**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**ETEC DA ZONA LESTE**

**Novotec Desenvolvimento de Sistemas AMS**

**João Pedro Silva de Oliveira**

**Luisa Santos Silva**

**Vinicius Valero Chabariberi**

**NAVIGATE BUY: Sistema Web Agregador de Compras Online Seguras e Simplificadas**

**São Paulo**

**2024**

**João Pedro Silva de Oliveira**

**Luisa Santos Silva**

**Vinicius Valero Chabariberi**

**NAVIGATE BUY: Sistema Web Agregador de Compras Online Seguras e Simplificadas**

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas da Etec Zona Leste, orientado pelo Prof. Jeferson Roberto de Lima, como requisito parcial para obtenção do título de técnico em Desenvolvimento de Sistemas.

**São Paulo**

**2024**

**RESUMO**

Compra online é uma tarefa muito comum nos dias de hoje, e os perigos e riscos que um consumidor pode encontrar são diversos, causando uma preocupação ainda maior devido ao crescimento de consumidores online nos últimos anos. Com o objetivo de tornar a experiência em compras online uma tarefa mais rápida e segura, será desenvolvido um sistema web que irá utilizar as técnicas web crawler e web scraping. Para isso, será adotada uma abordagem exploratória qualitativa, com uma implementação que agrega diversos produtos de lojas renomadas, permitindo ao consumidor tomar decisões com base em comentários de sites de reclamação e gráficos. Além disso, a compra online se torna algo demorado quando se faz pesquisas para comparar preços de um produto em específico. Espera-se que o sistema web possa direcionar os consumidores para sites seguros e recomendados, facilitando a busca pelos menores preços entre variados sites, encontrando tudo em um só lugar. Portanto, pretende-se contribuir para avanço do comércio online oferecendo maior simplicidade e segurança para a principal peça desta engrenagem, o consumidor.

**Palavras-chave**: Compra online. Consumidor. Sistema Web. Simplicidade. Segurança.

**ABSTRACT**

Online purchasing is a very common task nowadays, and the dangers and risks that a consumer can find are diverse, causing even greater concern due to the increase in online consumers in the last few years. With the aim of this work being to make online purchasing a fast and secure task, it will be necessary to develop a web system that will use web crawlers and web scraping techniques. For that, an exploratory qualitative approach will be adopted, implementing an aggregation of different products from renowned shops, allowing consumers to make decisions based on comments from complaint websites and charts. Furthermore, online purchasing becomes time-consuming when searching to compare prices of a specified product. As a result, we expect that the web system can direct consumers to secure and recommend sites, making it easier to search for the lowest prices among different sites, finding everything in one place. Therefore, the pretend is to contribute to the advancement of online purchasing by offering greater simplicity and security to the principal part of this gear, the consumer.

**Keywords**: Online purchasing. Consumer. Web System. Simplicity. Security.

**LISTA DE ILUSTRAÇÕES**

[Figura 1 - Exemplo de código em HTML para a criação de uma avaliação simples 14](#_Toc181346305)

[Figura 2 - Resultado do código HTML 16](#_Toc181346306)

[Figura 3 - Exemplo de código em CSS 17](#_Toc181346307)

[Figura 4 - Conexão entre CSS e HTML 18](#_Toc181346308)

[Figura 5 - Exemplo de código em CSS para a criação de uma estilização 18](#_Toc181346309)

[Figura 6 - Resultado do código CSS 19](#_Toc181346310)

[Figura 7 - Código construído com a linguagem JavaScript 20](#_Toc181346311)

[Figura 8 - Resultado do código com HTML, CSS e JavaScript 22](#_Toc181346312)

[Figura 9 – Resultado de caixa de alerta com JavaScript 22](#_Toc181346313)

[Figura 10 - Exemplo de código para aplicação de um login simples em React 24](#_Toc181346314)

[Figura 11 - Resultado da aplicação de um login simples feito em React 25](#_Toc181346315)

[Figura 12 - Exemplo de código para aplicação de cálculo de média em Python 26](#_Toc181346316)

