



Instituto Federal Catarinense  
**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**  
*Campus Blumenau*

**JONAS DA SILVA STASIAK**

**DESENVOLVIMENTO DE APLICAÇÃO WEB PARA E-COMMERCE DE  
VESTUÁRIO, COM USO DE CAMADAS E PADRÕES DE PROJETO**

Blumenau

2021

**JONAS DA SILVA STASIAK**

**DESENVOLVIMENTO DE APLICAÇÃO WEB PARA E-COMMERCE DE  
VESTUÁRIO, COM USO DE CAMADAS E PADRÕES DE PROJETO**

Artigo apresentado como requisito parcial à  
conclusão do curso de Tecnologia em Análise  
e Desenvolvimento de Sistemas, *campus*  
Blumenau, Instituto Federal Catarinense.

Orientador: Prof.<sup>(a)</sup>. Hylson Vescovi Netto, Dr.

Blumenau

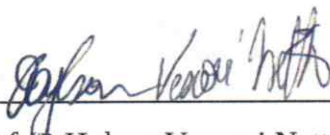
2021

**JONAS DA SILVA STASIAK**

**DESENVOLVIMENTO DE APLICAÇÃO WEB PARA E-COMMERCE DE  
VESTUÁRIO, COM USO DE CAMADAS E PADRÕES DE PROJETO**

Este artigo foi julgado adequado para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas e aprovado em sua forma final pelo curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal Catarinense – *Campus Blumenau*.

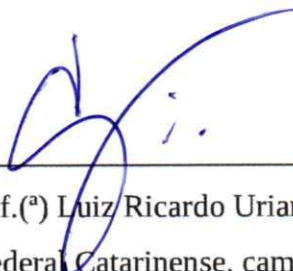
Orientador: Prof.(<sup>a</sup>). Hylson Vescovi Netto, Dr.



Prof.(<sup>a</sup>) Hylson Vescovi Netto, Dr.

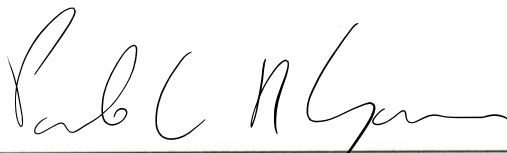
Orientador – IFC *campus* Blumenau

**BANCA EXAMINADORA**



Prof.(<sup>a</sup>) Luiz Ricardo Uriarte, Dr.

Instituto Federal Catarinense, *campus* Blumenau



Prof.(<sup>a</sup>) Paulo Cesar Rodacki Gomes, Dr.

Instituto Federal Catarinense, *campus* Blumenau

Blumenau

2021

# **Desenvolvimento de Aplicação Web para E-Commerce de Vestuário, com Uso de Camadas e Padrões de Projeto**

**Jonas da Silva Stasiak, Hylson Vescovi Netto (orientador)**

Instituto Federal de Educação, Ciência e Tecnologia Catarinense (IFC)  
Blumenau, SC - Brasil

jonas.stasiak@hotmail.com, hylson.vescovi@ifc.edu.br

**Resumo.** *Este artigo trata sobre o desenvolvimento de um sistema web de E-Commerce no âmbito de vestuário, e foi desenvolvido pelo primeiro autor, sob orientação do segundo autor, como requisito parcial para a obtenção do grau de tecnólogo em análise e desenvolvimento de sistemas no IFC Campus Blumenau. O texto inicialmente contextualiza o sistema a ser desenvolvido, faz as devidas delimitações, apresenta as técnicas utilizadas para a construção do sistema, menciona trabalhos relacionados e traz detalhes sobre o projeto e a implementação do sistema. A programação foi realizada em camadas, utilizando as linguagens C# e Javascript, com uso de padrões de projeto. O código-fonte do sistema está disponível em repositório público.*

## **1. Introdução**

Nem sempre encontrar o que se necessita é uma tarefa fácil. Às vezes há condições financeiras de pagar caro por um item de moda, um artigo esportivo ou um eletrônico importado, mas não se sabe aonde ir para encontrar o que se busca em lojas físicas de cidades convencionais, o que pode demandar tempo e dinheiro na procura por algo específico. Em busca da facilidade de encontrar itens e por ser muito mais rápido e prático realizar compras em lojas distantes, o comércio eletrônico vem facilitar a vida dos compradores e aumentar o potencial de vendas das lojas, independentemente de suas localizações (GALINARI, 2015).

*E-commerce* é definido como uma transação de venda ou compra de bens ou serviços, conduzida por meio de redes de computadores e métodos especificamente concebidos para a recepção ou efetuação de pedidos (GALINARI, 2015). O comércio eletrônico vem revolucionando a forma de organização das empresas varejistas para atender os mais diversos clientes (LASTRES, ALBAGLI, 1999). No Brasil, já existem sites consolidados onde é possível fazer compras de produtos de lojas de todo território nacional com apenas alguns passos. A entrega desses pedidos, na grande maioria das vezes, é efetuada com sucesso. Um exemplo de um site de comércio eletrônico com bom nível de confiança é o Mercado Livre, que liga vendedores do Brasil inteiro aos clientes.

Diante do contexto apresentado, esse trabalho tem como objetivo desenvolver uma loja eletrônica de venda de artigos de vestuário, onde se pode efetuar a compra de produtos através da Web.

### **1.1. Tema/Problema**

A necessidade de comprar um item de vestuário que atende aos requisitos de cor, tamanho e preço pode se transformar num grande problema quando há o conflito com outros compromissos do dia-a-dia. Nem sempre é possível reservar um período do dia em horário comercial para visitarmos lojas de vestuário no local físico da mesma. E ainda existe o risco de não encontrarmos o artigo procurado.

Pessoas com emprego formal ou autônomo têm dificuldade de tirar momentos de folga para fazer compras, tendo que, às vezes, recorrer a parentes ou conhecidos de confiança para ajudar em casos de necessidade. Ou, como geralmente é feito, ter que dedicar os finais de semana para efetuar a compra, o que acarreta grandes gargalos, aglomerações e filas que roubam muito tempo e acabam estressando o cliente.

### **1.2. Objetivos Propostos/Solução dos Problemas**

Este trabalho tem como objetivo desenvolver uma aplicação web que permite ao cliente comprar produtos em uma loja virtual. Por ser uma aplicação web, essa loja é acessada de qualquer lugar do mundo, e está 24 horas por dia disponível para os usuários. É possível, mediante um cadastro rápido, encomendar o item escolhido que supre a necessidade do usuário sem que ele precise sair do lugar no qual se encontra.

### **1.3. Escopo**

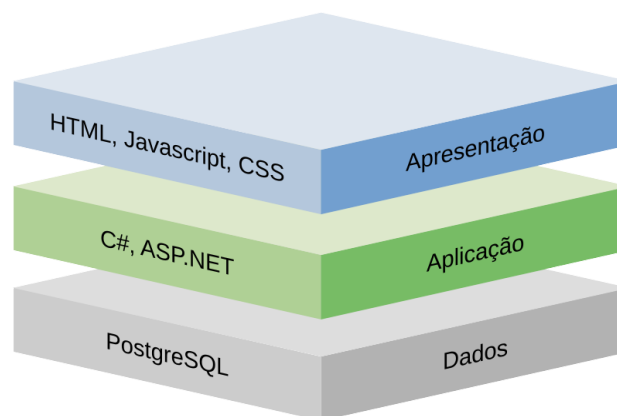
Na loja virtual é possível navegar pelos produtos disponíveis, selecionar os produtos de interesse, adicioná-los a um “carrinho de compras” e efetuar uma compra. Ao consultar os itens no carrinho, será dado início ao processo de compra dos itens selecionados. É obrigatória a realização de login de usuário, e quando necessário, efetuar cadastro do novo usuário. O sistema também abrange o local de entrega do pedido, podendo sempre ser cadastrado um endereço novo, se assim o usuário desejar. A finalização da compra disponibiliza uma área de consulta para o usuário, listando os pedidos por ele já executados, sendo possível verificar o “Status” do pedido.

