uso: Finalizar Compra **Descrição:** Cliente adiciona os produtos desejados no carrinho de compras e seleciona o endereço de entrega. **Atores:** Cliente. **Pré-condição:** Cliente deve estar autenticado no sistema e o(s) produto(s) deve(m) estar inserido(s) no carrinho de compras. **Sequência de Eventos:** 1. Cliente acessa o sistema. 2. O sistema exibe a quantidade de itens no carrinho de compras e a listagem de produtos. 3. Cliente clicar no botão "Continuar". 4. Sistema exibe tela de endereços. 4.1. Cliente seleciona um endereço para entrega ou inclui um novo. 4.1.1. Cliente seleciona um endereço cadastrado para entrega. 4.1.1.1. Sistema mostra mensagem que endereço foi selecionado. 4.1.2. Cliente clica na opção "Novo Endereço". 4.1.2.1. Cliente preenche os campos para inclusão de outro endereço de entrega e clica no botão "Salvar". 4.1.2.2. Sistema retorna para a tela de endereço. 5. Sistema exibe a tela de resumo 5.1. Cliente muda endereço de entrega ou finaliza a compra. 5.1.1. Cliente clica sobre "Editar". 5.1.1.1. Sistema exibe a tela de endereços. 5.1.2. Cliente clica sobre "Finalizar compra". 5.1.2.1. Sistema exibe mensagem informando que a compra foi realizada com sucesso e envia e-mail para o cliente com os dados da compra. **Pós-Condição**: Pedido concluído, compra efetuada.	
Nome do fluxo alternativo (extensão)	**Descrição**
1. Campos obrigatórios não preenchidos.	1.1 Sistema retorna para a tela de inclusão de endereço com os campos não preenchidos destacados e exibe

	mensagem informando que esses campos são de preenchimento obrigatório.
2. Campos preenchidos com formato inválido.	2.1 Sistema retorna para a tela de inclusão de endereço com os campos preenchidos de forma incorreta destacados e exibe mensagem informando que esses campos estão preenchidos de forma inadequada.

Fonte: Autoria própria.

O Quadro 9 apresenta o caso de uso "Validar parceiro" que permite que o administrador possa validar um usuário que tenha solicitado se tornar um parceiro, para que o mesmo possa vender seus produtos.

Quadro 9 – Caso de uso "Validar Parceiro"

Caso de uso:
Validar parceiro.
Descrição:
O administrador poderá validar os usuários que tenham solicitado se tornar um parceiro.
Atores:
Administrador.
Pré-condição:
Ator deve estar autenticado no sistema.
Sequência de Eventos:

1. Ator acessa o sistema para escolher a opção de acesso a área do administrador.
2. Ator seleciona no menu lateral a opção validar usuários.
3. Ator verifica usuários sem validação, na coluna validar.
4. Ator clica sobre o botão "Validar".
5. Ator fecha a mensagem de alerta ao clicar em "OK".
6. Ator verifica os dados do parceiro e clica sobre "Validar".

Pós-Condição:

- Deverá ser mostrado na coluna "Permissões" a descrição "ROLE_LOJISTA".

Nome do fluxo alternativo (extensão)	**Descrição**
1. Parceiro com dados inválidos	1.1 Ator verifica dados e nota inconsistência. 1.2 Ator clica sobre "Não aceitar". 1.3 O sistema fecha a janela *modal* (tela secundária) e não altera a permissão do usuário.

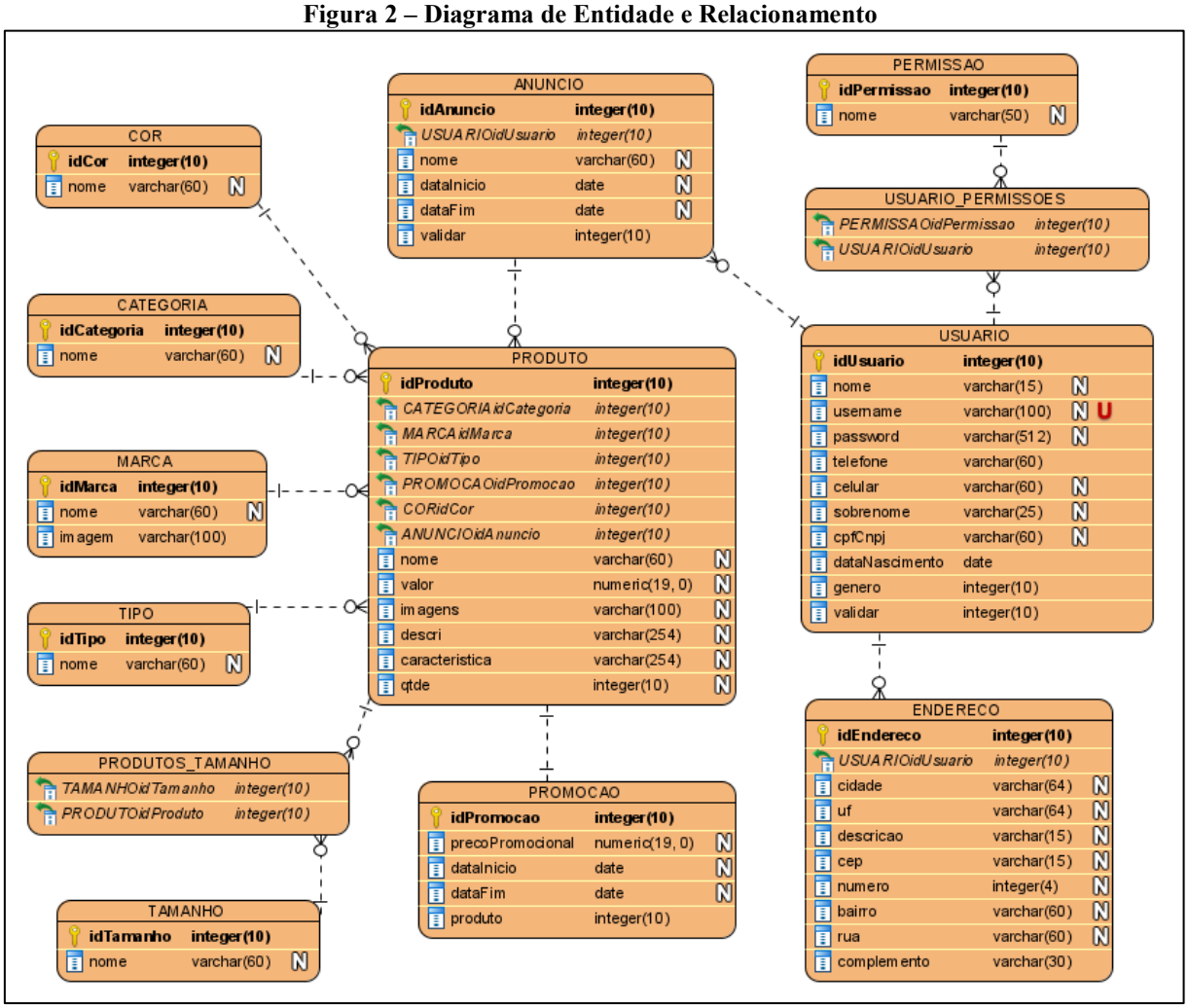
Fonte: Autoria própria.

A Figura 2 apresenta o diagrama de entidades e relacionamentos do banco dados do sistema. Na entidade denominada usuário, poderá ser definido três tipos de permissões do usuário. Os tipos possíveis para o sistema são: administrador, parceiros e cliente. Esses usuários serão carregados por meio de *enum* que será implementado no sistema. Parceiros e clientes

deverão fazer seus próprios cadastros, sendo que o primeiro deverá passar por validação do administrador. A entidade usuário ainda se relaciona com a endereço, em que todos os usuários podem ter endereços cadastrados, porém apenas o usuário parceiro pode inserir anúncios, produtos e promoções. Um anúncio pode ter vários produtos, o produto deve estar obrigatoriamente vinculado com um anúncio, o produto deve ter uma categoria, marca, tipo e cor, no mínimo um tamanho, e pode ser adicionado uma promoção.

Os endereços cadastrados possuem cidade e estado pegos automaticamente utilizando o serviço da *web* dos correios, que ainda pega o valor do frete que será calculado de acordo com o Código de Endereçamento Postal (CEP) definido pelo cliente.

Figura 2 – Diagrama de Entidade e Relacionamento



Fonte: Autoria própria.

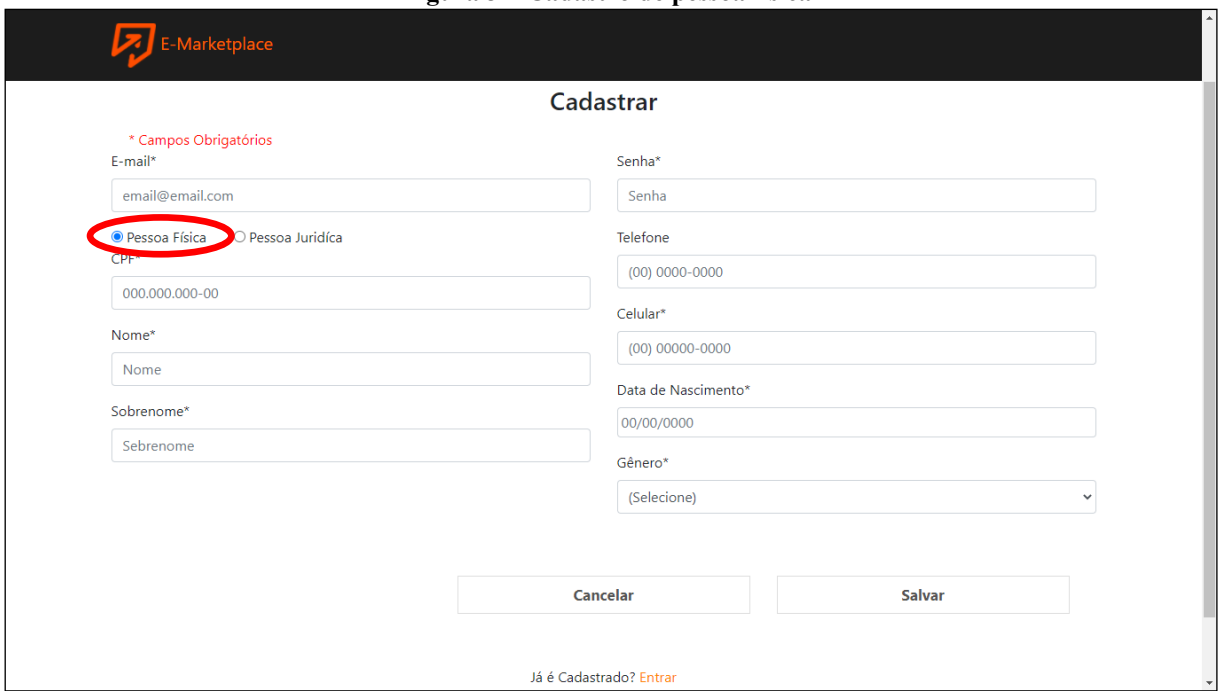
4.3 APRESENTAÇÃO DO SISTEMA

Essa seção apresenta as principais funcionalidades do sistema, por meio de figuras que exibem as principais telas implementadas de acordo com a análise elaborada para o desenvolvimento do sistema.

O sistema iniciará com a conta de administrador criada, a qual será repassada ao responsável pela loja, na qual ele poderá fazer as validações do sistema, já os demais usuários deverão criar suas próprias contas.

Na Figura 3 é apresentada a tela de cadastro de pessoa física. Após o usuário realizar o cadastro, ele terá permissões para visualizar os produtos, adicionar ao carrinho, manter seus endereços e fazer suas compras.

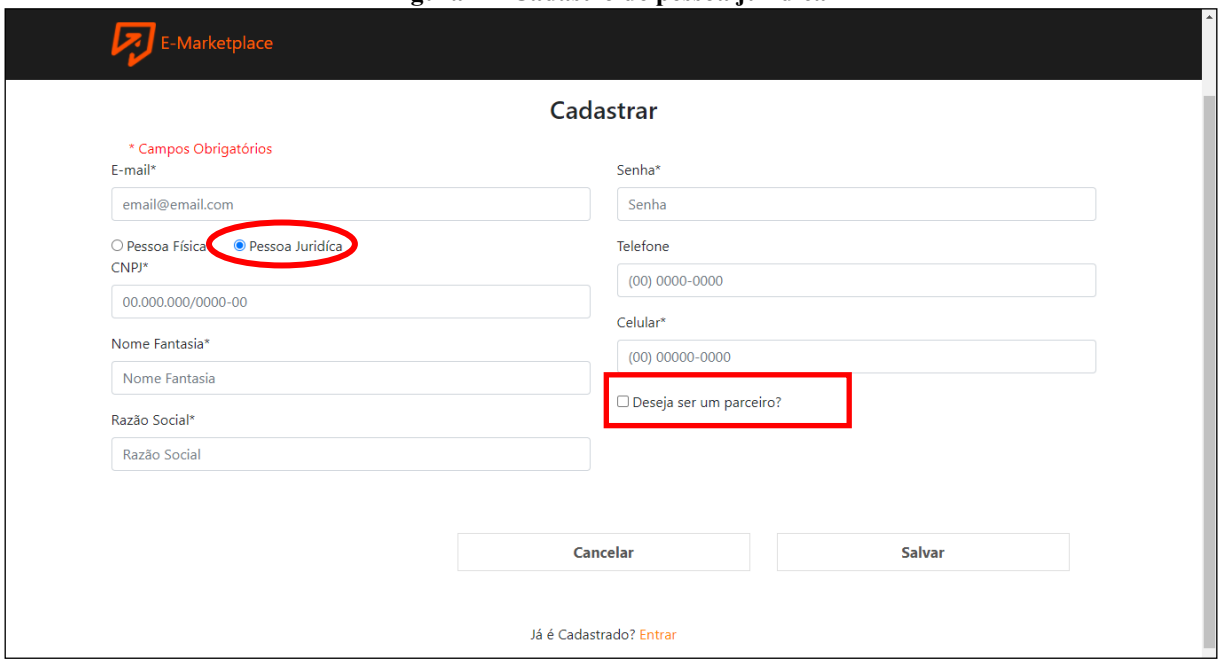
Figura 3 – Cadastro de pessoa física



Fonte: Autoria própria.