[Figura 13 - Resultado do código do cálculo da média em Python pelo terminal 27](#_Toc181346317)

[Figura 14 - Exemplo de inicialização de um projeto NPM 28](#_Toc181346318)

[Figura 15 - Exemplo de inserção dos dados exigidos pelo NPM 29](#_Toc181346319)

[Figura 16 - Exemplo de instalação do Tailwind 31](#_Toc181346320)

[Figura 17 - Exemplo de criação da conexão entre HTML e o Tailwind 32](#_Toc181346321)

[Figura 18 - Acrescentando diretivos do Tailwind 32](#_Toc181346322)

[Figura 19 – Exemplo executando comando Tailwind 33](#_Toc181346323)

[Figura 20 - Código HTML com a implementação do Tailwind 33](#_Toc181346324)

[Figura 21 - Resultado do código HTML com a implementação do Tailwind 35](#_Toc181346325)

[Figura 22 - Exemplo de código para aplicação de cálculo de média com o Flask 36](#_Toc181346326)

[Figura 23 - Tela de inserimento dos dados em HTML 37](#_Toc181346327)

[Figura 24 - Resultado da codificação com o Flask no terminal 37](#_Toc181346328)

[Figura 25 - Exemplo de codificação em Python com o Scrapy 38](#_Toc181346329)

[Figura 26 - Resultado da codificação com o Scrapy 39](#_Toc181346330)

[Figura 27 - Exemplo de código com a AwesomeAPI 40](#_Toc181346331)

[Figura 28 - Resultado da aplicação com a AwesomeAPI 41](#_Toc181346332)

[Figura 29 - Exemplo da criação de um DER 42](#_Toc181346333)

[Figura 30 - Exemplo da criação de um MER 43](#_Toc181346334)

[Figura 31 - Exemplo de um código de banco de dados MySQL 44](#_Toc181346335)

[Figura 32 - Resultado de uma consulta de uma tabela MySQL 45](#_Toc181346336)

[Figura 33 - Exemplo de Wireframe de baixa fidelidade. 46](#_Toc181346337)

[Figura 34 - Exemplo de Wireframe de média fidelidade. 46](#_Toc181346338)

[Figura 35 - Exemplo de Wireframe de alta fidelidade. 47](#_Toc181346339)

[Figura 36 - Exemplo de Diagrama de Casos de Uso 48](#_Toc181346340)

[Figura 37 - Exemplo de documentação de Caso de Uso 49](#_Toc181346341)

[Figura 38 - Exemplo de Diagrama de Sequência 50](#_Toc181346342)

[Figura 39 - Exemplo de Diagrama de Atividade 51](#_Toc181346343)

[Figura 40 - Exemplo de Diagrama Máquina de Estados 52](#_Toc181346344)

[Figura 41 - Diagrama de Casos de uso do sistema web 55](#_Toc181346345)

[Figura 42 - Diagrama de sequência "Cadastrar Consumidor" 64](#_Toc181346346)

[Figura 43 - Diagrama de sequência "Fazer login" 65](#_Toc181346347)

[Figura 44 - Diagrama de sequência "Consultar Consumidor" 66](#_Toc181346348)

[Figura 45 - Diagrama de sequência "Editar Consumidor” 67](#_Toc181346349)

[Figura 46 - Diagrama de sequência "Buscar produto” 68](#_Toc181346350)

[Figura 47 - Diagrama de sequência "Buscar comentários de avaliação” 69](#_Toc181346351)

[Figura 48 - Diagrama de sequência "Favoritar produto” 70](#_Toc181346352)

[Figura 49 - Diagrama de sequência "Consultar produtos favoritados” 71](#_Toc181346353)

[Figura 50 - Diagrama de sequência "Editar produtos favoritados” 71](#_Toc181346354)

[Figura 51 - Diagrama de sequência "Comparar preços de produtos internacionais” 72](#_Toc181346355)

[Figura 52 - Diagrama de Atividade "Cadastrar Consumidor" 73](#_Toc181346356)

[Figura 53 - Diagrama de Atividade "Fazer login" 74](#_Toc181346357)