### **1.4. Método de Trabalho (arquitetura, ferramentas, tecnologias aplicadas)**

O sistema web de comércio eletrônico proposto neste trabalho é desenvolvido em camadas (Figura 1). A construção de aplicações divididas em partes que se comunicam entre si, tem a intenção de modularizar o programa em partes distintas com funções fortemente estabelecidas, possibilitando o reaproveitamento de código em diversas partes do sistema, e facilita a manutenção, uma vez que as funções estando modularizadas, o concerto de determinada falha em um só lugar, vai refletir em todo a aplicação (MACORATTI, 2021).

Na camada superior, denominada Apresentação, encontra-se o cliente do sistema, que interage via navegador web. Essa camada, também conhecida como *front-end*, foi desenvolvida utilizando HTML para estruturação das páginas da aplicação. Para estilização das páginas foi utilizado CSS, fazendo uso da biblioteca Bootstrap, que é um popular *front end open source* para desenvolvimento de páginas web já implementando o desenvolvimento responsivo de sites (BOOTSTRAP, 2021).

O controle da parte visual da aplicação foi realizado com uso do framework Vue.js, que é “um framework JavaScript progressivo para a construção de interfaces visuais para o usuário usando JavaScript” (LIMA; PETRUCCELLI; DO ESPÍRITO SANTO, 2019). Também foi utilizada a biblioteca Vuex, uma das bibliotecas oficiais deste framework. Essas duas ferramentas juntas permitem ao desenvolvedor administrar, de forma clara e limpa, a comunicação de dados dentro do projeto, armazenando-os de forma centralizada e permitindo a reutilização de código entre componentes do projeto.



**Figura 1. Arquitetura de sistema em camadas. Adaptado de (LOGI, 2021).**

A camada intermediária é denominada Aplicação, e contém a lógica do sistema, onde ficam as regras de negócio. O conjunto formado pela camada de Aplicação mais a camada de Dados define o módulo conhecido como *back-end*. A camada de aplicação foi desenvolvida utilizando a linguagem de programação C# com o framework ASP.NET Core 2.2 disponibilizado pela Microsoft. O ASP.NET Core é “um framework de alta performance, open-source e cross-platform para criação de aplicações modernas, conectadas à internet e baseadas em nuvem” (KITAMURA, 2018).

Na camada de dados, o sistema de gerenciamento de banco de dados (SGBD) escolhido para administrar o banco de dados da aplicação foi o PostgreSQL 9.6.5. Esse SGBD é um sistema de gerenciamento de banco de dados objeto-relacional e foi desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. A licença liberal dessa ferramenta permite sua utilização para diversos fins acadêmicos e até comerciais sem encargos financeiros (POSTGRESQL, 2021). Além dos diversos recursos disponíveis, o SGBD permite a ampliação das funcionalidades básicas de diversas formas, como por exemplo: novos tipos de dados, funções, operadores, funções de agregação, métodos de índice e programação em linguagem procedural.

A conexão do SGBD com a aplicação foi realizada com uso de um framework de mapeamento objeto-relacional (ORM - *Object Relational Mapper*) chamado Dapper (DAPPER, 2020). Essa biblioteca permite mapear objetos de banco de dados diretamente para classes do C# sem a preocupação de lidar com comandos SQL básicos, como SELECT, UPDATE ou INSERT. Esse ORM é considerado um dos mais rápidos e

performáticos disponíveis compatíveis com ASP.NET Core. Existem outros micro ORM similares disponíveis, como o Hibernate, por exemplo.

Foram utilizados alguns padrões de projeto no desenvolvimento do *back-end*. Primeiramente, foi utilizado o padrão *Repository*, que encapsula a lógica necessária para persistir os objetos no banco (MACORATTI, 2021). Outro padrão utilizado foi o *Mediator*, que é um padrão de projeto comportamental. O *Mediator* tem por objetivo:

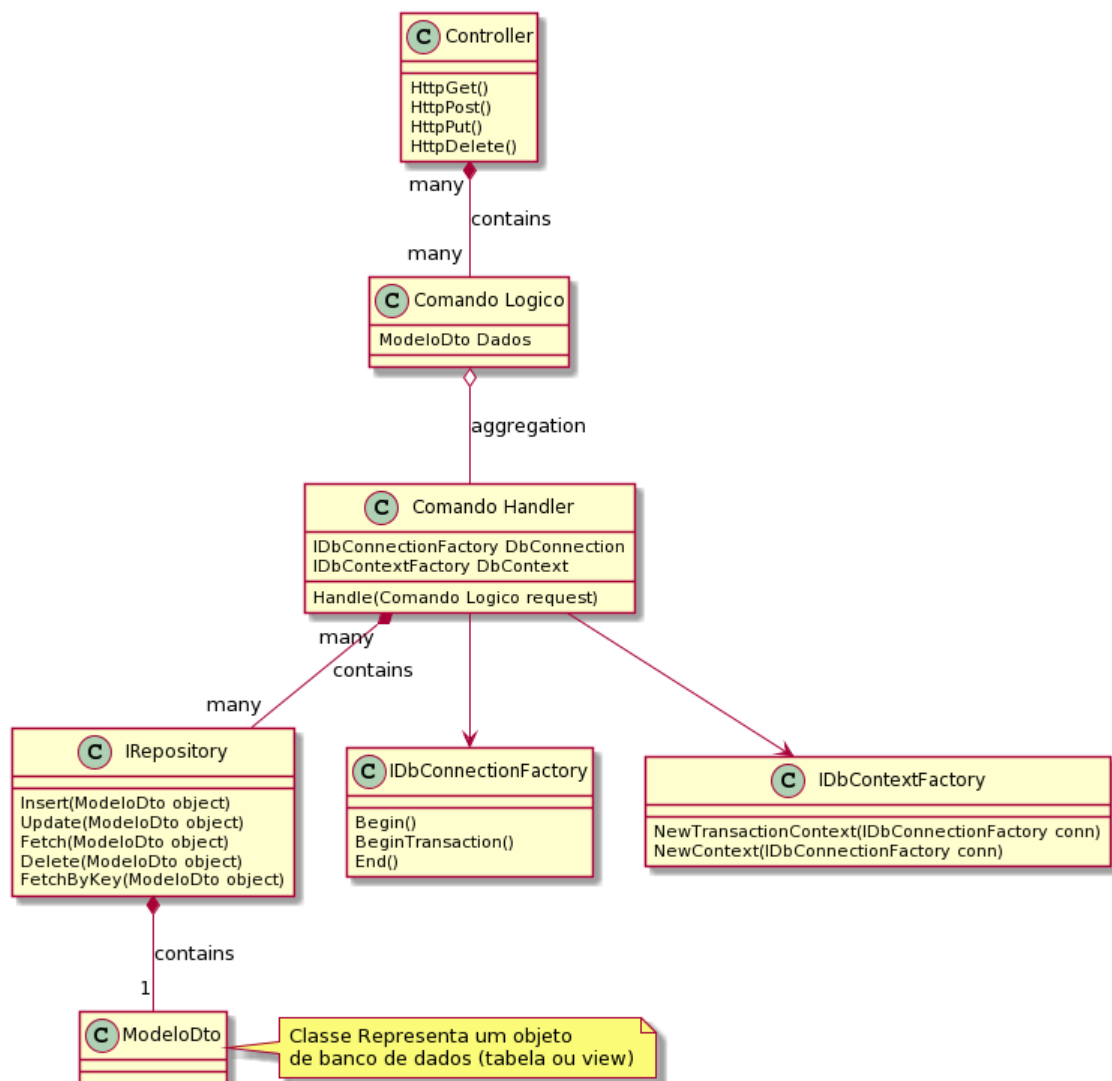
“[...] definir um objeto que encapsula a forma como um conjunto de objetos interage. O Mediator promove o acoplamento fraco ao evitar que os objetos se refiram uns aos outros explicitamente e permite variar suas interações independentemente” (PRANGE, 2019).

O padrão *Mediator* foi implementado no projeto utilizando a biblioteca *MediatR* (biblioteca da comunidade .NET) que já tem a complexidade do padrão *Mediator* implementada em classes e dependências que permitem desenvolver o projeto utilizando este padrão.