A Figura 4 apresenta a tela de cadastro para pessoa jurídica, na qual o usuário deve marcar a opção de “Pessoa Jurídica”. Quando o cadastro é finalizado ele terá as mesmas permissões que a pessoa física, porém se marcar a opção “Deseja ser um parceiro”, após seu cadastro ser validado pelo administrador do sistema, ele poderá adicionar seus produtos e vendê-los no site.

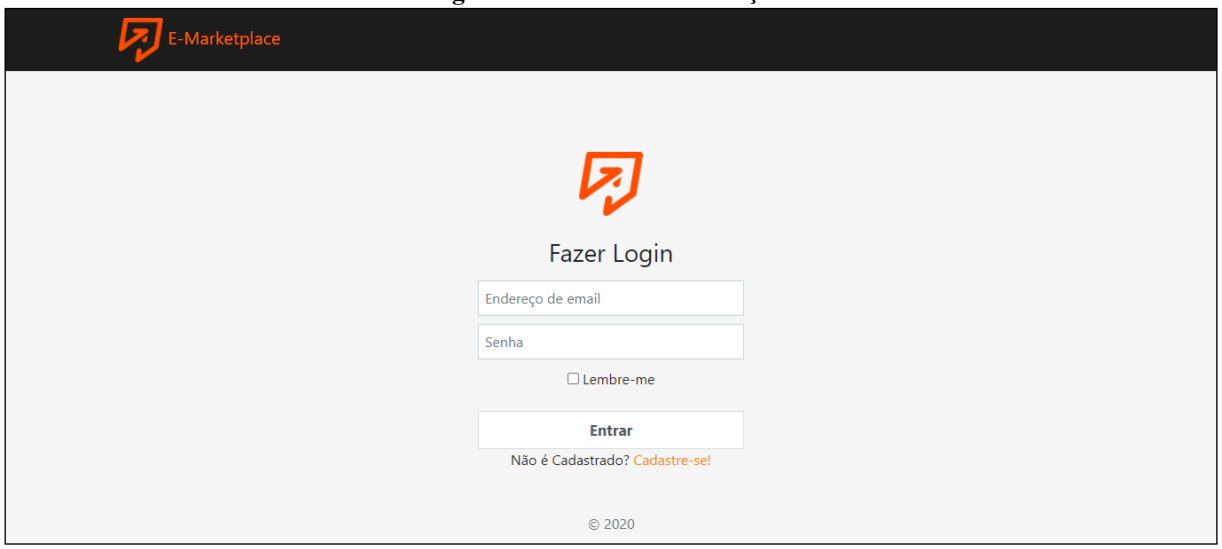
Figura 4 – Cadastro de pessoa jurídica



Fonte: Autoria própria.

A Figura 5, exibe a tela de autenticação no sistema para que os usuários possam acessar o sistema, informando os dados de e-mail e senha, e realizar suas operações.

Figura 5 – Tela de autenticação



Fonte: Autoria própria.

Após realizar a autenticação no sistema, o usuário será redirecionado para a página inicial que possui uma barra de navegação na parte superior da tela com informações, como: logomarca, campo de busca, menu para o usuário autenticado e o ícone do carrinho de compras.

Abaixo da barra de navegação, há a barra de menu que contém os menus de navegação e um menu para acesso às permissões do usuário autenticado.

Na parte destacada da Figura 6 está o menu para acesso às permissões de administrador.

Figura 6 – Tela inicial com permissões de Administrador

Fonte: Autoria própria.

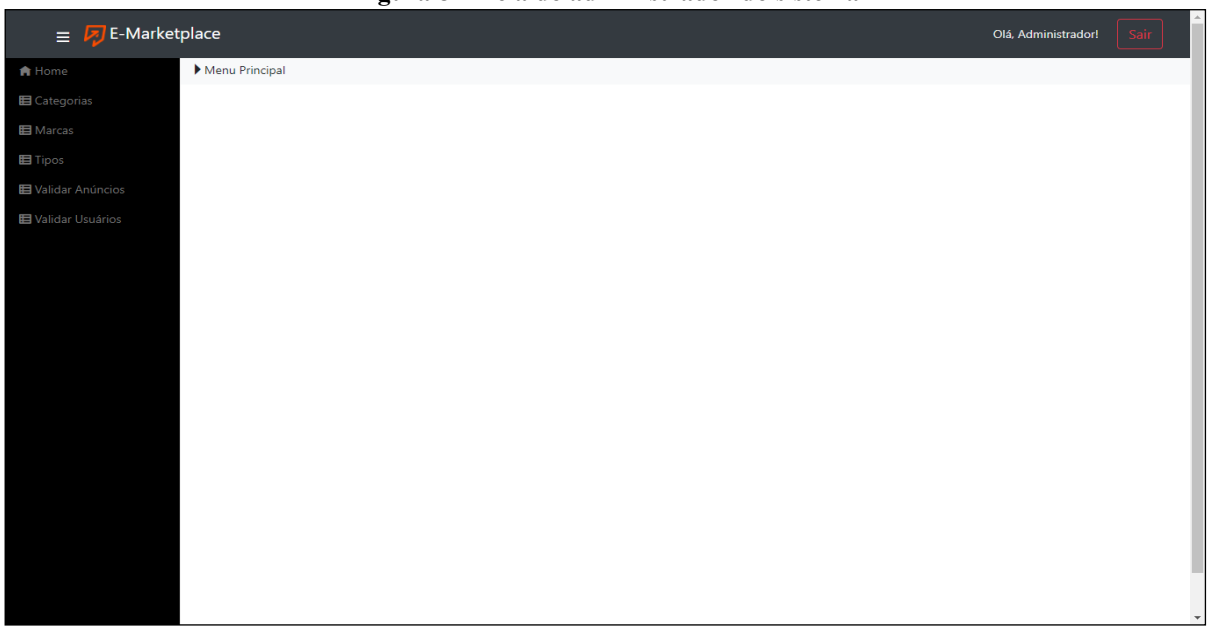
Na parte destacada da Figura 7 está o menu “E-Parceiros” que dá acesso ao parceiro às suas funcionalidades no sistema.

Figura 7 – Tela inicial com permissões de Parceiros

Fonte: Autoria própria.

Quando o administrador clicar o menu de acesso para acessar suas permissões será possível visualizar o menu lateral, apresentado à esquerda da Figura 8, e o administrador poderá manter categorias, marcas e tipos e, também, validar os parceiros que solicitaram acesso na página de cadastro, além de verificar os produtos e validar os anúncios postados pelos parceiros.

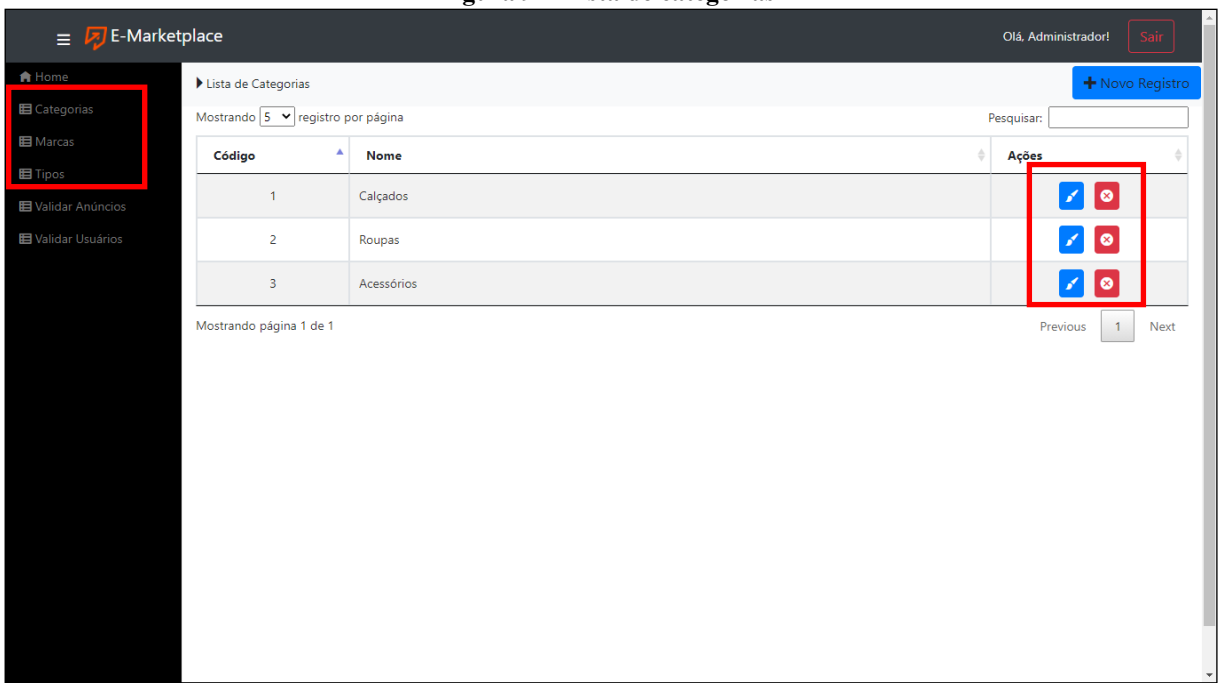
Figura 8 – Tela do administrador do sistema



Fonte: Autoria própria.

Ao escolher uma das três primeiras opções do menu (categorias, marcas e tipos), o administrador visualizará uma lista de itens cadastrados. Além disso é possível realizar operações como adicionar novos itens, editar e excluir. O botão “Novo Registro” permite fazer a inclusão de novos registros, as operações de editar ou excluir podem ser acessadas pelos botões apresentados na coluna “Ações” em cada linha da listagem dos produtos, conforme destacados na Figura 9.

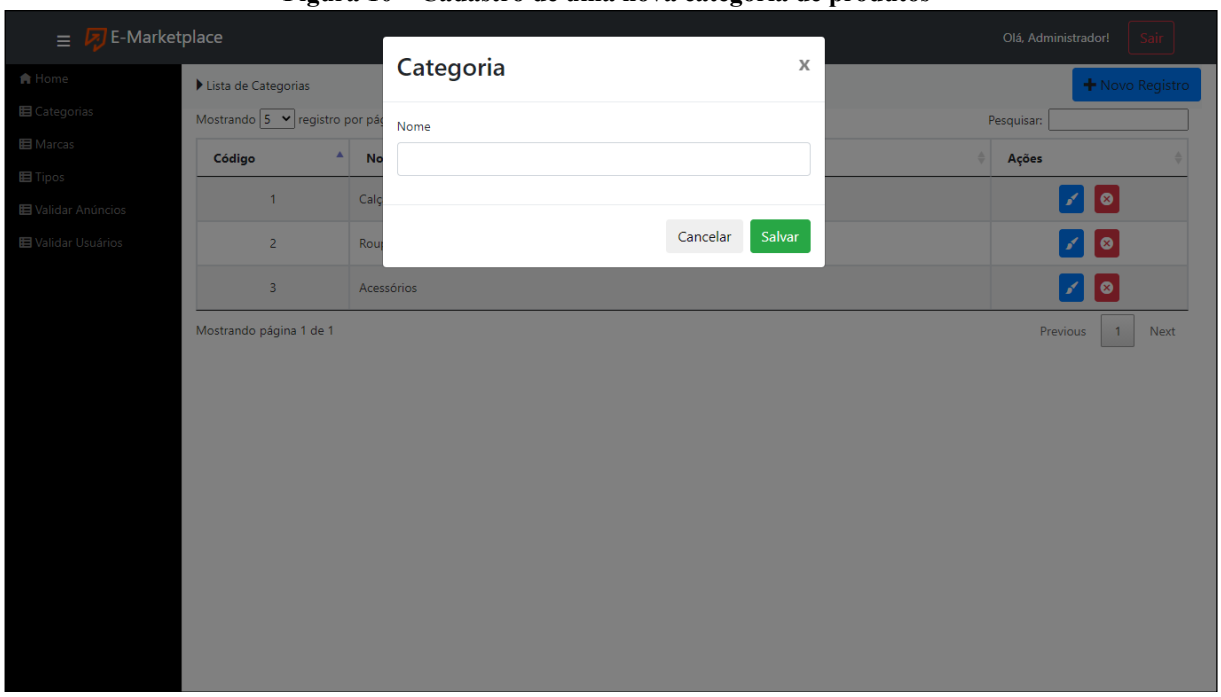
Figura 9 – Lista de categorias



Fonte: Autoria própria.