[Figura 54 - Diagrama de Atividade "Consultar Consumidor" 75](#_Toc181346358)

[Figura 55 - Diagrama de Atividade "Editar Consumidor" 75](#_Toc181346359)

[Figura 56 - Diagrama de Atividade "Buscar produto" 76](#_Toc181346360)

[Figura 57 - Diagrama de Atividade "Buscar comentários de avaliação" 76](#_Toc181346361)

[Figura 58 - Diagrama de Atividade "Favoritar produto" 77](#_Toc181346362)

[Figura 59 - Diagrama de Atividade "Consultar produtos favoritados" 78](#_Toc181346363)

[Figura 60 - Diagrama de Atividade "Editar produtos favoritados" 78](#_Toc181346364)

[Figura 61 - Diagrama de Atividade "Comparar preços de produtos internacionais" 79](#_Toc181346365)

[Figura 62 - Wireframe de baixa fidelidade da interface “Cadastro” 80](#_Toc181346366)

[Figura 63 – Wireframe de alta fidelidade da interface "Cadastro" 80](#_Toc181346367)

[Figura 64 - Wireframe de baixa fidelidade da interface “Login” 81](#_Toc181346368)

[Figura 65 - Wireframe de alta fidelidade da interface "Login" 81](#_Toc181346369)

[Figura 66 - Wireframe de baixa fidelidade da interface "Perfil" 82](#_Toc181346370)

[Figura 67 - Wireframe de alta fidelidade da interface "Perfil" 83](#_Toc181346371)

[Figura 68 - Wireframe de baixa fidelidade da interface "Home" 84](#_Toc181346372)

[Figura 69 - Wireframe de alta fidelidade da interface "Home" 85](#_Toc181346373)

[Figura 70 - Wireframe de baixa fidelidade da interface "Buscar produto" 86](#_Toc181346374)

[Figura 71 - Wireframe de alta fidelidade da interface "Buscar produto" 87](#_Toc181346375)

[Figura 72 - Wireframe de baixa fidelidade da interface "Produtos favoritados" 88](#_Toc181346376)

[Figura 73 - Wireframe de alta fidelidade da interface "Produtos favoritados" 89](#_Toc181346377)

[Figura 74 - Wireframe de baixa fidelidade da interface "Buscar Comentários de avaliação" 90](#_Toc181346378)

[Figura 75 - Wireframe de alta fidelidade da interface "Buscar Comentários de avaliação" 91](#_Toc181346379)

[Figura 76 - Wireframe de baixa fidelidade da interface "Comparar preços de produtos internacionais" 92](#_Toc181346380)

[Figura 77 - Wireframe de alta fidelidade da interface "Comparar preços de produtos internacionais" 93](#_Toc181346381)

**LISTA DE ABREVIATURAS E SIGLAS**

Application Program Interface (API)

Cascading Style Sheet (CSS)

Diagrama Entidade Relacionamento (DER)

Hypertext Markup Language (HTML)

Hypertext Transfer Protocol (HTTP)

JavaScript (JS)

Modelo Entidade Relacionamento (MER)

Node Package Manager (NPM)

Sistema Gerenciador de Banco de Dados (SGBD)

Unified Modeling Language (UML)

Uniform Resource Locator (URL)

User Experience (UX)

SUMÁRIO

[**1. INTRODUÇÃO 11**](#_Toc181346383)

[**2. REFERENCIAL TEÓRICO 13**](#_Toc181346384)

[**2.1. Dificuldade na pesquisa de compras online seguras e simplificadas 13**](#_Toc181346385)

[**2.2. HTML 13**](#_Toc181346386)

[**2.3. CSS 17**](#_Toc181346387)

[**2.4. JavaScript 20**](#_Toc181346388)

[**2.5. TypeScript 22**](#_Toc181346389)

[**2.6. React 23**](#_Toc181346390)

[**2.7. Python 25**](#_Toc181346391)

[**2.8. NPM 27**](#_Toc181346392)

[**2.9. JSON 30**](#_Toc181346393)

[**2.10. Web Crawler 30**](#_Toc181346394)