Existe uma dependência dos objetos que implementam cada comando lógico necessário para atender as regras de negócio da aplicação (Figura 2). Na composição das classes utilizadas para materializar os padrões de projeto, a classe Controller recebe as requisições HTTP, e delega para o comando lógico processar a regra. O comando Handler recebe o objeto *request*, que é um parâmetro da requisição, e utiliza informações desse parâmetro para acionar os objetos de conexão com o SGBD (IDbConnectionFactory e IDbContextFactory). Esses objetos de conexão serão necessários para que os objetos de repositório (IRepository) possam executar os métodos de persistência de dados.

A comunicação entre o front-end e o back-end acontece através do protocolo HTTP, trafegando dados no formato padrão JSON. O teste da camada de back-end é realizado sobre as APIS implementadas (API - *Application Programming Interface*), com uso da ferramenta Postman:

O Postman é um cliente para realizar testes e documentar APIs. Criado em 2012 o Postman aprimorou-se para apresentar suporte a todos os recursos necessários para o desenvolvimento e teste de API. É possível realizar solicitações HTTP como GET, POST, PUT e DELETE (SILVEIRA; LINEMBURGER, 2018).



**Figura 2. Modelo de Classe Representando Padrão de Desenvolvimento Utilizado.**  
**Fonte: o autor.**

## 2. Trabalhos Correlatos

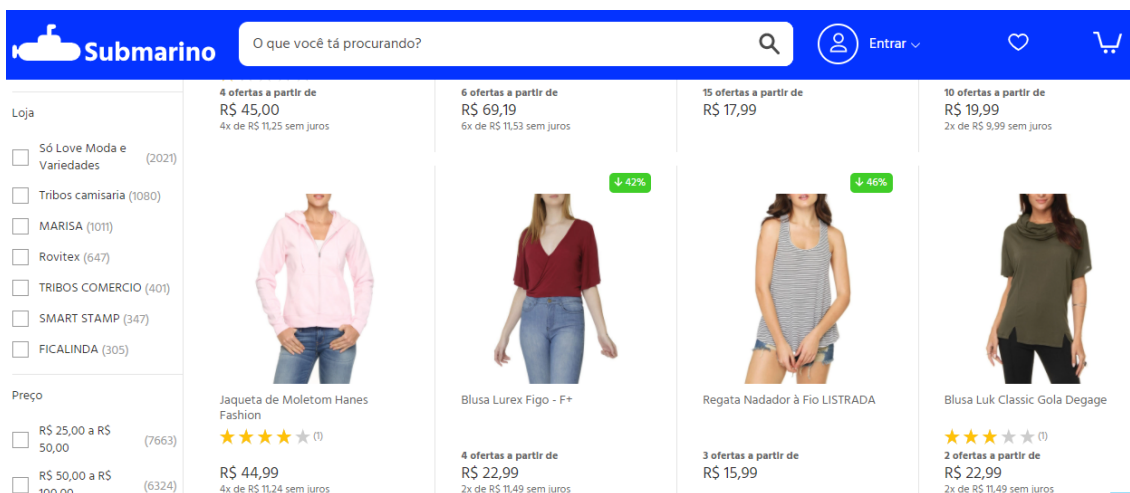
Existem diversos sites de mercado eletrônico atualmente conhecidos pelos compradores de itens pela internet. Um dos mais conhecidos é o Mercado Livre (ML), que é uma plataforma onde lojas de diversos locais podem anunciar seus produtos, que serão visualizados por compradores de muitos lugares diferentes (Figura 3). No ML, os produtos são exibidos no centro da tela, havendo na lateral esquerda uma seção que exibe filtros, a fim de selecionar quais produtos devem ser exibidos.





**Figura 3. Site Mercado Livre (MERCADOLIVRE, 2021).**

Outro comércio eletrônico bem conhecido pelos usuários atende pelo nome de Submarino. Esse site tem a mesma funcionalidade do ML, sendo um concorrente do mesmo. Diversas categorias de produtos podem ser anunciados no mesmo local (Figura 4). Na parte superior do site há um campo de busca por assunto, e na parte esquerda do site existe um menu com opções de filtro dos itens apresentados na parte central, que são os produtos disponíveis para venda.

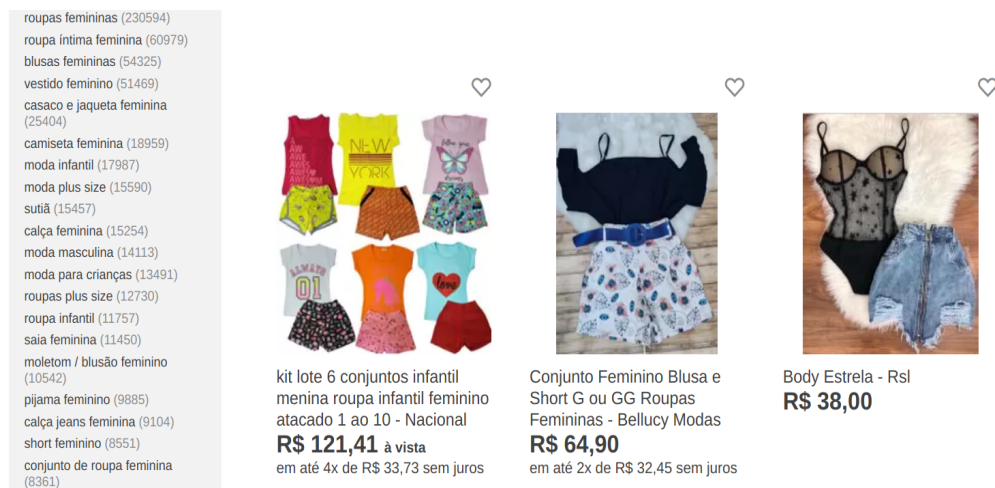


**Figura 4. Site Submarino (SUBMARINO, 2021).**

As grandes marcas varejistas em sua grande maioria possuem seus próprios e-commerces, como é o caso também, por exemplo, da loja Magazine Luiza, que é bem sucedida no mercado de comércio eletrônico, mas também possui departamento referente à vestuário. Na parte superior do site (Figura 5), existe uma barra de busca, links por departamento e área de login. Na lateral esquerda (Figura 6) encontram-se categorias pelas quais podem ser filtrados os produtos.



**Figura 5. Site Magazine Luiza, parte superior (MAGAZINELUIZA, 2021).**



**Figura 6. Site Magazine Luiza, área de categorias e produtos (MAGAZINELUIZA, 2021).**

As lojas apresentadas como exemplo nesta seção são referentes a vestuário, mas o sistema desenvolvido neste TCC tem capacidade de manipular produtos de diferentes contextos, como eletrônicos, livros, ferramentas e outros tipos de produtos.

### 3. Requisitos

A seguir encontram-se os requisitos funcionais e não-funcionais para o sistema proposto neste trabalho. No total, onze requisitos funcionais e cinco requisitos não-funcionais integram os requisitos do sistema.

#### 3.1. Requisitos Funcionais

RF001 - Apresentar itens disponíveis para comprar.

RF002 - Consultar carrinho de compras do usuário.

RF003 - Permitir excluir itens que não serão comprados.

RF004 - Manter cadastro de comprador com informações mínimas.

RF005 - Manter cadastro de endereço de entrega.

FR006 - Listar os endereços cadastrados pelo usuário

RF007 - Listar as compras efetuadas.

RF008 - Consultar o “Status” da compra.

RF009 - Autenticar o usuário.

RF010 - Mostrar detalhes do Produto.

RF011 - Listar imagens do Produto.

### **3.2. Requisitos Não-Funcionais**

RN001 - A aplicação deve ser desenvolvida para a plataforma web

RN002 - O design da aplicação deve ser responsivo, deve se comportar bem em qualquer browser, de qualquer dispositivo (smartphones, tablets).

RN003 - Deve ser desenvolvido utilizando no framework ASP.NET Core 2.2.

RN004 - Linguagem de programação deve ser C#.