Para cadastrar uma nova categoria, o usuário deverá clicar no botão “Novo Registro” (apresentado na Figura 9) e uma janela *modal* será aberta, conforme apresentado na Figura 10 e, nela, o administrador digitará o nome da categoria e clicará no botão “Salvar” para que o registro seja gravado no banco de dados.

Figura 10 – Cadastro de uma nova categoria de produtos



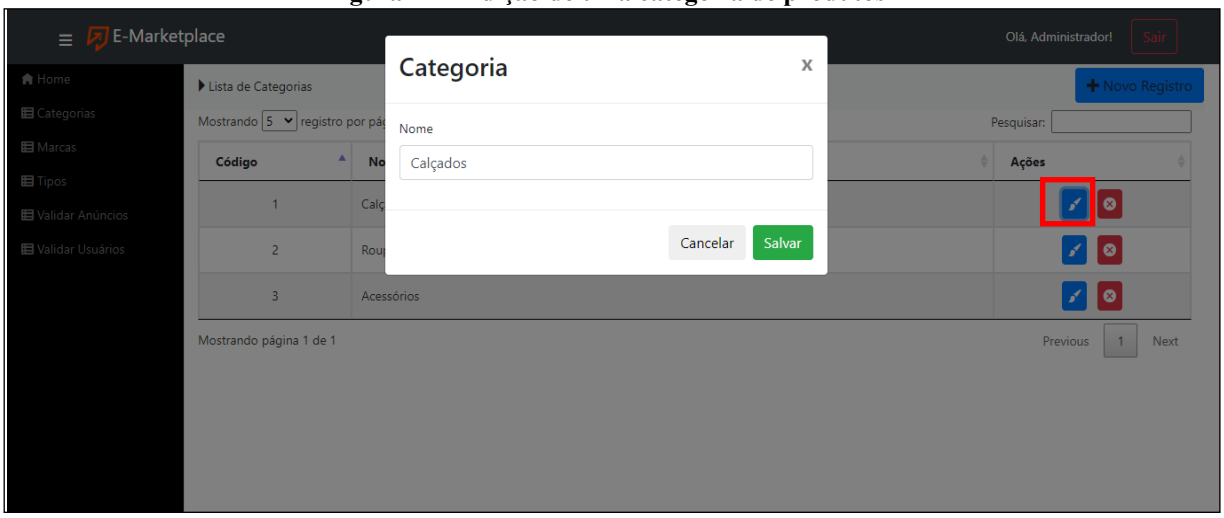
Fonte: Autoria própria.

As telas para cadastrar as marcas e os tipos dos produtos seguem o mesmo padrão apresentado no cadastro de categorias.

Ao clicar no botão editar de uma determinada categoria, será realizada uma consulta no banco de dados, e o registro será mostrado na tela em uma janela *modal*, conforme apresentado na Figura 11, na qual o usuário poderá alterá-lo ao clicar no botão “Salvar”. Caso clique no botão “Cancelar” ou sobre o botão fechar (ícone representado pelo x na parte superior da janela *modal*), a janela será fechada sem alterar o registro.

As telas para editar os registros de marcas e de tipos, seguem o mesmo padrão apresentado para alterar uma categoria.

Figura 11 – Edição de uma categoria de produtos

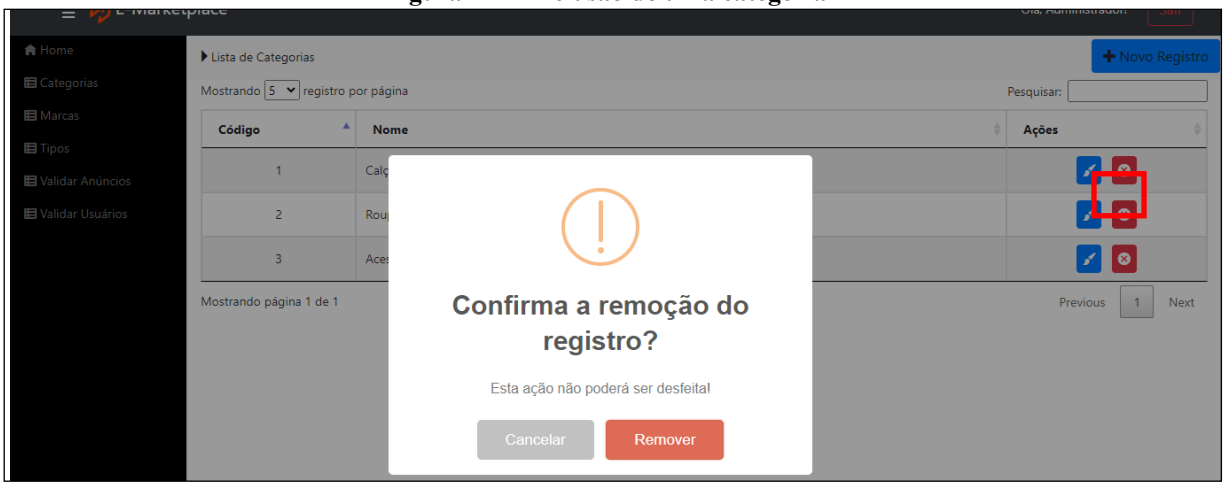


Fonte: Autoria própria.

Ao clicar no botão com a opção para excluir uma determinada categoria, será apresentada uma janela *modal* de confirmação, conforme exibido na Figura 13, ao clicar no botão "Remover" da janela *modal,* a categoria será excluída do banco de dados, caso escolha a opção "Cancelar" a *modal* será fechada sem excluir a categoria.

Todas as operações de exclusão do sistema seguem o mesmo padrão apresentado na Figura 12.

Figura 12 – Exclusão de uma categoria



Fonte: Autoria própria.

A tela apresentada na Figura 13 apresenta a página de validação de anúncios que exibe todos os anúncios disponíveis, os itens de cada anúncio e um botão para que o administrador possa validar o anúncio cadastrado pelo parceiro para que possam estar disponíveis para venda.

O acesso a essa tela pode ser feito pela opção do menu lateral esquerdo denominado “Validar” (conforme área destacada na Figura 13).

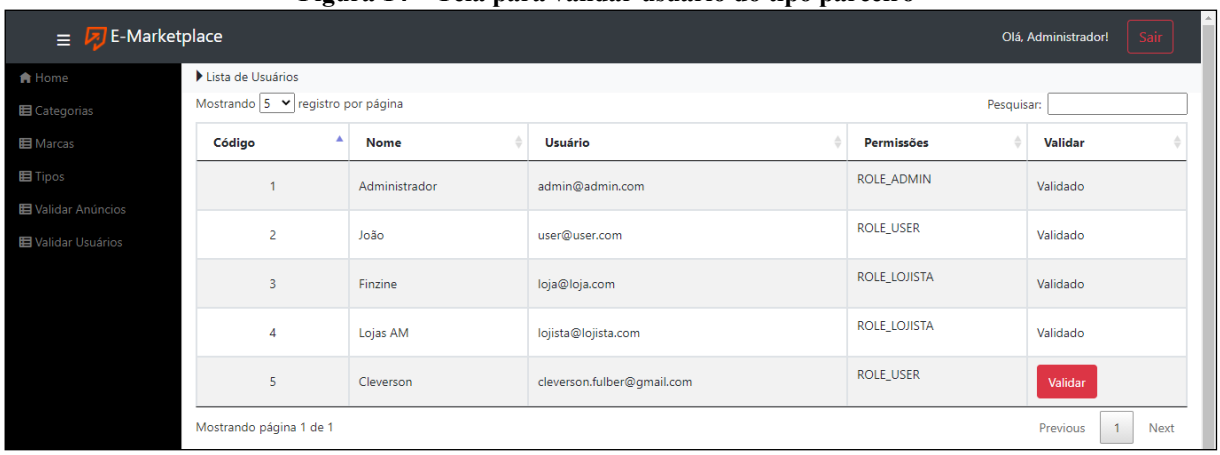
Os anúncios que não foram validados apresentarão um botão com o *label* “Validar” e ao clicar nesse botão aparecerá uma mensagem de confirmação para que o administrador possa confirmar a validação para que o anúncio com sua lista de produtos fique disponível na loja.

Figura 13 – Tela para validar anúncios

Fonte: Autoria própria.

O mesmo procedimento é realizado para validar um usuário do tipo parceiro, caso o usuário tenha efetuado seu cadastro como pessoa jurídica e tenha marcado a opção “Deseja ser um parceiro” na tela de cadastro apresentada na Figura 4. Na coluna “Validar” da lista de usuários apresentada na Figura 14, aparecerá uma opção para que o administrador possa validar o cadastro do parceiro.

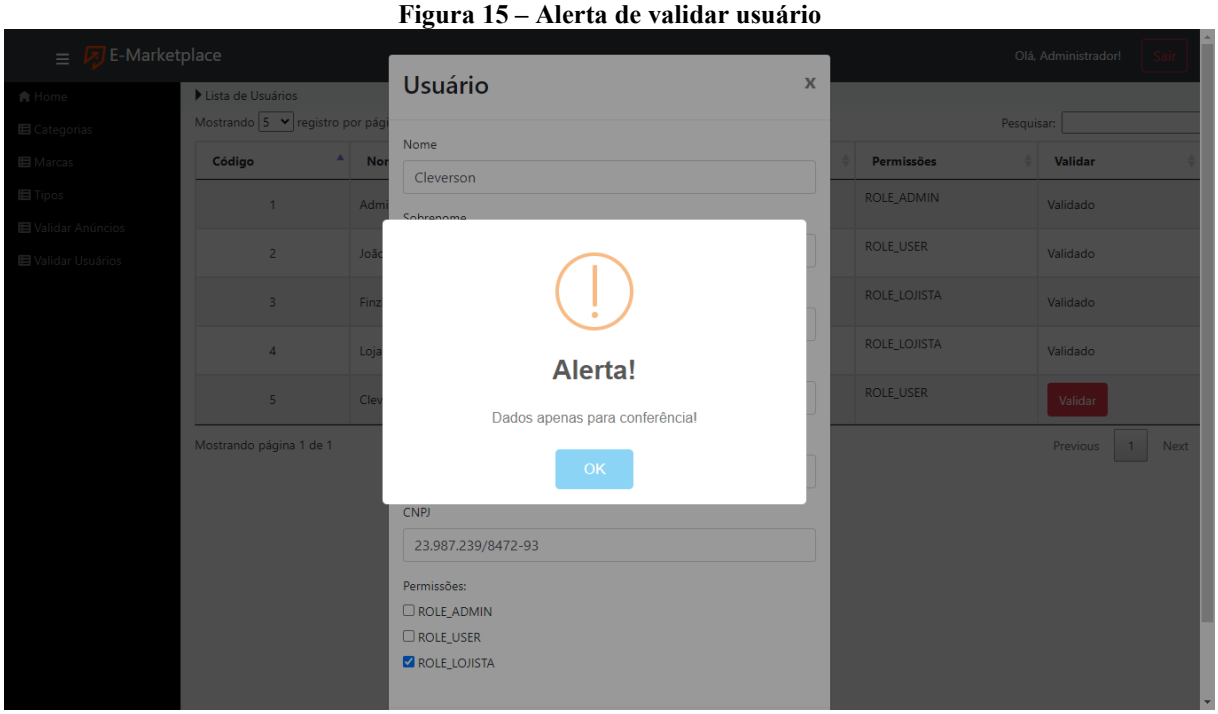
Figura 14 – Tela para validar usuário do tipo parceiro



Fonte: Autoria própria.

Ao clicar no botão “Validar” aparecerá uma mensagem informando que os dados são apenas para conferência, então alterações realizadas não terão qualquer ação no banco de dados. Ao clicar sobre “OK” os dados aparecerão para que o administrador possa conferi-los.