[**2.11. Web Scraping 30**](#_Toc181346395)

[**2.12. Framework 31**](#_Toc181346396)

[**2.12.1. Tailwind 31**](#_Toc181346397)

[**2.12.2. NextJS 35**](#_Toc181346398)

[**2.12.3. Flask 35**](#_Toc181346399)

[**2.12.4. Scrapy 38**](#_Toc181346400)

[**2.13. APIs 40**](#_Toc181346401)

[**2.13.1. AwesomeAPI 40**](#_Toc181346402)

[**2.14. Banco de Dados 41**](#_Toc181346403)

[**2.14.1. Modelagem de Dados 42**](#_Toc181346404)

[**2.14.2. MySQL 43**](#_Toc181346405)

[**2.15. UX 45**](#_Toc181346406)

[**2.16. Wireframe 45**](#_Toc181346407)

[**2.17. UML 47**](#_Toc181346408)

[**3. DESENVOLVIMENTO 53**](#_Toc181346409)

[**3.1. Diagrama de Casos de Uso 53**](#_Toc181346410)

[**3.1.1. Documentações de Caso de Uso 55**](#_Toc181346411)

[**3.2. Diagramas de Sequência 63**](#_Toc181346412)

[**3.3. Diagramas de Atividade 73**](#_Toc181346413)

[**3.4. Diagramas de Máquina de Estados 79**](#_Toc181346414)

[**3.5. Prototipação das páginas da aplicação 79**](#_Toc181346415)

[**4. CONSIDERAÇÕES FINAIS 94**](#_Toc181346416)

[**REFERÊNCIAS 95**](#_Toc181346417)

# INTRODUÇÃO

O Navigate Buy foca no desenvolvimento de um sistema web que busque assegurar e simplificar a experiência do consumidor nas compras online. Conforme o site E-Commerce Brasil (2024), O e-commerce está projetado para crescer de R$ 349 bilhões para R$ 557 bilhões no intervalo entre 2023 e 2027, evidenciando a transição dos padrões de consumo. Com esse aumento, a necessidade de ferramentas que tornem as compras online mais seguras e simplificadas tornou-se crucial.

Apesar da facilidade e acessibilidade das compras online, os consumidores enfrentam dificuldades com a análise de melhores ofertas e a verificação da segurança dos sites de compra. Segundo Bolzani (2022), um levantamento feito em conjunto com a Opinion Box pela Octadesk mostrou que 73% dos consumidores declaram que os preços na internet são mais acessíveis. Porém, de acordo com Faustino e Lobato (2023), uma verificação feita pelo Fórum Brasileiro de Segurança Pública (FBSP) apresenta que, em 2022, houve uma elevação de 65,2% de fraudes em âmbito online.

Diante dessa problemática, a hipótese central é a implementação de um sistema web agregador de compras online que facilite a tomada de decisões dos consumidores, integrando a análise de lojas e seus respectivos produtos, apresentação de diferentes preços e comentários de avaliação para uma melhor comparação, conversão de moedas em tempo real e uma seguridade mais eficiente em suas compras.

Logo, o objetivo geral é desenvolver um sistema web que permita que os consumidores comparem produtos e preços entre diferentes lojas verificadas, buscando uma maior segurança em suas compras, analisando as avaliações vigentes de diversos produtos com base em conteúdos de sites de reclamação, sendo possível ver conversões de moedas em tempo real para maior acessibilidade. Dentre os objetivos específicos, estão inclusos:

* Aprofundar as pesquisas e estudos sobre as dificuldades dos consumidores em seus processos de compra online;
* Analisar os principais requisitos de um sistema web que vise facilitar as dificuldades enfrentadas pelos compradores;
* Fornecer conversão de moedas em tempo real, examinando como isso facilita a transparência nos preços e reduz a incerteza do consumidor;
* Implementar uma eficiente aplicação web, fornecendo informações relevantes, ajudando a minimizar as adversidades enfrentadas pelos compradores;
* Explorar o impacto da comparação de preços e comentários de avaliação na tomada de decisões dos consumidores.