RN004 - SGBD PostgreSQL 9.6 ou superior.

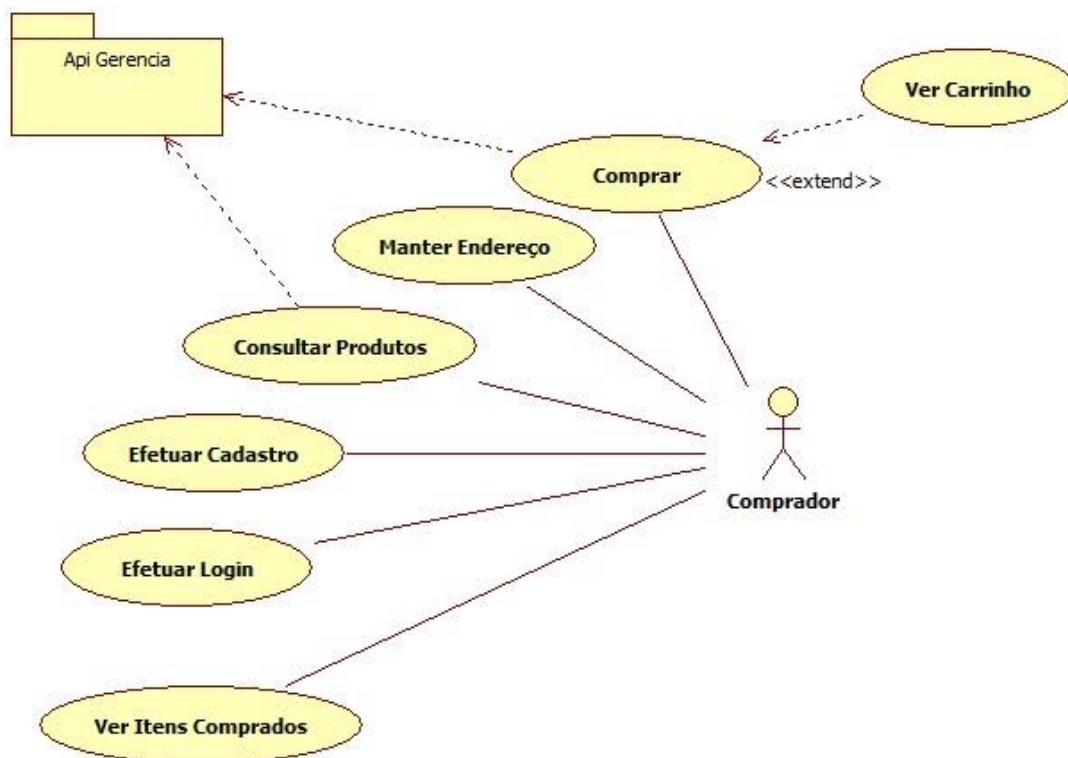
RN005 - Front-end utilizando Vue.js.

## **4. Diagramas UML**

Os diagramas UML são uma forma de representar o sistema de maneira gráfica. UML significa *Unified Modelling Language* (Linguagem de Modelagem Unificada), e é “uma linguagem gráfica para visualização, construção e documentação de artefatos de sistemas complexos de software” (BOOCH; RUMBAUGH; JACOBSON, 2006). Nas subseções a seguir se encontram: um diagrama de casos de uso, um diagrama de sequência da principal operação do sistema e um diagrama de classes que foi utilizado para representar as entidades do sistema.

### **4.1 Diagramas de Caso de Uso**

O modelo de casos de uso da aplicação possui como ator principal o Comprador, que realiza todas as ações possíveis (Figura 7). Ele poderá efetuar um cadastro caso não tenha conta, poderá efetuar Login caso já possua cadastro, e prosseguir com a compra de um produto, realizando também cadastramento ou alterações no endereço de entrega (Manter Endereço). Durante a compra, a qualquer momento é possível visualizar os itens que estão sendo adquiridos (Ver Carrinho). O histórico de compras também será fornecido para esse ator (caso de uso Ver Itens Comprados).



**Figura 7. Diagrama de Casos de Uso. Fonte: o autor.**

## 4.2 Diagramas de Sequência

O Diagrama de Sequência disponível a seguir (Figura 8) descreve a principal atividade que será executada pelo usuário na aplicação, que é a compra de produtos. O processo de compra abrange todas as tarefas. Primeiramente, o usuário comprador irá escolher os produtos disponíveis, processo que requer o carregamento dos produtos do banco de dados para a tela do usuário. Cada produto selecionado é inserido no carrinho de compras. Após a finalização e confirmação de uma compra, o sistema requer um cadastro do usuário, caso o usuário não seja registrado. Na sequência, o usuário deve fazer um login, que no caso de sucesso, levará o usuário a um cadastro de endereços, para que seja informado o endereço de entrega. Após essa definição, é exibido o resumo da compra, cujas informações são salvas em definitivo, encerrando o processo.

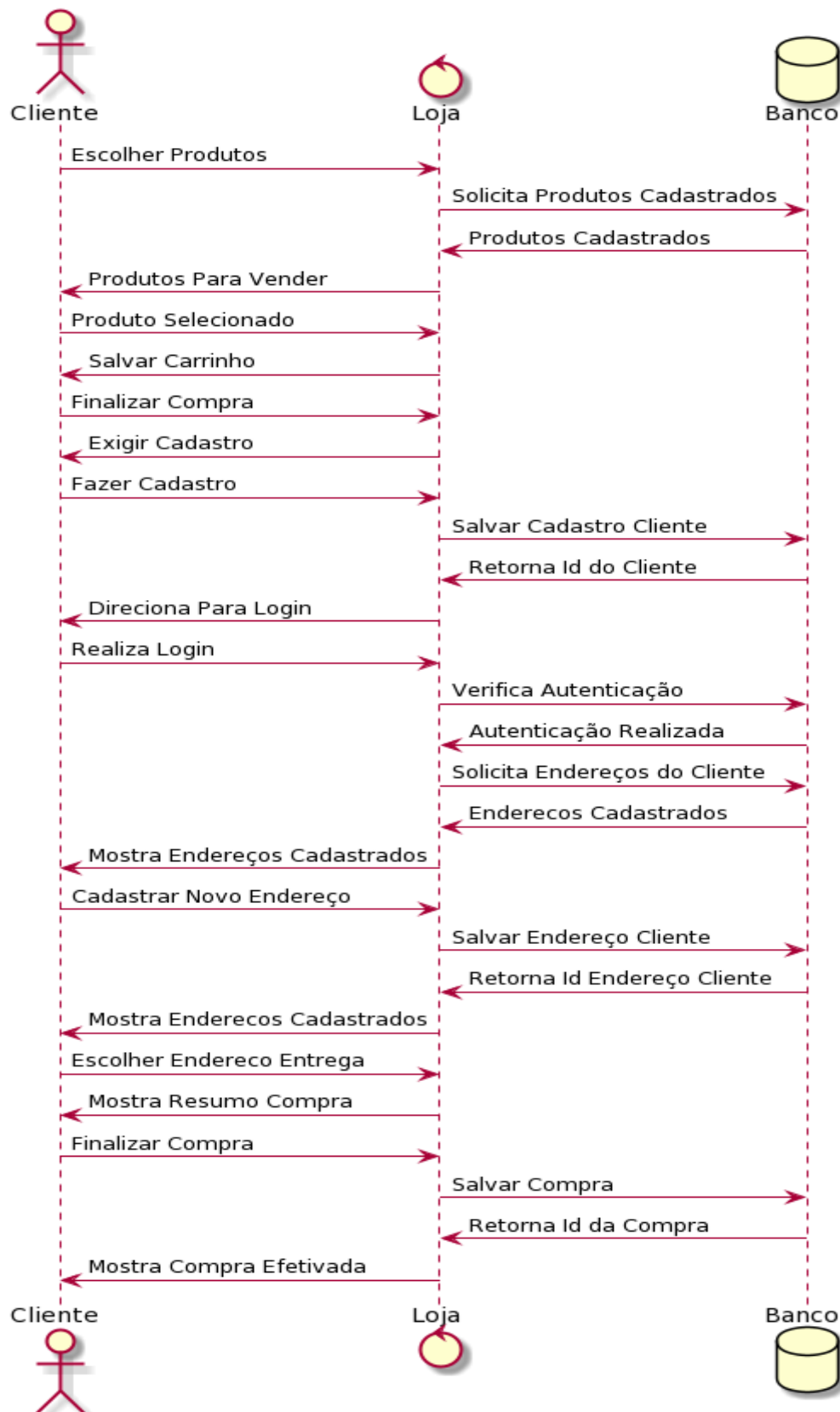


Figura 8. Diagrama de Sequência Processo de Compra. Fonte: o autor.