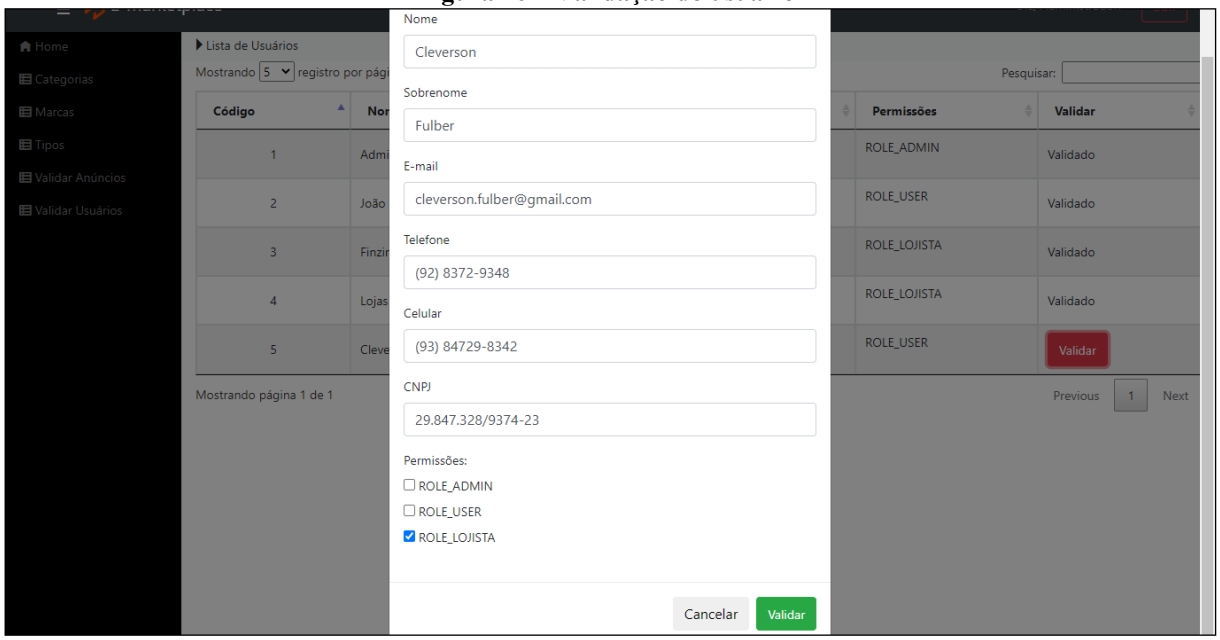
Figura 15 – Alerta de validar usuário



Fonte: Autoria própria.

Ao concluir a conferência, o administrador deverá clicar sobre a opção “Validar” (Figura 16) e então o usuário se tornará um lojista e poderá adicionar produtos no site após passar por validação.

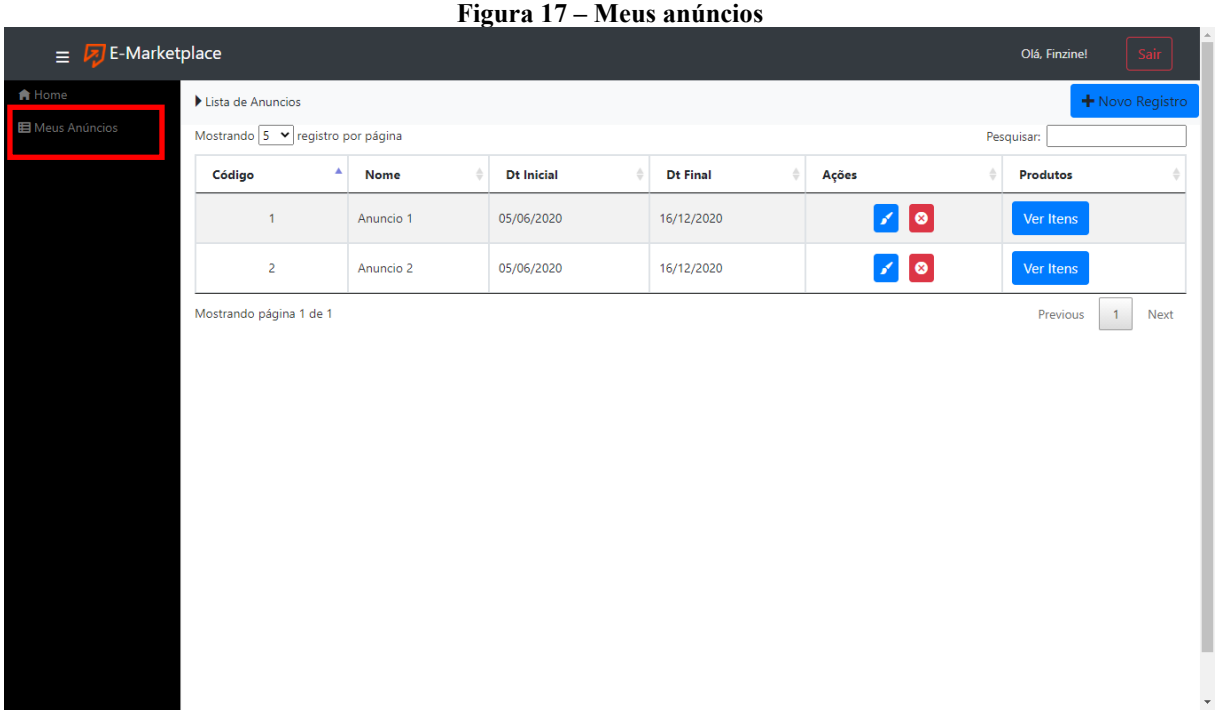
Figura 16 – Validação do usuário



Fonte: Autoria própria.

Quando o parceiro clicar sobre seu botão de acesso às suas permissões (apresentado na Figura 7), ele terá no menu lateral a opção “Meus Anúncios” conforme área destacada na Figura 17, na qual poderá realizar operação de consulta, inserção, edição e exclusão dos seus anúncios. Também poderá visualizar os produtos de cada anúncio clicando sobre o botão “Ver Itens” localizado na coluna denominada “Produtos” da Figura 17. Para inserir um novo anúncio, o parceiro deverá clicar no botão “Novo registro” e para alterar ou excluir os anúncios ele deverá clicar na opção desejada na coluna “Ações” da Figura 17.

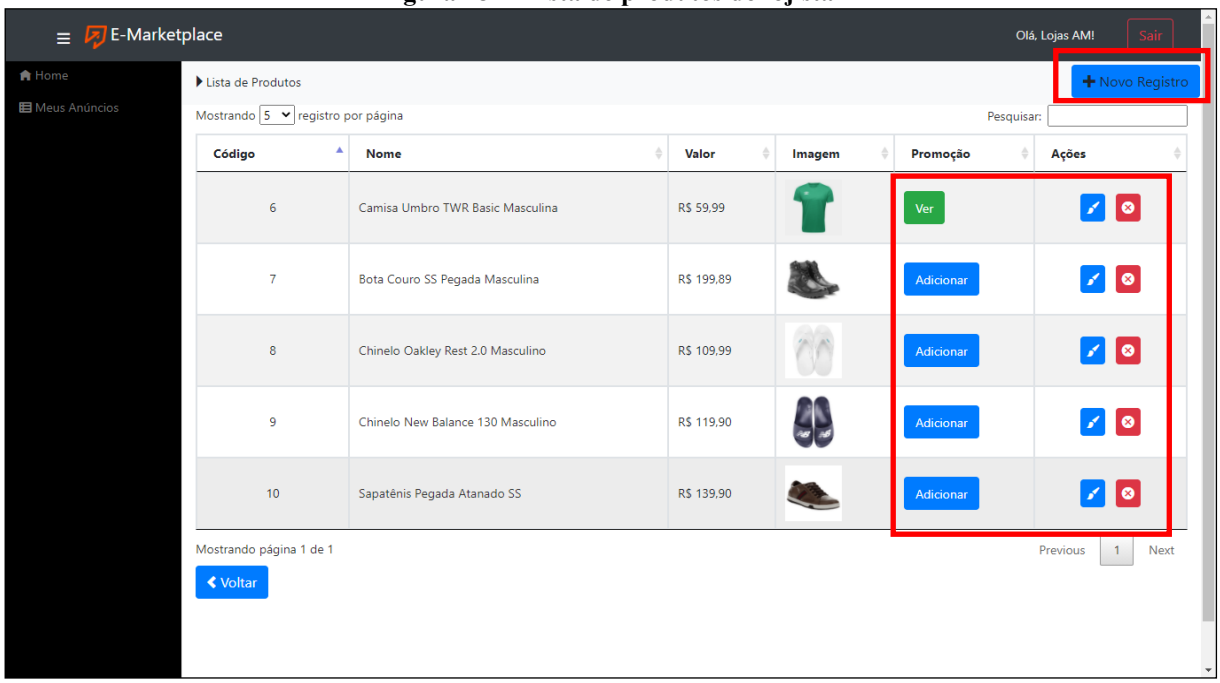
Figura 17 – Meus anúncios



Fonte: Autoria própria.

Quando o parceiro clicar no botão “Ver itens” apresentado na Figura 17, todos os produtos do anúncio selecionado serão listados (Figura 18), e o parceiro poderá realizar as operações para manter o produto no banco de dados, além de visualizar, adicionar ou excluir uma promoção conforme coluna “Promoção”, ressaltando que ao editar um produto, ele perderá a promoção.

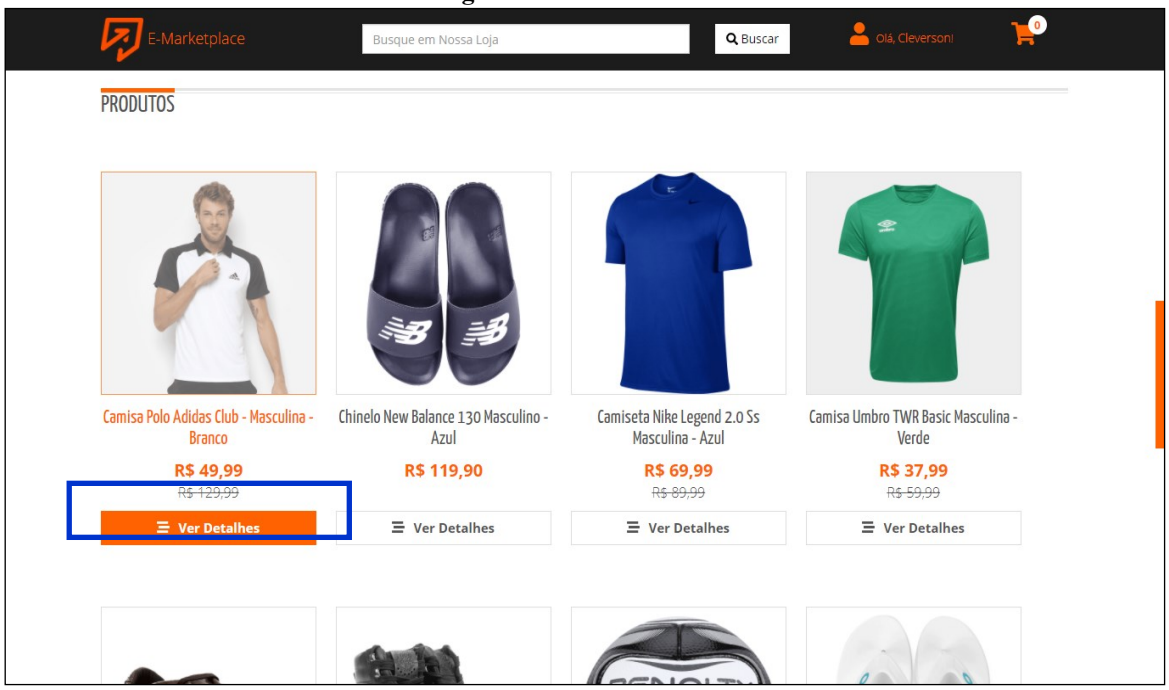
Figura 18 – Lista de produtos do lojista



Fonte: Autoria própria.

A Figura 19 exibe a tela com os produtos disponíveis para compras no site. Para isso, o usuário deve clicar sobre a opção “Ver detalhes” ou clicar no menu de produtos para que seja redirecionado para a página interna de produtos.

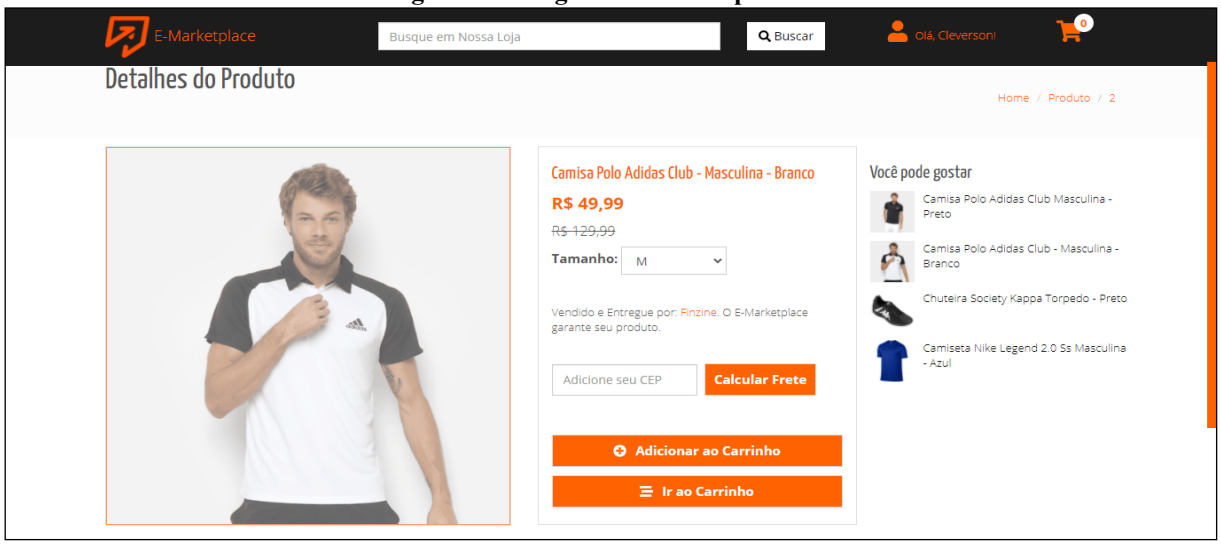
Figura 19 – Lista de Produtos



Fonte: Autoria própria.