### 4.3 Diagrama de Classes

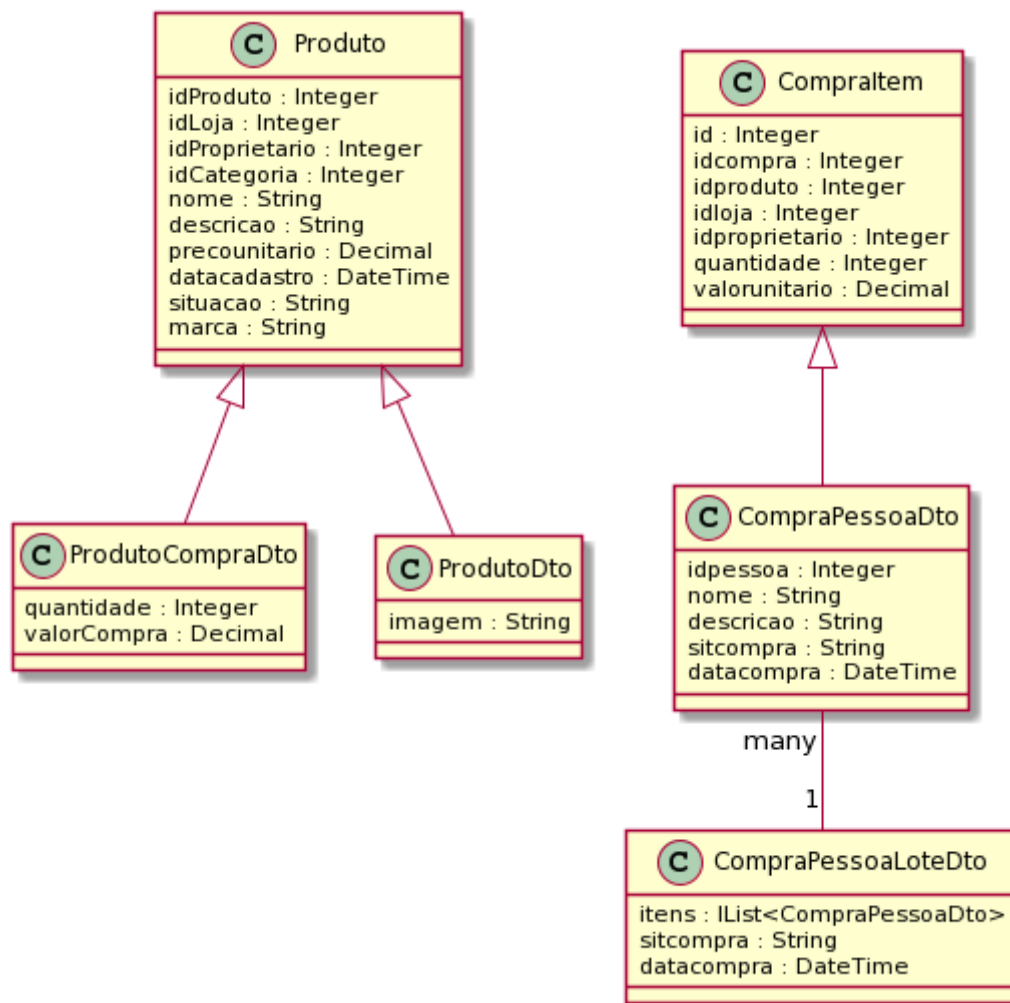
O Diagrama de Classes tem como objetivo permitir a visualização das classes que compõem o sistema, com seus respectivos atributos e métodos, bem como mostrar como as entidades se relacionam (GUEDES, 2018). O diagrama da aplicação web aqui proposta representa os modelos de dados que são manipulados na aplicação e são utilizados pelas regras de negócio do Sistema (Figura 9).

O atributo Dados representado na Figura 2 (Seção 1.4), do tipo ModeloDTO, é instanciado com um objeto de negócio, cujo nome é igual ao nome da classe de negócio acrescido do sufixo DTO (esse atributo conecta os diagramas das Figuras 9 e 2). A sigla DTO significa *Data Transfer Object*, e as classes DTO, em geral, possuem atributos iguais aos campos das respectivas tabelas às quais essas classes se referenciam (por exemplo: por meio da classe PessoaDTO se realiza o acesso à tabela Pessoa). Porém, algumas classes foram modificadas para atender as demandas da aplicação. As classes DTO que foram acrescentadas de informações são: ProdutoDTO, ProdutoCompraDTO, e CompraPessoaDTO. A classe CompraPessoaLoteDTO apenas agrega as informações da compra, obtendo informações das tabelas Compra e CompraItem.

O transporte de dados de um produto (ProdutoDTO) consiste das informações de um produto acrescido da imagem do produto (por meio de herança). De forma similar, as informações sobre um produto existente em uma compra (ProdutoCompraDTO) carregam, além das informações da compra, a quantidade do produto comprado e o valor total da compra daquele produto, em quantidade (ou seja, o valor unitário de cada produto multiplicado pela quantidade daquele produto que foi adquirida).

A classe DTO da compra da pessoa (CompraPessoaDTO) agrega as informações da classe “CompraItem” com as informações que identificam o produto (idproduto, idloja, idproprietario) que foi comprado por um usuário, com a sua quantidade e valor que foi praticado no momento da compra. Essa classe agrega também a data na qual ocorreu a compra (datacompra) e qual a situação do pedido (sitcompra), além de conter o nome e um identificador da pessoa (idpessoa) que efetuou o pagamento.

A classe “Produto” contém as informações referentes ao produto cadastrado para uma loja, contendo um identificador do produto (id), nome, descrição, preço unitário, situação e a data de cadastro. Além dessas informações, também são transportados campos que identificam as origens do produto (idloja e idproprietario). Um produto pode ter uma categoria, sendo o campo “idcategoria” aquele que contém a informação sobre a categoria do produto.