A Figura 20 exibe a página interna do produto, na qual o usuário deverá escolher o tamanho do produto (caso seja roupa ou calçado) e, em seguida clicar no botão “Adicionar ao Carrinho”, para que o produto será adicionado.

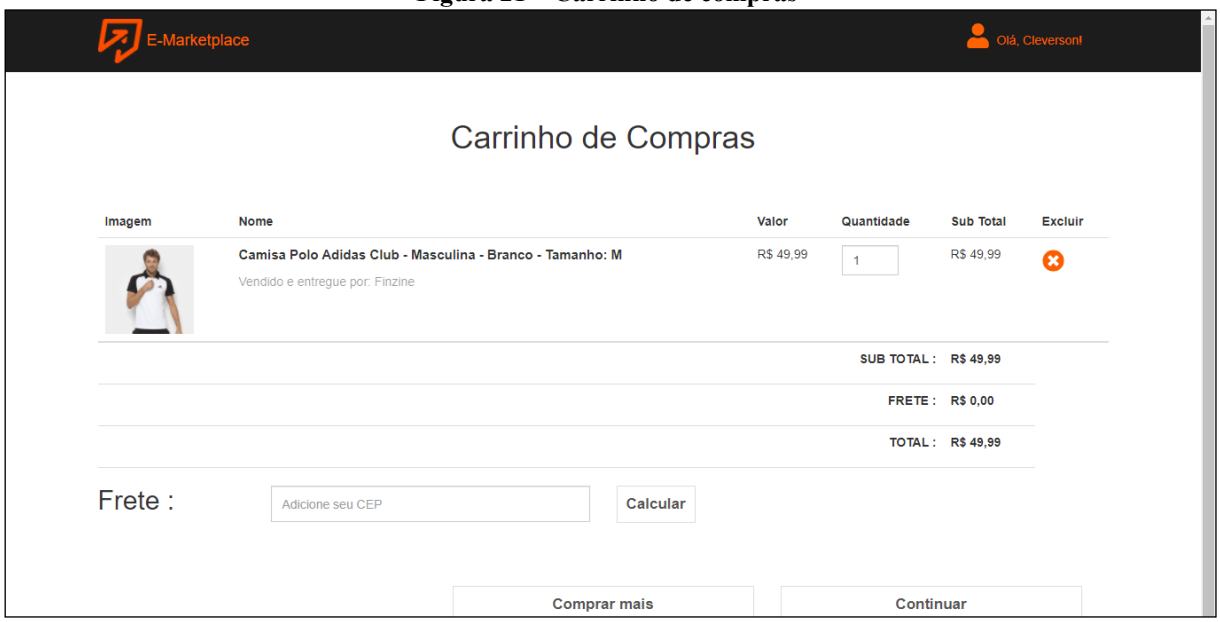
Figura 20 – Pagina interna do produto



Fonte: Autoria própria.

Na tela do carrinho de compras, apresentada na Figura 20, o usúario poderá alterar a quantidade de produtos que deseja comprar e, clicar sobre o botão “Continuar” para ser redirecionado para a tela de autenticação (apresentada na Figura 5) e, em seguida escolher o endereço cadastrado ou cadastrar novo endereço para entrega. Caso o usuário queira continuar comprando, após adicionar um determinado produto no carrinho de compras, poderá clicar no botão “Comprar mais”, apresentado na Figura 21.

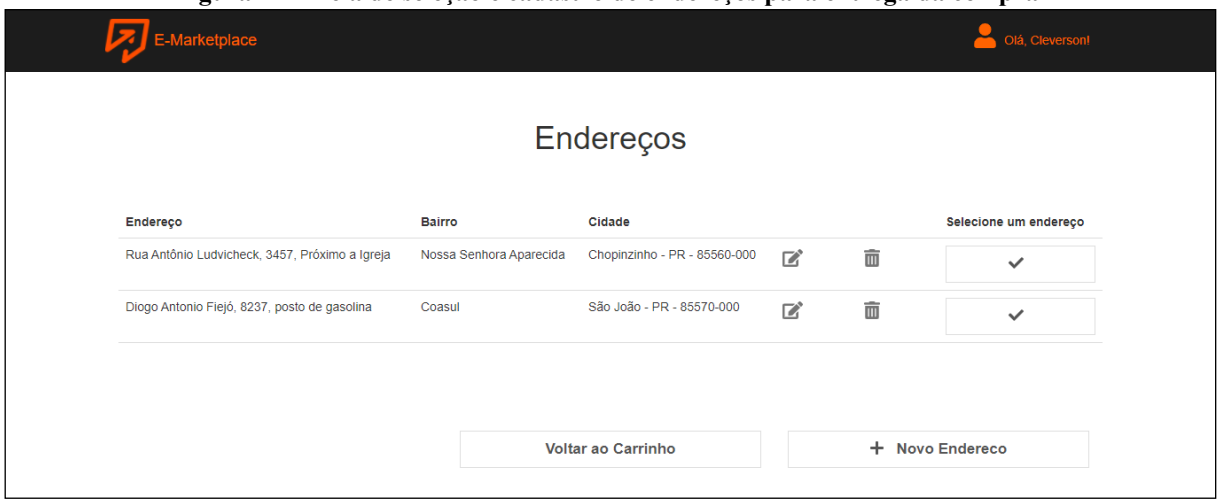
Figura 21 – Carrinho de compras



Fonte: Autoria própria.

Na página de endereços o usúario poderá adicionar endereços, alterar ou excluir os endereços já existentes, e para concluir a compra o usúario deve selecionar o endereço desejado para entrega clicando no botão, localizado na coluna “Selecione um endereço” da Figura 22.

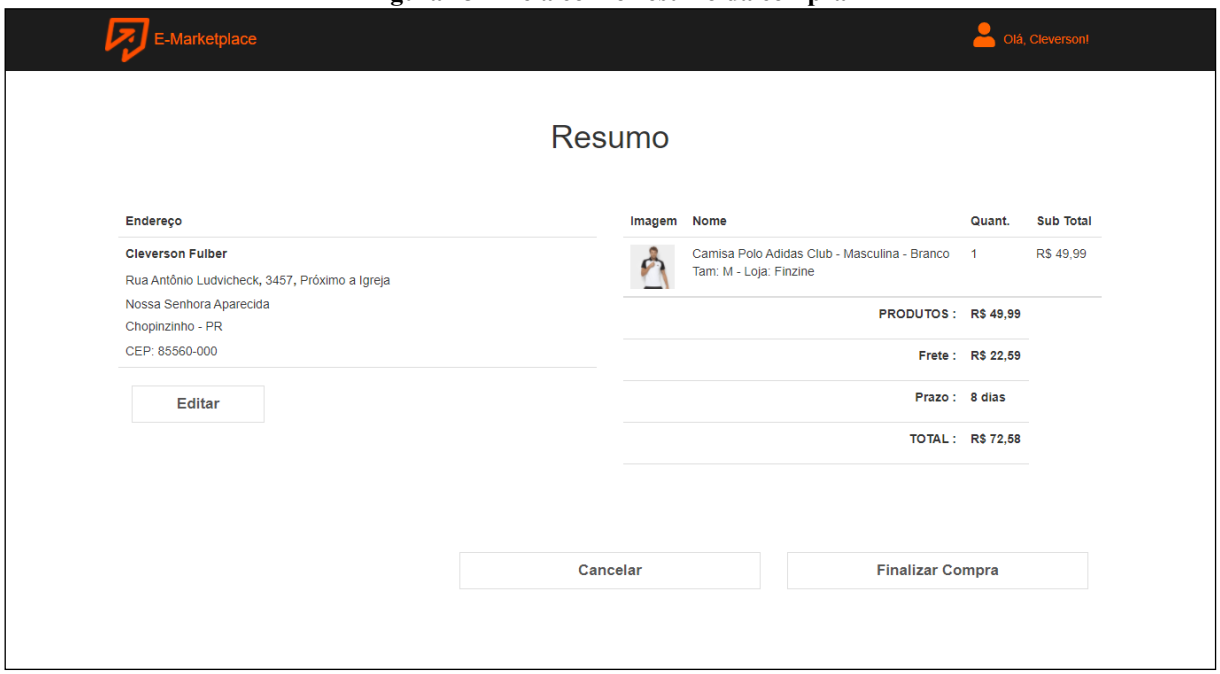
Figura 22 – Tela de seleção e cadastro de endereços para entrega da compra



Fonte: Autoria própria.

E, por fim, a Figura 23 exibe a tela com o resumo da compra, na qual o usuário poderá alterar o endereço ao clicar em “Editar”, calcular o frete conforme o endereço selecionado e calcular os valores de subtotal e total que são calculados automaticamente conforme os produtos adicionados e o cálculo do frete. Ao clicar em “Finalizar Compra”, o usuário receberá um *e-mail* com a confirmação de sua compra, os dados do endereço e dos produtos comprados.

Figura 23 – Tela com o resumo da compra



Fonte: Autoria própria.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Visando tornar o código do desenvolvimento das interfaces do sistema menos verboso foram utilizados dois leiautes na aplicação que contém o cabeçalho e o rodapé da página, que são padronizados, e acessados conforme a página solicitada.

No primeiro arquivo com o leiaute padrão, denominado de “layout.html”, fica o código utilizado nas páginas administrativas, de cadastro e validação dos dados, e no segundo, denominado de “layoutinicial.html” é o leiaute utilizado na página inicial, dos produtos e nas páginas individuais.

A Listagem 1 apresenta o código do arquivo “layout.html”, no qual é possível observar o uso do *framework* Thymeleaf, utilizado durante o código por meio das *tags* “th:” e “layout:”.

Listagem 1 – Estrutura do leiaute padrão fornecido pelo Thymeleaf

```
<!DOCTYPE html>
<html lang="pt" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
   <title>E-Marketplace</title>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width, initial-scale=1,
      shrink-to-fit=no" />
</head>
<body>
```

```
    <header th:replace="fragments/header :: cabecalho">
       <div>header</div>
    </header>

    <div  class="layout-main">
       <aside>
          <nav th:replace="fragments/sidebar :: nav-sidebar">
             <span>menu</span>
          </nav>
       </aside>

       <section layout:fragment="corpo">
          <div>conteudo principal das páginas</div>
       </section>
    </div>

    <footer th:replace="fragments/footer :: rodape">
       <div>footer</div>
    </footer>
</body>
</html>
```

Fonte: Autoria própria.

O framework Thymeleaf permite a criação de trechos de código, chamados fragmentos, que podem ser utilizados em mais de uma página. Para acrescentar um fragmento na página é utilizado o código “th:fragment="nome_fragmento"” conforme mostrado na Listagem 2, utilizado para o cabeçalho no arquivo denominado “header.html”.

Listagem 2 – Fragmento do cabeçalho padrão

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head><meta charset="UTF-8"></head>
<body>
   <header th:fragment="cabecalho">
      <nav class="navbar navbar-inverse navbar navbar-dark bg-dark fixed-
top">
         <div class="container-fluid">
            <div class="navbar-header col-md-10">
               <button type="button" class="btn btn-dark navbar-toggle
pull-left">
                  <i class="oi oi-menu"></i>
               </button>
               <img th:src="@{/image/icone.png}">
               <a class="navbar-brand " href="/">E-Marketplace</a>
            </div>
            <div class="usuario">
               <sec:authorize access="isAuthenticated()">
                  Olá, <span sec:authentication="principal.nome">
                     Erro!!!
                  </span>!
               </sec:authorize>
            </div>
            <div>
```

```
                <a th:href="@{/logout}" class="btn btn-outline-danger"><i
class="fa fa-sign-out-alt"></i> Sair</a>
            </div>
         </div>
      </nav>
   </header>
</body>
</html>
```

Fonte: Autoria própria.

Para acrescentar o corpo ao código deverá ser acrescentado a *tag* *"layout:decorate="~{layout/layout}""* e o código *"layout:fragment="corpo""* conforme apresentado na Listagem 3.

Listagem 3 – Fragmento do corpo da tela de categorias

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
     xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
     layout:decorate="~{layout/layout}">
<head><meta charset="UTF-8"/></head>
<body>

<section class="layout-content" layout:fragment="corpo">

   <nav class="navbar navbar-expand-md bg-light">
      <div class="collapse navbar-collapse" id="navbarsExampleDefault">
         <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
               <i class="oi oi-caret-right"></i>
               <span>Lista de Categorias</span>
            </li>
         </ul>
         <div class="row">
            <div class="col-xs-4">
               <a class="btn btn-primary" onclick="novo()"
                  data-toggle="modal" data-target="#modal-form">
                  <i class="oi oi-plus"></i> Novo Registro   </a>
            </div>
         </div>
      </div>
   </nav>


   <div class="container-fluid">
      <div class="alert  alert-success"
          th:if="${!#strings.isEmpty(mensagem)}">
         <i class="fa fa-check-circle"></i> <span
th:text="${mensagem}">Mensagem de sucesso!</span>
      </div>

      <div class="table-responsive">
         <table id="minhaTable" class="table table-striped
                table-bordered  table-hover  table-condensed">

            <thead class="ut-table-header-solid">
               <tr>
                  <th>Código</th>
                  <th>Nome</th>
```

```
                    <th>Ações</th>
                </tr>
            </thead>
            <tbody>
            <tr th:id="'row_' + ${categoria.id}" th:each="categoria :
${categorias}">

                <td style="width:150px" class="text-center"
th:text="${categoria.id}">0</td>
                <td><a class="texto"
th:href="'javascript:editar(\'/categorias/ajax/'
                    + ${categoria.id} + '\' );'"
th:text="${categoria.nome}">...</a>
                </td>
                <td style="width:200px" class="text-center">
                    <a th:href="'javascript:editar(\'/categorias/ajax/' +
${categoria.id} + '\' );'"
                        class="btn btn-primary btn-xs botao"> <span
class="oi oi-brush" title="Editar" aria-hidden="true"></span>
                    </a>  
                    <a th:href="'javascript:excluir('+ ${categoria.id}
+',\'/categorias\');'"
                        class="btn btn-danger btn-xs botao"><span class="oi
oi-circle-x " title="Exclusão desabilitada" aria-
hidden="true"></span></span></i></a>
                </td>

            </tr>
            </tbody>
        </table>
    </div>
</div>


<div class="modal modal-form" th:id="modal-form">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h3>Categoria</h3>
                <button type="button" class="close" data-dismiss="modal"
                    onclick="limpar()" aria-hidden="true">x</button>
            </div> <!-- modal-header -->
            <div class="modal-body">
                <form id="frm" name="frm"
                    method="POST" class="form-vertical"
                    th:action="@{/categorias/ajax}">

                    <input type="hidden" id="id" name="id"/>
                    <div class="form-group">
                        <label for="nome">Nome</label>
                        <input type="text" class="form-control" id="nome"
                            name="nome" />
                    </div>
                </form>
            </div> <!-- modal-body -->
            <div class="modal-footer">
                <button type="button" class="btn" data-dismiss="modal"
                    onclick="limpar()" aria-
hidden="true">Cancelar</button>
                <button type="button" class="btn btn-success"
                    onclick="salvar('/categorias')">Salvar</button>
```

```
            </div>
         </div> <!-- modal-content -->
      </div> <!-- modal-dialog -->
   </div> <!-- modal-form -->
</section>

</body>
</html>
```

Fonte: Autoria própria.

Na Listagem 4 é mostrada a classe de produtos, na qual é possível observar a utilização da anotação *@Data*, disponibilizada pelo uso do *Lombok* que é uma biblioteca Java focada em produtividade e redução de código. A anotação *@Data* serviu como um atalho para criar automaticamente os métodos *getters* e *setters* e sobrescrever o método *toString* das classes. Também foi utilizado o *@Entity* para estabelecer a conexão entre uma classe, com uma tabela com o mesmo nome no banco de dados, e a anotação *@Table* define o nome da tabela no banco de dados.

Listagem 4 – Classe de produtos

```
package br.edu.utfpr.tcc.model;

import lombok.Data;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Set;

@Data
@Entity
@Table(name = "produtos")
public class Produto {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @Column(name = "nome", nullable = false, length = 60)
   private String nome;

   @Column(name = "valor", nullable = false)
   private Double valor;

   @Column(name = "imagem", length = 100, nullable = true)
   private String imagem;

   @Column(name = "descri", nullable = false, length = 254)
   private String descri;

   @Column(name = "caracteristica", nullable = false, length = 254)
   private String caracteristica;

   @ManyToOne
   @JoinColumn(name = "cor_id", referencedColumnName = "id")
```

```
    private Cor cor;

    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private Set<Tamanho> tamanhos;

    @ManyToOne
    @JoinColumn(name = "categoria_id", referencedColumnName = "id")
    private Categoria categoria;

    @ManyToOne
    @JoinColumn(name = "marca_id", referencedColumnName = "id")
    private Marca marca;

    @ManyToOne
    @JoinColumn(name = "tipo_id", referencedColumnName = "id")
    private Tipo tipo;

    @Column(name = "qtde", nullable = false)
    private Integer qtde;

    @ManyToOne
    @JoinColumn(name = "anuncio_id", referencedColumnName = "id")
    private Anuncio anuncio;

    @ManyToOne
    @JoinColumn(name = "promocao_id", referencedColumnName = "id")
    private Promocao promocao;


    public Collection<?> getTamanhos() {
        List<Tamanho> list = new ArrayList<>();
        list.addAll(tamanhos);
        return list;
    }
}
```

Fonte: Autoria própria.

Na Listagem 5 é mostrada a classe denominada “ProdutoController”, na qual é possível gerenciar várias tarefas no sistema, como, por exemplo, produtos que devem ser mostrados, onde e como serão salvos. Juntamente com os demais *controllers* do sistema, eles serão responsáveis por controlar todo o fluxo de informações do sistema. Nestas classes percebe-se a utilização das anotações @Controller e @RequestMapping, este último, define a *Uniform Resource Locator* (URL) que deve ser acessada para receber as requisições.

Listagem 5 – Classe controladora de produtos

```
package br.edu.utfpr.tcc.controller;

import br.edu.utfpr.tcc.model.Produto;
import br.edu.utfpr.tcc.model.Usuario;
import br.edu.utfpr.tcc.model.service.UsuarioService;
import br.edu.utfpr.tcc.repository.*;
import br.edu.utfpr.tcc.services.S3Services;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;

@Controller
@RequestMapping("/produtos")
public class ProdutoController {

   @Autowired
   S3Services s3Services;

   @Autowired
   private ProdutoRepository produtoRepository;
   @Autowired
   private UsuarioService usuarioService;

   @Autowired
   private CategoriaRepository categoriaRepository;
   @Autowired
   private MarcaRepository marcaRepository;
   @Autowired
   private TipoRepository tipoRepository;
   @Autowired
   private PromocaoRepository promocaoRepository;
   @Autowired
   private TamanhoRepository tamanhoRepository;
   @Autowired
   private CorRepository corRepository;

   @Autowired
   private AnuncioRepository anuncioRepository;

   @GetMapping
   public ModelAndView listar() {

      Usuario usuario = (Usuario) usuarioService.loadUserByUsername(

SecurityContextHolder.getContext().getAuthentication().getName()
      );

      ModelAndView modelAndView = new ModelAndView("produto/lista");
      modelAndView.addObject("produtos",
produtoRepository.buscarProdutoUsuario(usuario.getId()) );
      modelAndView.addObject("categorias", categoriaRepository.findAll()
);
      modelAndView.addObject("marcas", marcaRepository.findAll() );
      modelAndView.addObject("tipos", tipoRepository.findAll() );
      modelAndView.addObject("promocoes", promocaoRepository.findAll() );
      modelAndView.addObject("tamanhos", tamanhoRepository.findAll() );
```

```
        modelAndView.addObject("cores", corRepository.findAll() );
        modelAndView.addObject("produto", new Produto());

        return modelAndView;
    }

    @GetMapping({"novo"})
    public ModelAndView novo(Produto produto) {
        ModelAndView modelAndView = new ModelAndView("produto/lista");

        if (produto != null) {
            modelAndView.addObject(produto);
        }else {
            modelAndView.addObject(new Produto());
        }
        return modelAndView;
    }

    @DeleteMapping("{id}")
    public ResponseEntity<?> excluir(@PathVariable Long id){
        try {
            tamanhoRepository.excluirTamanhoProduto(id);
            produtoRepository.deleteById(id);
            return new ResponseEntity<>(HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
    }

    @GetMapping("ajax/{id}")
    @ResponseBody
    public Produto editar(@PathVariable Long id) {
        produtoRepository.validarAnuncio(id);

        return produtoRepository.findById(id).orElse(new Produto());
    }

    //método para salvar com upload de arquivos
    @PostMapping("upload")
    public ResponseEntity<?> salvar(@Valid Produto produto, BindingResult
result,
 @RequestParam("anexos") MultipartFile[] anexos,
                            HttpServletRequest request){

        if ( result.hasErrors() ) {
            return new ResponseEntity<>(result.getAllErrors(),
HttpStatus.BAD_REQUEST);
        }

        produtoRepository.save(produto);

        if (anexos.length > 0 &&
!anexos[0].getOriginalFilename().isEmpty()) {
            saveFile(produto.getId(), anexos, request);
        }

        return new ResponseEntity<>(HttpStatus.OK);
    }

    private void saveFile(Long id, MultipartFile[] anexos,
HttpServletRequest request) {
```

```
      File dir = new
File(request.getServletContext().getRealPath("/images/"));
      if (!dir.exists()) { //verifica se o diretório de armazenamento
existe
         dir.mkdirs(); //não existindo, cria o diretório
      }

      String caminhoAnexo =
request.getServletContext().getRealPath("/images/");
      int i = 0;
      for (MultipartFile anexo : anexos) {
         i++;
         String extensao = anexo.getOriginalFilename().substring(
               anexo.getOriginalFilename().lastIndexOf("."),
               anexo.getOriginalFilename().length());

         String nomeArquivo = id + "_" + i + extensao;

         try {
            FileOutputStream fileOut = new FileOutputStream(
                  new File (caminhoAnexo + nomeArquivo));

            BufferedOutputStream stream = new
BufferedOutputStream(fileOut);
            stream.write(anexo.getBytes());
            stream.close();

            String url = s3Services.uploadFile(nomeArquivo, caminhoAnexo
+ nomeArquivo);
            Produto p = this.produtoRepository.getOne(id);
            p.setImagem(url);
            this.produtoRepository.save(p);
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
   }

   @GetMapping("/anuncios/{id}")
   public ModelAndView anuncio(@PathVariable Long id){

      Usuario usuario = (Usuario) usuarioService.loadUserByUsername(

SecurityContextHolder.getContext().getAuthentication().getName()
      );

      ModelAndView modelAndView = new ModelAndView("produto/lista");
      modelAndView.addObject("produtos",
produtoRepository.buscarProdutoAnuncio(id,usuario.getId()) );
      modelAndView.addObject("categorias", categoriaRepository.findAll()
);
      modelAndView.addObject("marcas", marcaRepository.findAll() );
      modelAndView.addObject("tipos", tipoRepository.findAll() );
      modelAndView.addObject("promocoes", promocaoRepository.findAll() );
      modelAndView.addObject("tamanhos", tamanhoRepository.findAll() );
      modelAndView.addObject("cores", corRepository.findAll() );
      modelAndView.addObject("usuncios", anuncioRepository.getOne(id) );
      modelAndView.addObject("produto", new Produto());

      return modelAndView;
```

```
    }
}
```

Fonte: Autoria própria.

Na Listagem 5 pode-se observar a utilização do *Amazon Web Services* (AWS) por meio do serviço de armazenamento S3, no qual é possível salvar arquivos em nuvem por meio de serviços da *web*. Para configurar o acesso ao serviço e poder armazenar os arquivos, foram utilizadas as linhas de códigos apresentadas na Listagem 6, na qual observa-se a utilização da anotação *@Configuration* que indica que a classe utiliza de um ou mais métodos *@Beans*, onde é utilizado a anotação *@Value*, que instrui que valores externos são utilizados nestas linhas. Valores externos estes que são encontrados no arquivo *application.properties* do sistema, mas que não podem ser compartilhados por motivos se segurança estabelecidos pelo próprio servidor.

Listagem 6 – Configurações do servidor S3

```
package br.edu.utfpr.tcc.config;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class S3Config {
   @Value("${jsa.aws.access_key_id}")
   private String awsId;

   @Value("${jsa.aws.secret_access_key}")
   private String awsKey;

   @Value("${jsa.s3.region}")
   private String region;

   @Bean
   public AmazonS3 s3client() {

      BasicAWSCredentials awsCreds = new BasicAWSCredentials(awsId,
awsKey);
      AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                          .withRegion(Regions.fromName(region))
                                .withCredentials(new
AWSStaticCredentialsProvider(awsCreds))
                                .build();

      return s3Client;
   }
}
```

Fonte: Autoria própria.

A Listagem 7 exibe o código referente à implementação do método *uploadFile*, que irá salvar os arquivos no servidor em um "*Bucket" container* de dados, ou caso contrário retornará um erro. O arquivo será salvo pegando o código do produto ao qual se refere "*keyName*", com o pacote de arquivos escolhidos pelo usuário "*uploadFilePath*".