**Figura 9. Diagrama de Classes Representando as Entidades. Fonte: o autor.**

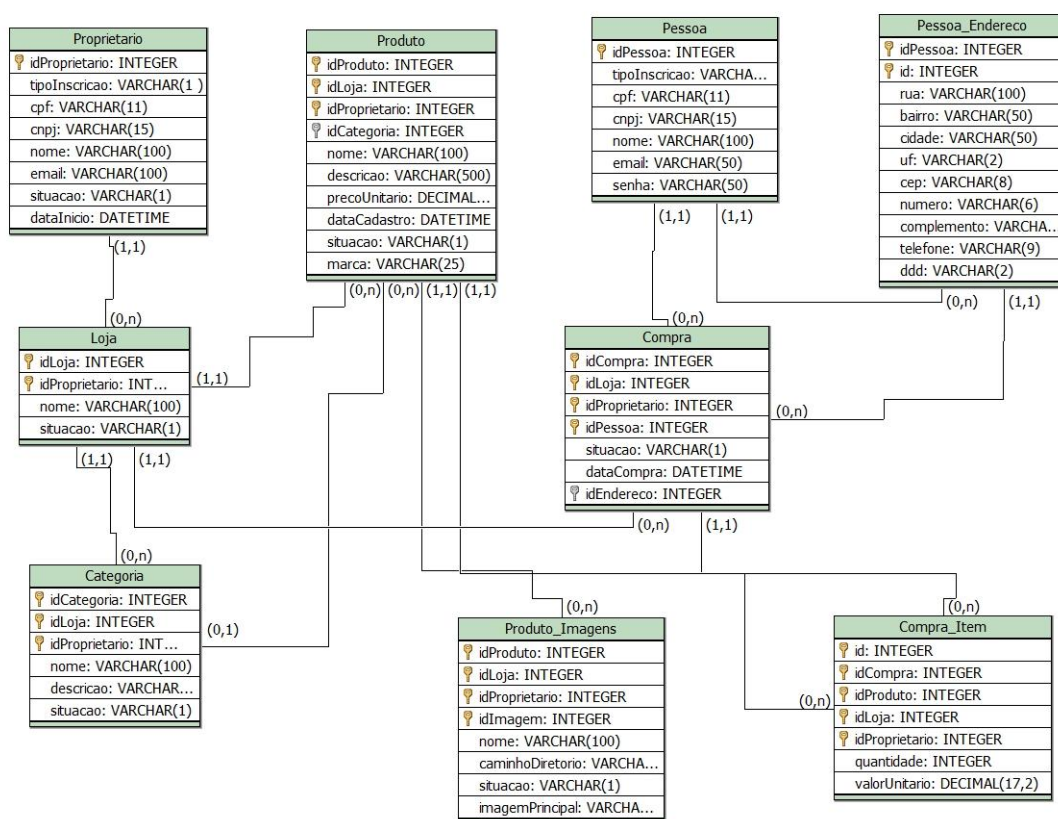
## 5 Modelagem de Dados

Para a modelagem de dados, primeiramente foi feito o levantamento dos requisitos para definir quais dados deveriam ser armazenados numa base de dados. A partir dessas informações, seguiu-se para a diagramação dos relacionamentos a serem implementados. Para a representação gráfica do modelo de banco foi utilizada a abordagem ER (Entidade Relacionamento). O modelo ER descreve os dados como entidades, relacionamentos e atributos (ELMASRI, 2005).

### 5.1 Diagrama de Entidades e Relacionamentos

O diagrama de entidades e relacionamentos da aplicação descreve como foi feita a parte de base de dados. Foram utilizadas nove (9) entidades que são armazenadas no

banco de dados, fornecendo todas as informações necessárias para que a loja consiga atender o cliente, entregando exatamente a compra que foi feita, no endereço informado. As entidades Proprietario e Pessoa possuem as informações que dão início a toda a relação com as demais entidades. Suas chaves estrangeiras se replicam por praticamente todas as tabelas no banco, chegando até a entidade Compra e Compra\_Item que compõem, juntas, a principal operação do sistema: a compra (Figura 10).



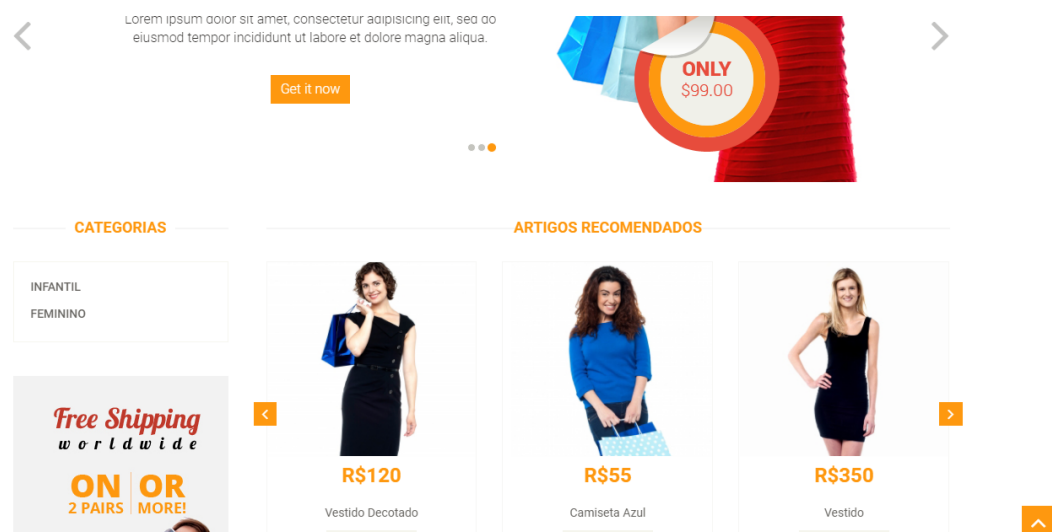
**Figura 10. Diagrama de Entidades e Relacionamentos. Fonte: o autor.**

## 6. Projeto da Interface

A parte de front-end foi desenvolvida tendo como base os sites de vendas online que existem atualmente disponíveis. O usuário primeiramente é apresentado para a tela na qual se encontram os produtos disponíveis para compra, e ao desejar finalizar a compra, as telas se encadeiam na ordem certa do processo. As páginas HTML do sistema foram obtidas a partir de um template adquirido do themehunt, que é um site que disponibiliza templates de páginas modernas que podem ser utilizados como base para desenvolvimento de aplicações *web* (THEMEHUNT, 2021).

A tela inicial da loja é também a página principal do sistema, que contém os itens disponíveis para a compra (Figura 11). No topo da página é exibido um *banner* promocional e na lateral da página existem categorias dos produtos (o filtro por categorias não está implementado no protótipo).





**Figura 11. Produtos para comprar. Fonte: o autor.**

Para cada produto listado, são exibidos também o respectivo preço e descrição. O usuário pode clicar no link “Adicionar” para acrescentar o produto que desejar na lista de compras (Figura 12).



**Figura 12. Botão de Adicionar. Fonte: o autor.**

Após executar a ação de adicionar o produto, ele é enviado para o carrinho, que pode ser consultado a qualquer momento. O link “carrinho” se encontra no canto superior direito, entre os links de Checkout e Entrar (Figura 13).



**Figura 13. Link “Carrinho” para consultar a lista de compras. Fonte: o autor.**

O carrinho de compras exibe os produtos que foram selecionados para aquisição, pelo usuário. No carrinho, ao lado de cada produto é exibido também o preço dele, a quantidade selecionada para a compra e o preço total por item (Figura 14). Caso queira remover o produto do carrinho, há um ícone que representa um “x” ao final do item, cuja ação remove o produto do fechamento final. Por fim, o botão Prosseguir Compra avança o processo, encerrando a seleção dos produtos a serem adquiridos.

Item	Preço	Quantidade	Total
 <div>Vestido Decotado Web ID: 1</div>	R\$ 120	+ 1 -	R\$ 120 
 <div>Camiseta Azul Web ID: 2</div>	R\$ 55	+ 1 -	R\$ 55 

O que você gostaria de fazer depois?

Escolha se você tem um código de desconto ou pontos de recompensa que deseja usar ou gostaria de estimar seu custo de entrega.

☐ Use meu Cupon de Desconto  
☐ Use meu Cupon de Presente  
☐ Estimar Custo de Expedição e Taxas

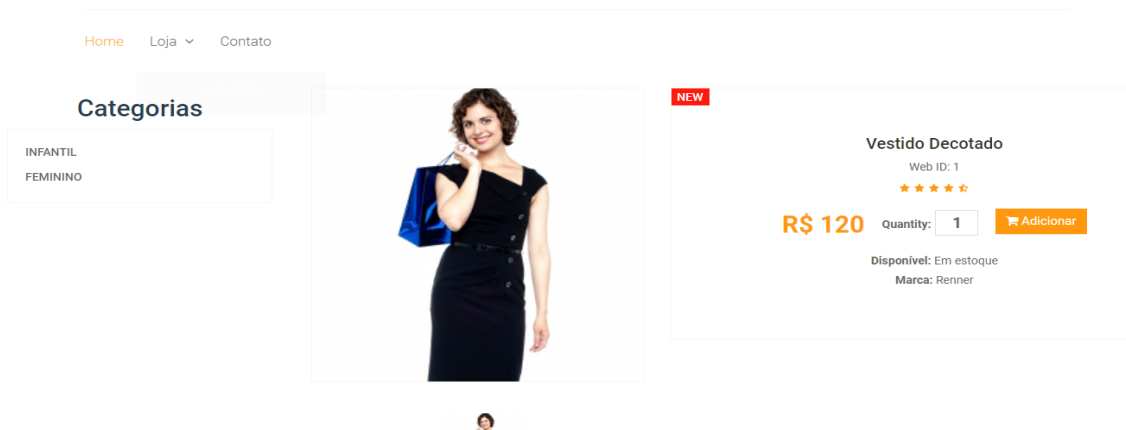
País:  Região / Estado:  CEP:

Sub Total Carrinho R\$ 175  
 Total R\$ 175

[Prosseguir Compra](#)

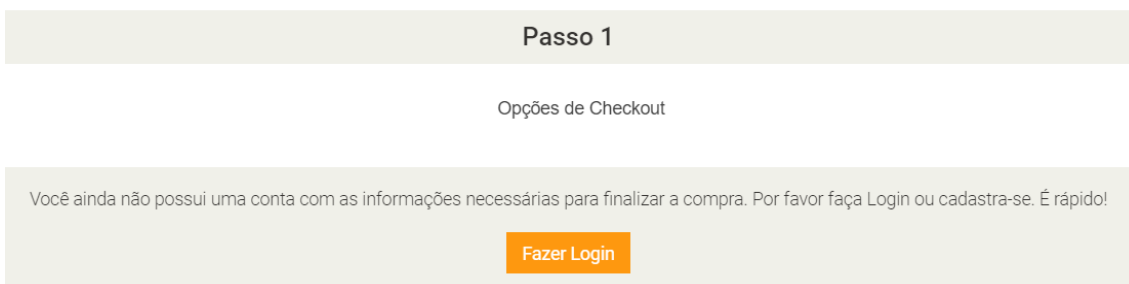
**Figura 14. Carrinho de compras. Fonte: o autor.**

Ainda na tela do carrinho de compras, é possível visualizar os detalhes dos produtos que estão sendo adquiridos, clicando-se sobre a descrição do produto. Nesse caso, uma tela de detalhes do produto será exibida (Figura 15).



**Figura 15. Detalhes do Produto. Fonte: o autor.**

A partir do carrinho de compras, caso o usuário ainda não esteja autenticado, a aplicação irá convidá-lo a se autenticar antes de prosseguir com o checkout. O usuário deverá clicar no botão “Fazer Login” para continuar o processo de compra (Figura 16).




**Figura 16. Usuário não autenticado deve fazer login. Fonte: o autor.**

O usuário pode se autenticar na tela informando seu e-mail que foi utilizado no momento do registro no site e sua senha privada (Figura 17).



**Figura 17. Login do usuário. Fonte: o autor.**

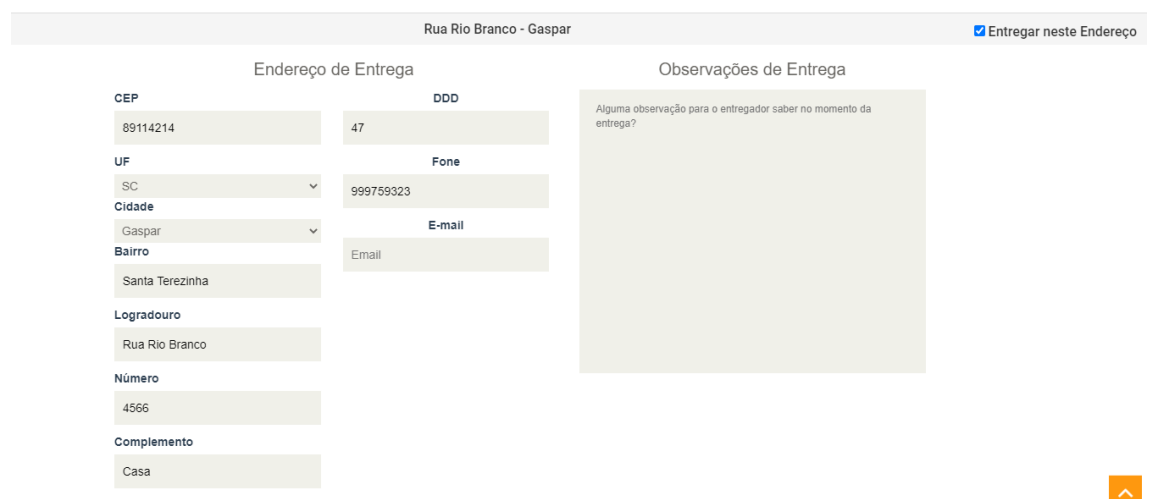
Caso o usuário não tenha cadastro ainda, a tela de login fornece o *link* de cadastro para que o usuário execute a ação de auto-cadastramento. É necessário informar nome, endereço de e-mail, número de CPF e uma senha de acesso (Figura 18).



Formulário de Cadastro de Usuário com o título "Suas Informações". O formulário contém campos para: Nome, E-mail, CPF, Senha e Confirme a Senha. Cada campo é precedido por um rótulo em negrito. Abaixo dos campos, há um botão laranja com o texto "Registrar".

**Figura 18. Tela de Cadastro de Usuário. Fonte: o autor.**

Após a confirmação das informações que se encontram no carrinho, o usuário prossegue com a compra e avança para a tela de checkout (Figura 19). Nessa tela a aplicação irá mostrar os endereços cadastrados deste usuário. A transição do carrinho para a tela de seleção de endereço ocorre apenas se o usuário estiver autenticado no sistema, ou seja, se o usuário efetuou um *login* na loja.



Tela de checkout com o título "Rua Rio Branco - Gaspar" e uma opção selecionada "Entregar neste Endereço". O formulário é dividido em duas seções principais: "Endereço de Entrega" e "Observações de Entrega".

**Endereço de Entrega:**

- CEP: 89114214
- DDD: 47
- UF: SC
- Fone: 999759323
- Cidade: Gaspar
- E-mail: Email
- Bairro: Santa Terezinha
- Logradouro: Rua Rio Branco
- Número: 4566
- Complemento: Casa

**Observações de Entrega:**

Alguma observação para o entregador saber no momento da entrega?

**Figura 19. Processo de *checkout*: selecionando endereço. Fonte: o autor.**

Na página de seleção de endereços é possível cadastrar um novo endereço: ao final da página, existe um formulário para o novo cadastramento (Figura 20). O botão “Salvar Endereço” acrescenta esse novo endereço à lista de endereços do usuário no sistema. Após a inclusão de um novo endereço, o fluxo do sistema retorna a essa mesma página, para que seja refeita a seleção do endereço de entrega.

### Endereço de Entrega

CEP

CEP\*

UF

Cidade

Bairro

Bairro\*

Logradouro

Logradouro\*

Número

Número\*

Complemento

Complemento

DDD

DDD\*

Fone

Fone\*

E-mail

Email

### Observações de Entrega

Alguma observação para o entregador saber no momento da entrega?

Salvar Endereço

**Figura 20. Cadastro de novo endereço de entrega. Fonte: o autor.**

O usuário também pode consultar as compras já realizadas: a loja fornece a tela de consulta de compras, na qual é apresentando um histórico de todos os pedidos já feitos pelo usuário por ordem cronológica (Figura 21). Também é possível ver os detalhes do produto comprado clicando no link “ver detalhes”, que irá redirecionar o usuário para tela de “Detalhe do Produto” (Figura 15, descrita anteriormente).

Checkout

Carrinho

Compras

jonas.stasiak@hotmail.com

Home

Loja

Contato

## Compras

Aguardando Pagamento dia 26 de junho

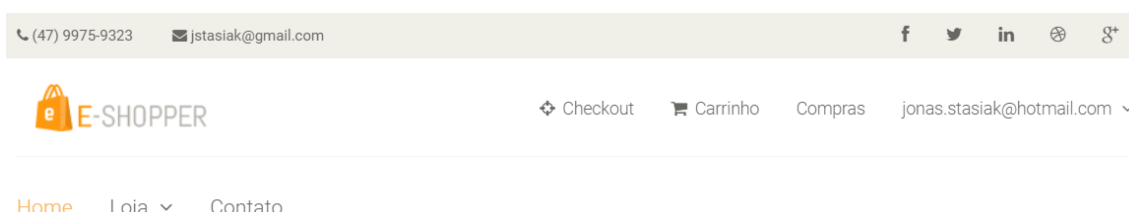
Vestido Decotado

R\$ 120 x 1 unidade

Ver detalhes

**Figura 21. Compras Realizadas. Fonte: o autor.**

Uma barra de navegação do site está localizada na parte superior da tela, sendo presente em todas as páginas do sistema. Caso o usuário esteja autenticado, é mostrado o e-mail do usuário autenticado no canto direito (Figura 22).