Listagem 7 – Implementação da classe salvar arquivos do S3

```
package br.edu.utfpr.tcc.services.impl;

import br.edu.utfpr.tcc.services.S3Services;
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.model.PutObjectRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.io.File;

@Service
public class S3ServicesImpl implements S3Services {

   private Logger logger = LoggerFactory.getLogger(S3ServicesImpl.class);

   @Autowired
   private AmazonS3 s3client;

   @Value("${jsa.s3.bucket}")
   private String bucketName;

   @Override
   public String uploadFile(String keyName, String uploadFilePath) {

      try {

         File file = new File(uploadFilePath);
           s3client.putObject(new PutObjectRequest(bucketName, keyName,
file));
           logger.info("====================== Upload File - Done!
======================");
           return s3client.getUrl(bucketName, keyName).toString();
      } catch (AmazonServiceException ase) {
         logger.info("Caught an AmazonServiceException from PUT requests,
rejected reasons:");
         logger.info("Error Message:    " + ase.getMessage());
         logger.info("HTTP Status Code: " + ase.getStatusCode());
         logger.info("AWS Error Code:   " + ase.getErrorCode());
         logger.info("Error Type:       " + ase.getErrorType());
         logger.info("Request ID:       " + ase.getRequestId());
        } catch (AmazonClientException ace) {
            logger.info("Caught an AmazonClientException: ");
            logger.info("Error Message: " + ace.getMessage());
        }
      return "";
   }
```

```
}
```

Fonte: Autoria própria.

Para acessar de forma pública os arquivos salvos no servidor, foi adicionado uma política de privacidade ao *Bucket*, o qual torna possível que qualquer pessoa possa visualizar os arquivos.

Listagem 8 – Política do *Bucket* no servidor

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPerm",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tcc-cleversonfulber/*"
        }
    ]
}
```

Fonte: Autoria própria.

Na Listagem 9 é definida a tela de cadastro de usuários, que poderá ser pessoa física ou jurídica, sempre cadastrado por padrão como um “USER”, sem permissões administrativas.

Listagem 9 – Código HTML de cadastro de usuário

```
<!DOCTYPE html>
<html lang="pt" xmlns="http://www.w3.org/1999/xhtml"
   xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1" />

   <title>Cadastrar</title>
</head>

<body >
   <nav id="top">
      <div class="container">
         <div class="row">
            <div class="col-md-9">
               <a class="navbar-brand " href="/"><img
th:src="@{/image/icone.png}" height="50px" width="50px">
                  E-Marketplace</a>
            </div>
         </div>
      </div>
   </nav>
   <div class="container">
      <div class="superior">
      </div>
```

```
      <div >
         <form id="frm" name="frm"
               method="POST"
               th:action="@{/usuario}">
            <h1 class="h3 mb-3 text-center top">Cadastrar</h1>
            <span class="obrigatorios">* Campos Obrigatórios</span>
            <div class="row">
               <div class="col">
                  <input type="hidden" id="id" name="id"/>
                  <div class="form-group">
                     <label for="username">E-mail*</label>
                     <input type="text" class="form-control"
placeholder="email@email.com"
                            id="username" name="username" required />
                  </div>
                  <input type="radio" name="FlgPontua" value="cpf"
checked> Pessoa Física      
                  <input type="radio" name="FlgPontua" value="cnpj">
Pessoa Juridíca
                  <div class="cpf" >
                     <div class="form-group">
                        <label for="cpfCnpj cpf">CPF*</label>
                        <input type="text" class="form-control"
placeholder="000.000.000-00"
                               id="cpfCnpj cpf" name="cpfCnpj"
                               required minlength="14" data-
mask="000.000.000-00"/>
                     </div>
                     <div class="form-group">
                        <label for="nome">Nome*</label>
                        <input type="text" class="form-control"
placeholder="Nome"
                               id="nome" name="nome" required
minlength="3" maxlength="15"/>
                     </div>
                     <div class="form-group">
                        <label for="sobrenome">Sobrenome*</label>
                        <input type="text" class="form-control"
placeholder="Sebrenome"
                               id="sobrenome" name="sobrenome" required
minlength="3" maxlength="25"/>
                     </div>
                  </div>
                  <div class="cnpj" style="margin: 0 auto; display:
none;" >
                     <div class="form-group">
                        <label for="cpfCnpj cnpj">CNPJ*</label>
                        <input type="text" class="form-control"
placeholder="00.000.000/0000-00" disabled
                               id="cpfCnpj cnpj" name="cpfCnpj"
                               required minlength="18" data-
mask="00.000.000/0000-00"/>
                     </div>
                     <div class="form-group">
                        <label for="nome">Nome Fantasia*</label>
                        <input type="text" class="form-control"
placeholder="Nome Fantasia" disabled
                                  id="nome" name="nome" required
minlength="3" maxlength="15"/>
                     </div>
                     <div class="form-group">
```

```
                        <label for="sobrenome">Razão Social*</label>
                        <input type="text" class="form-control"
placeholder="Razão Social" disabled
                               id="sobrenome" name="sobrenome" required
minlength="3" maxlength="25"/>
                     </div>
                  </div>
               </div>

               <div class="col">
                  <div class="form-group">
                     <label for="password">Senha*</label>
                     <input type="password" class="form-control"
placeholder="Senha"
                            id="password" name="password" required
maxlength="20" />
                  </div>

                  <div class="form-group">
                     <label for="telefone">Telefone</label>
                     <input type="text" class="form-control"
placeholder="(00) 0000-0000"
                            id="telefone" name="telefone" data-mask="(00)
0000-0000" minlength="14"/>
                  </div>
                  <div class="form-group">
                     <label for="celular">Celular*</label>
                     <input type="text" class="form-control"
placeholder="(00) 00000-0000"
                            id="celular" name="celular" required data-
mask="(00) 00000-0000"/>
                  </div>

                  <div class="cpf">
                     <div class=" form-group">
                        <label for="dataNascimento">Data de
Nascimento*</label>
                        <input type="text" class="datepicker form-
control" placeholder="00/00/0000"
                               id="dataNascimento" name="dataNascimento"
required data-mask="00/00/0000"/>
                     </div>
                     <div class="form-group">
                        <label for="genero">Gênero*</label>
                        <select class="form-control" id="genero"
name="genero" >
                           <option value=""> (Selecione) </option>
                           <option value="Masculino"> Masculino </option>
                           <option value="Feminino"> Feminino </option>
                        </select>
                     </div>
                  </div>
                  <div class="form-group validar" style="display: none;">
                     <input type="checkbox" id="validar" name="validar"
value="true"> Deseja ser um parceiro?
                  </div>
                  <div class="form-group" style="display:none">
                     <input type="checkbox" checked name="permissoes"
                           th:id="'chk_' + ${2}"
                           th:value="${2}"/>
                     <label
```

```
                            th:for="'chk_' + ${2}"
                            th:text="${ROLE_USER}"></label>
                     </div>
                  </div>
               </div >
               <div class="col-12 col-md-12 final">
                  <div class="col-xs-4 col-md-4 direita">
                     <button type="button" class="view-link shutter"
                           onclick="salvarUsuario('/login')">Salvar</button>
                  </div>
                  <div class="col-xs-4 col-md-4 direita">
                     <a type="button" class="view-link shutter"
                        th:href="@{../}">Cancelar</a>
                  </div>
               </div>
            </form>
         </div>
         <div class="text-center top">
            <span class="">Já é Cadastrado?</span>
            <a th:href="@{/login}" class=" btn2">Entrar</a>
            <p class="mt-5 mb-3 text-muted text-center">© 2020</p>
         </div>
      </div>
   </body>
   </html>
```

Fonte: Autoria própria.

Na Listagem 10 é possível visualizar o código fonte referente à tela de controle de cadastro de usuários, que irá receber as requisições e salvar os dados.

Listagem 10 – Controller de cadastro de usuários

```
package br.edu.utfpr.tcc.controller;

//imports

import javax.validation.Valid;

@Controller
@RequestMapping("usuario")
public class UsuarioController{

   @Autowired
   private UsuarioRepository usuarioRepository;
   @Autowired
   private PermissaoRepository permissaoRepository;

   @GetMapping
   public ModelAndView listar() {
      ModelAndView modelAndView = new ModelAndView("cadastrar");
      modelAndView.addObject("permissoes",
permissaoRepository.findAll());
      modelAndView.addObject("usuario", new Usuario());

      return modelAndView;
   }

   @PostMapping()
```

```
   public ResponseEntity<?> saveAjax(@Valid Usuario entity,
                              BindingResult result, RedirectAttributes
attributes) {
      if (result.hasErrors()) {
         return new ResponseEntity<>(result.getAllErrors(),
HttpStatus.BAD_REQUEST);
      }

      entity.setPassword(
            entity.getEncodedPassword(entity.getPassword()));
      usuarioRepository.save(entity);

      return new ResponseEntity<>(HttpStatus.OK);
   }

}
```

Fonte: Autoria própria.

Para ter acesso ao sistema, os usuários precisarão passar por autenticação. Na Listagem 11 é possível visualizar o arquivo de configurações, no qual estão configuradas as URLs e requisições que podem ser acessadas quando é necessário realizar a autenticação no sistema, definidos com “permitAll()” e algumas linhas que somente os usuários citados terão acesso definido com “hasAnyRole(“...”)”.

Listagem 11 – Configurações de permissão

```
package br.edu.utfpr.tcc.config;

import br.edu.utfpr.tcc.model.service.UsuarioService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import
org.springframework.security.config.annotation.authentication.builders.Au
thenticationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.Enabl
eGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWe
bSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecur
ityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
```

```
    private UsuarioService usuarioService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()

            .exceptionHandling().accessDeniedPage("/403")
            .and().formLogin().loginPage("/login")
            .defaultSuccessUrl("/")
            .failureUrl("/login?error=bad_credentials").permitAll()
            .and().logout()
            .logoutSuccessUrl("/login")
            .and().authorizeRequests()
               .antMatchers("/categorias/**").hasAnyRole( "ADMIN")
               .antMatchers("/marcas/**").hasAnyRole( "ADMIN")
               .antMatchers("/tipos/**").hasAnyRole("ADMIN")
               .antMatchers("/promocoes/**").hasAnyRole( "LOJISTA", "ADMIN")
               .antMatchers("/produtos/**").hasAnyRole("LOJISTA", "ADMIN")
               .antMatchers("/usuarios/**").hasAnyRole("LOJISTA", "ADMIN")
               .antMatchers("/anuncios/**").hasAnyRole("LOJISTA", "ADMIN")
               .antMatchers("/index/**").hasAnyRole("LOJISTA", "ADMIN")
               .antMatchers("/home/**").permitAll()
               .antMatchers("/403/**").permitAll()
               .antMatchers("/**").permitAll()
               .antMatchers("/images/**").permitAll()
               .antMatchers("/fragmentsinicial/**").permitAll()
               .antMatchers("/permissoes/**").permitAll()
               .antMatchers("/empresas/**").authenticated();
    }

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring()
            .antMatchers("/css/**")
            .antMatchers("/js/**")
            .antMatchers("/images/**")
            .antMatchers("/image/**")
            .antMatchers("/assets/**")
            .antMatchers("/webjars/**");
    }

    @Bean
    public UserDetailsService userDetailsService() {
        return usuarioService;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(10);
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth.userDetailsService(userDetailsService())
            .passwordEncoder(passwordEncoder());
    }
}
```

Fonte: Autoria própria.

Na Listagem 12 é mostrada a classe UsuariosController que permite realizar a chamada de validação do parceiro, pelo método “salvarValidacao”, quando o administrador do sistema aceita a validação, e também no “cancelarValidacao” que é quando divergências são encontradas e o parceiro não é validado.

Listagem 12 – Controller de usuários

```
package br.edu.utfpr.tcc.controller;

import br.edu.utfpr.tcc.model.Usuario;
import br.edu.utfpr.tcc.repository.PermissaoRepository;
import br.edu.utfpr.tcc.repository.UsuarioRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.validation.Valid;

@Controller
@RequestMapping("usuarios")
public class UsuariosController {

   @Autowired
   private UsuarioRepository usuarioRepository;

   @Autowired
   private PermissaoRepository permissaoRepository;


   @GetMapping
   public ModelAndView listar() {
      ModelAndView modelAndView = new ModelAndView("usuario/lista");
      modelAndView.addObject("usuarios", usuarioRepository.findAll());
      modelAndView.addObject("permissoes",
permissaoRepository.findAll());
      modelAndView.addObject("usuario", new Usuario());

      return modelAndView;
   }

   @GetMapping({"novo"})
   protected ModelAndView novo(Usuario usuario) {
      ModelAndView modelAndView = new ModelAndView("usuario/lista");
      if (usuario != null) {
         modelAndView.addObject(usuario);
      }else {
         modelAndView.addObject(new Usuario());
      }
      return modelAndView;
   }

   @GetMapping("/{id}")
   @ResponseBody
```

```
    public Usuario editar(@PathVariable Long id) {
        return usuarioRepository.findById(id).orElse(null);
    }

    @GetMapping("validar/{id}")
    @ResponseBody
    public void salvarValidacao(@PathVariable Long id) {
        usuarioRepository.validarParceiro(id);