**Figura 22. Menu Principal. Fonte: o autor.**

## 7. Implementação

O Código fonte da aplicação pode ser encontrado em um repositório público disponível na Internet (<https://github.com/jsilvastasiak/E-shopping>), demonstrando a cadeia de *commits* que foram feitos durante o projeto. O desenvolvimento da aplicação iniciou com a elaboração da camada de dados da aplicação, que foi representado pelo Diagrama de Entidade Relacionamento. Com a camada de dados definida, foi feita a pesquisa das ferramentas e do padrão de projeto que seria usado para a construção da aplicação Web.

Após a escolha do Postgresql como banco de dados, foi pesquisado um Framework para se comunicar com o SGBD, e desenvolvida a infraestrutura do back-end, com Dapper e Asp .Net Core, finalizando a parte de dados da aplicação. No próximo passo, foram desenvolvidos os comandos lógicos da aplicação com a utilização da biblioteca *MediatR*. Foram também desenvolvidos os micro serviços disponibilizados pelo framework Asp .Net Core, finalizando assim a parte de Camada Lógica e Serviço Rest (aplicação).

Por fim, realizou-se o desenvolvimento do *front-end*, que simula uma loja virtual, consumindo a aplicação *back-end*, utilizando o framework *client-side* denominado Vue.js. As regras de layout foram implementadas no *front-end*.

## 8. Resultados Obtidos

Ter um comércio online é uma alternativa viável para vender produtos. O grande potencial de alcance de clientes pela internet valoriza esse tipo de negócio, que já fez grandes marcas do varejo adquirirem poderosas estruturas de TI para estabelecerem seus próprios e-commerces. O desenvolvimento de uma aplicação para lojas virtuais possibilita entrarmos aos poucos neste mercado que cresce diariamente. Esse tipo de sistema exige um forte conhecimento de modelagem de dados e a relação que se deve manter com o usuário final da aplicação, sendo necessário armazenar informações de

cunho pessoal do comprador, pois o serviço oferecido, no final de tudo, vai avançar para o mundo real dos negócios.

O objetivo proposto foi alcançado no desenvolvimento deste sistema: pode-se efetuar a simulação completa de uma compra, por meio das telas que representam a loja virtual, que consome a API de e-commerce desenvolvida. Algumas dificuldades foram encontradas, sendo a principal delas a criação da tela de checkout, que consiste de um processo passo-a-passo, no qual é preciso guiar o usuário até preencher todas as informações necessárias.

## 9. Considerações Finais e Trabalhos Futuros

É satisfatório desenvolver um projeto no qual se pode visualizar um futuro útil para o mesmo. O trabalho desenvolvido neste documento pode ser continuado, avançando para a área de BI (*Business Intelligence*), com extração de dados de vendas dos clientes, para fornecer informações relevantes que podem alavancar os negócios das lojas que estão consumindo a API. Também podem ser desenvolvidos relatórios de vendas, melhoria nas interfaces, uma nova interface *mobile*, dentre outros recursos.

Para dar seguimento ao desenvolvimento da aplicação, pode ser implementada uma integração com um sistema de pagamento real disponível, que irá fazer as transações financeiras das compras realizadas na loja. Ou a implementação do mesmo.

Outra possível continuação, será a criação do sistema de cadastro dos produtos, por parte do usuário administrador da loja, onde ele poderá manter suas mercadorias que estão disponíveis para a venda, utilizando uma *interface* simples de formulário para a utilização de um empresário, por exemplo.

## 10. Referências Bibliográficas

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Elsevier Brasil, 2006.

DAPPER. **Dapper Tutorial**. Disponível em: <https://dapper-tutorial.net/dapper>. Acesso em: 03 abr. 2021.

ELMASRI, Ramez et al. **Sistemas de banco de dados**. São Paulo: Pearson Addison Wesley, 2005.

GALINARI, Rangel et al. **Comércio eletrônico, tecnologias móveis e mídias sociais no Brasil**. BNDES Setorial, Rio de Janeiro, n. 41, p. [135]-180, 2015.

GUEDES, Gilleanes TA. **UML 2-Uma abordagem prática**. Novatec Editora, 2018.

KITAMURA, Celso. **ASP.NET Core – O que é?** 2018. Disponível em: <https://celsokitamura.com.br/asp-net-core/>. Acesso em: 03 abr. 2021.

LASTRES, H. M. M; ALBAGLI, S. **Comércio eletrônico e globalização: desafios para o Brasil**. Rio de Janeiro, RJ - Editora Campus. p. 84, 1999.

- LIMA, Lucas Galhardo; PETRUCELLI, Erick Eduardo; DO ESPÍRITO SANTO, Felipe. **Visão geral sobre o Gerenciamento de Estado no Vue.js com a biblioteca Vuex**. Revista Interface Tecnológica, v. 16, n. 1, p. 56-66, 2019.
- LOGI. **3-Tier Architecture: A Complete Overview**. 2021. Disponível em: <https://www.jinfony.com/resources/bi-dened/3-tier-architecture-complete-overview>. Acesso em: 03 abr. 2021.
- MACORATTI, José Carlos. **Apresentando o padrão de projeto Repository**. Disponível em: [http://www.macoratti.net/11/10/net\\_prl.htm#:~:text=O%20que%20%C3%A9%20o%20padr%C3%A3o,camada%20de%20neg%C3%B3cios%20\(BLL\)](http://www.macoratti.net/11/10/net_prl.htm#:~:text=O%20que%20%C3%A9%20o%20padr%C3%A3o,camada%20de%20neg%C3%B3cios%20(BLL)). Acesso em: 03 abr. 2021.
- PRANGE, Leo. **Design Pattern - Mediator**. 2019. Disponível em: <https://medium.com/qualyteam-engineering/design-pattern-mediator-6b4722b5a1ce>. Acesso em 03 abr. 2021.
- POSTGRESQL. **O que é o PostgreSQL?**. 2021. Disponível em: <http://pgdocptbr.sourceforge.net/pg82/intro-what-is.html>. Acesso em 03 abr. 2021.
- SILVEIRA, Thiago Mohr; LINEMBURGER, Filipe. **Desenvolvimento de ferramenta para o teste de requisições REST**. 2018. 126f. Monografia (Graduação em Sistemas de Informação) - Universidade Federal de Santa Catarina, Florianópolis, 2018.
- BOOTSTRAP, **The most popular HTML, CSS, and JS library in the world**. Página Inicial. Disponível em: <https://getbootstrap.com/>. Acesso em 03 abr. 2021.
- THEMEHUNT, **Free Bootstrap Templates & Website Themes - ThemeHunt**. Página Inicial. Disponível em: <https://themehunt.com/>. Acesso em 03 abr. 2021.
- MACORATTI, José Carlos. **Definindo um infra-estrutura baseada em camadas**. Disponível em: [http://www.macoratti.net/09/01/net\\_arq1.htm](http://www.macoratti.net/09/01/net_arq1.htm). Acesso em: 20 jun. 2021.
- MERCADOLIVRE, **Mercado Livre Brasil**. Disponível em: <https://www.mercadolivre.com.br>. Acesso em: 04 abr. 2021.
- SUBMARINO, **Submarino - sua história começa aqui**. Disponível em: <https://www.submarino.com.br/>. Acesso em: 04 abr. 2021.
- MAGAZINELUIZA, **Magazine Luiza | Pra você é Magalu!**. Disponível em: <https://www.magazineluiza.com.br/>. Acesso em: 04 abr. 2021.