    }

    @GetMapping("cancelar/{id}")
    @ResponseBody
    public void cancelarValidacao(@PathVariable Long id) {
        usuarioRepository.cancelarParceiro(id);

    }

    @PostMapping("/")
    public ResponseEntity<?> salvar(@Valid Usuario entity,BindingResult
result,
                                    RedirectAttributes attributes) {
        if (result.hasErrors()) {
            return new ResponseEntity<>(result.getAllErrors(),
HttpStatus.BAD_REQUEST);
        }

entity.setPassword(entity.getEncodedPassword(entity.getPassword()));

        usuarioRepository.save(entity);

        return new ResponseEntity<>(HttpStatus.OK);
    }

    @DeleteMapping("{id}")
    public ResponseEntity<?> excluir(@PathVariable Long id){
        try {
            usuarioRepository.deleteById(id);
            return new ResponseEntity<>(HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
    }

}
```

Fonte: Autoria própria.

Ao acessar os métodos “salvarValidacao” ou “cancelarValidação”, ambos irão redirecionar para a interface UsuarioRepository, na qual o método irá fazer a alteração no banco por meio de uma consulta. Para isso, utilizou- se na classe as anotações @Query, que é utilizada para realizar consultas no banco de dados, @Modifying(clearAutomatically = true), que permite fazer alteração no banco e faz a limpeza após a persistência, e também a anotação @Transactional que controla as transações dentro do atributo.

Listagem 13 - Interface de validação de usuários

```
package br.edu.utfpr.tcc.repository;

import br.edu.utfpr.tcc.model.Usuario;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

public interface UsuarioRepository extends JpaRepository<Usuario, Long> {

   Usuario findByUsername(String username);

   @Transactional
   @Modifying(clearAutomatically = true)
   @Query(value = "with changed_hosts as ( update usuarios set validar =
null where id = ?1 returning * ) update usuarios_permissoes set
permissoes_id = 3 where usuario_id in (select id from changed_hosts)",
nativeQuery = true)
   int validarParceiro(Long id);

   @Transactional
   @Modifying(clearAutomatically = true)
   @Query(value = "update usuarios set validar = null where id = ?1",
nativeQuery = true)
   int cancelarParceiro(Long id);
}
```

Fonte: Autoria própria.

Na Listagem 14 são mostrados os códigos referente à inserção dos itens do carrinho. O método “lerDadosProduto(produto)” irá percorrer as linhas do produto, lerá os itens adicionados no *localStorage* (que serve para armazenar os dados na sessão do navegador), caso haja, e adicionará os itens que o cliente selecionar por meio do método “inserirCarrinho(produto)”.

Listagem 14 – Adicionar itens ao carrinho

```
    lerDadosProduto(produto){
        const infoProduto = {
            imagem : produto.querySelector('img').src,
            nome : produto.querySelector('h4').textContent,
            valor : produto.querySelector('p').textContent,
            id : produto.querySelector('h1').textContent,
            sub: produto.querySelector('p').textContent,
            tamanho: produto.querySelector('#tamanho').value,
            qtda : 1,
            loja: produto.querySelector('.loja').textContent

        }
        let produtosLS, produtosSL;

        produtosLS = this.pegarProdutosLocalStorage();

        produtosLS.forEach(function(produtoLS){
```

```
            if(produtoLS.id == infoProduto.id){
                produtosSL = produtoLS.id;
            }});
        if(infoProduto.tamanho == ""){
            Swal.fire({
                type: 'info',
                title: 'Oops',
                text: 'Selecione um tamanho',
                timer: 1500,
                showConfirmButton: false
            })
        }else{
            if(produtosSL == infoProduto.id){
                    Swal.fire({
                    type: 'info',
                    title: 'Oops',
                    text: 'Este produto já está no carrinho',
                    timer: 1500,
                    showConfirmButton: false
                })
            }else{
                this.inserirCarrinho(infoProduto);
                setTimeout(function() {
                    document.location.reload(true);
                }, 1500);

                Swal.fire({
                  type: 'success',
                  title: 'Produto adicionado ao carrinho!',
                  timer: 1500,
                  showConfirmButton: false
                })
            }
        }
    }

    inserirCarrinho(produto){
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>
                <img src="${produto.imagem}" width=100>
            </td>
            <td>${produto.nome}</td>
            <td>${produto.valor}</td>


        `;
        listaProdutos.appendChild(row);
        this.salvarProdutosLocalStorage(produto);
        this.calcularQtde();

    }

    salvarProdutosLocalStorage(produto){
        let produtos;
        produtos = this.pegarProdutosLocalStorage();
        produtos.push(produto);
        localStorage.setItem('produtos', JSON.stringify(produtos));
    }
```

```
    calcularQtde(){

        let produtosLS;
        let qtde = 0;
        produtosLS = this.pegarProdutosLocalStorage();
        for(let i = 0; i < produtosLS.length; i++){
            qtde++;
        }

        document.getElementById('qtde').innerHTML = qtde;
    }
```

Fonte: Autoria própria.

Para pegar os itens adicionados ao *localStorage*, e mostrá-los no carrinho foi criado o método "lerLocalStorage()", que pegará os itens existentes e mostrará por meio de linha HTML.

Listagem 15 – Mostrar itens no carrinho

```
    lerLocalStorage(){
        let produtoLS;
        produtoLS = this.pegarProdutosLocalStorage();
        produtoLS.forEach(function(produto){
            const row = document.createElement('tr');
            row.innerHTML = `
                <td>
                    <img src="${produto.imagem}" width=100>
                </td>
                <td>${produto.nome}</td>
                <td>R$ ${produto.valor.replace('.',',')}</td>
            `;
            listaProdutos.appendChild(row);
        });

        if(produtoLS.length === 0) {
            $("tr.sem-registros").show();

        }
        else{
            $("tr.sem-registros").hide();
        }
        this.calcularQtde();
    }

    pegarProdutosLocalStorage(){
        let produtoLS;

        if(localStorage.getItem('produtos') == null){
            produtoLS = [];
        }
        else{
            produtoLS = JSON.parse(localStorage.getItem('produtos'));
        }
        return produtoLS;
    }
```

Fonte: Autoria própria.

Na Listagem 16 são mostrados os códigos que vão excluir um item do carrinho “excluirProduto(e)”, que encontrará o código do item selecionado e excluirá do *localStorage*, e também o código que esvazia o carrinho “esvaciarCarrinho(e)” e limpa o *localStorage*.

Listagem 16 – Excluir itens do carrinho

```
excluirProduto(e){
    e.preventDefault();

    const produtoID = e.target.parentElement.parentElement
        .parentElement.querySelector('h1').innerText;

    if(e.target.parentElement.parentElement.parentElement.localName ==
'tr'){
        e.target.parentElement.parentElement.parentElement.remove();
    }
    else {

e.target.parentElement.parentElement.parentElement.parentElement.remove()
;
    }

    this.excluirProdutoLocalStorage(produtoID);
    this.calcularTotal();

    if((this.pegarProdutosLocalStorage() || []).length === 0)
    {
       $("tr.sem-registros").show();
       $("tr.com-registros").hide();
       $("div.form-group").hide();
   }
   else{
       $("tr.com-registros").show();
       $("tr.sem-registros").hide();
       $("div.form-group").show();
   }
}

excluirProdutoLocalStorage(produtoID){
    let produtoLS;
    produtoLS = this.pegarProdutosLocalStorage();
    localStorage.setItem('produtos', JSON.stringify(produtoLS.filter(x =>
x.id != produtoID)));
}

esvaciarCarrinho(e){
    e.preventDefault();
    while(listaProdutos.firstChild){
        listaProdutos.removeChild(listaProdutos.firstChild);

    }
    const row = document.createElement('tr');
    row.innerHTML = `
            <td class="sem-registros">
                <h5>O carrinho está vazio.</h5>
            </td>
    `;

    this.esvaciarLocalStorage();
    listaProdutos.appendChild(row);
```

```
    return false;
}

esvaciarLocalStorage(){
    localStorage.clear();
    this.calcularQtde();
}
```

Fonte: Autoria própria.

5 CONCLUSÃO

Este trabalho teve como objetivo realizar o desenvolvimento de uma aplicação *web* de *e-marketplace* para comércio de materiais esportivos. A aplicação visa facilitar a venda dos produtos pelo parceiro, por meio de um único ambiente, sendo implementada utilizando principalmente recursos do Thymeleaf e do *framework* Spring.

Após o levantamento dos requisitos foi desenvolvido o diagrama de casos de uso para identificar as funcionalidades de cada ator e foi feita a modelagem do banco de dados. Posteriormente, foram desenvolvidas as interfaces, a estrutura do servidor, as classes de persistência e os *controllers* da aplicação.

Dentre as dificuldades encontradas durante o desenvolvimento do trabalho proposto, pode-se citar a falta de experiência na implementação do projeto usando as tecnologias supracitadas. Porém, com o auxílio de cursos *on-line*, acompanhamento de aulas extras na universidade nas disciplinas de programação para a *web*, fóruns e da documentação das tecnologias, foi possível concluir o trabalho.

O Spring é um *framework* que se mostrou bastante útil no desenvolvimento desse trabalho, pois fornece todos os recursos necessários para o desenvolvimento *web*, como, módulos para persistência de dados, integração, segurança, testes. Além disso, peremite criar soluções menos acopladas e mais coesas devido aos conceitos de injeção de dependência e inversão de controle, tornando, assim, mais fáceis de manter e de compreender. O Thymeleaf é uma boa alternativa ao *JavaServer Pages* (JSP) como *engine* para Java, sendo que as páginas possuem uma boa integração com o Spring, e sua sintaxe permite criar variáveis, mensagens, *links*, fragmentos de marcação e movê-los pelos modelos, condicionais e iteração.

Como trabalhos futuros da aplicação desenvolvida, sugere-se novas implementações, como, por exemplo, salvar os pedidos realizados pelo cliente, desenvolver funcionalidades de receitas para que os parceiros e administradores possam gerenciar seus lucros emitindo relatórios por períodos escolhidos por eles e implementar um sistema de pagamento das compras realizadas pelo site.

REFERÊNCIAS

COBRA, Marcos. **Administração de Marketing no Brasil**. 3 ed. Campus, 2009.

E-BIT. **31ª edição WebShoppers**: Comércio eletrônico cresce 24% em 2014 e maior acesso aos smartphones ajuda a alavancar mobile commerce. 2015. Disponível em: http://www.ebitempresa.com.br/clip.asp?cod_noticia=3958&pi=1. Acesso em: 29 ago. 2017.

GILBERTONI, Mariana. **Marketplaces: sua loja já aderiu?**. 2014. Disponível em: http://www.ecommercebrasil.com.br/artigos/marketplaces-sua-loja-ja-aderiu. Acesso em: 29 ago. 2017.

HONG, Ilyoo B; CHO, Hwihyung. The Impact of Consumer Trust on Attitudinal Loyalty and Purchase Intention in B2C *E-marketplace*: Intermediary Trust vs Seller Trust. International **Journal of Information Management**, 31, 2011.

HUGHES D., Assessing the value of electronic marktpelaces for business to business trading. **Proceedings of the second Internacional Conference of Electronic Trade in the CIS and Eastern European Countries**, 2000.

KOTLER, P.; KELLER, K.L. **Administração de marketing**. 12 ed. São Paulo: Pearson Prentice Hall, 2006.

LAUDON, Kenneth C.; LAUDON, Jane P.; **Gereneciamento de Sistemas de Informação**: Comercio Eletronico e Tecnologias de Negocios Eletronicos. Ed III. Rio de Janeiro: LTC, 2001.

LAUDON, Kenneth C.; LAUDON, Jane P.; **Sistema de Informação Gerenciais**: Comercio Eletronico: mercados digitais, mercadorias digitais. Ed VII. São Paulo: Pearson, 2007.

MORINISHI, Marcio Toyoki; GUERRINIB, Fábio Muller. Formação de redes de cooperação para o desenvolvimento de e-marketplaces verticais. **Produção**, v. 21, n. 2, p. 355-365, 2011.

PRESSMAN, Roger; MAXIM, Bruce. Engenharia de Software, 8 ed.. McGraw Hill Brasil, 2016.

SASSE, Maicon; BRAZ JUNIOR Osmar de Oliveira. Projetando uma API de serviços paradisponibilização de um marketplace. **Revista Científica do Alto Vale do Itajaí**, 2015.

SEBRAE. **Conheça as vantagens do *e-marketplace* para os pequenos negócios**. Disponível em: https://www.sebrae.com.br/sites/PortalSebrae/artigos/conheca-as-vantagens-do-*e-marketplace*-para-os-pequenos-negocios,3f6402b5b0d36410VgnVCM1000003b74010aRCRD. Acesso em: 29 ago. 2017.

TORRES, Norberto. Principais fatores de sucesso para o varejo online. 2013. Disponível em: https://www.ecommercebrasil.com.br/artigos/principais-fatores-de-sucesso-para-o-varejo-online. Acesso em: 30 nov. 2020.

TURBAN, Efraim. **Comércio eletrônico: estratégia e gestão**. São Paulo: Prentice Hall, 2004.