Em vista disso, a metodologia adotada incluiu pesquisas exploratórias qualitativas para investigar as dificuldades dos consumidores e o impacto de um sistema web agregador de compras online seguras e simplificadas. Dados estatísticos foram coletados e analisados para o desenvolvimento que atendesse às necessidades identificadas, proporcionando uma ferramenta eficaz para a tomada de decisões nas compras online.

Nos capítulos sucessores, serão apresentadas todas as etapas de desenvolvimento e fundamentação teórica do Navigate Buy, sustentada por diversos autores relevantes na área, como Guedes (2018), que fornecerá descrições sobre a documentação e a modelagem do sistema; Milani (2006), abordará sobre a gestão do banco de dados; Mckinney (2018), explorará a linguagem de programação Python que terá papel central no desenvolvimento do projeto e Mitchell (2019), que aborda a definição sobre o Web crawler, uma das principais tecnologias da aplicação.

# REFERENCIAL TEÓRICO

Nesta seção, designa-se o problema de pesquisa que deu origem à necessidade do sistema web, assim como as tecnologias adotadas para seu desenvolvimento, proporcionando, portanto, o embasamento teórico essencial para a concepção do Navigate Buy.

# Dificuldade na pesquisa de compras online seguras e simplificadas

Segundo Sé (2023), um estudo realizado pela consultoria Offerwise em janeiro de 2023 mostrou que em cada dez brasileiros, nove pesquisam online antes de fazer uma compra. Isso reflete um aumento na adoção do ambiente online, com indivíduos dedicando tempo às pesquisas para encontrar o que desejam.

Diante disso, conforme abordado por Dias e Hemais (2015), apesar da crescente inclusão de pessoas no ambiente virtual, ainda existem receios relacionados a questões de segurança e privacidade. Portanto, o foco principal é atender às necessidades do consumidor e reduzir suas adversidades nas compras online, proporcionando mais facilidade na busca por ofertas e maior segurança.

# HTML

Segundo Cardoso (1999), Hypertext Markup Language (HTML), é uma linguagem baseada em arquivos-texto, onde pode ser editada por meio de um programa.

Conforme Freeman e Freeman (2006), um navegador tem a função de solicitar uma página HTML a um servidor, recebendo-a e exibindo-a em uma janela. O HTML diz ao navegador tudo sobre o conteúdo e a estrutura da página, constituída por tags.

Para Miletto e Bertagnolli (2014), a função das *tags* é representar os elementos que compõem uma página envolta por sinais (<) e (>). Essas *tags* podem aparecer em pares, porém elas sempre têm um começo e um fim determinado por (/).

Figura 1 - Exemplo de código em HTML para a criação de uma avaliação simples

**

Fonte: Do próprio autor, 2024.

O código exibido acima na Figura 1 resultará em uma interface de uma avaliação simples em HTML que será representada na próxima figura.

As principais *tags* utilizadas neste exemplo foram:

* <!DOCTYPE html>: é obrigatória do corpo do HTML, ele mostra ao navegador, qual é o tipo de linguagem que está sendo utilizada;
* <html>: declara o início e o fim para que o navegador funcione da melhor maneira;
* *Lang*: é um atributo que define o idioma de um elemento para os desenvolvedores saberem;
* <*head*>: armazena todas as informações do arquivo, como título e rotas para outros arquivos;
* <*meta*>: tem a função de fornecer informações para o navegador a categorizar uma página, além de adaptar a página a diferentes tipos de tela automaticamente;
* *Charset* UTF-8: este atributo consiste em indicar qual é o formato de codificação para os navegadores, neste caso, o UTF-8;
* *Name*: o atributo “*name*” tem a função de identificar um elemento por um nome;
* *Viewport* e *Content*: os dois atributos tem a função de adaptar a página *web* feita para computador, às diferentes proporções de tela, como as de um celular;
* <*title*>: indica o que será exibido na aba do navegador da página;
* <*body*>: é fundamental para a estrutura do HTML, ela determina onde está localizado tudo visível para o usuário;
* <div>: as diversas divisões servem para dividir o conteúdo que está presente na página. No exemplo, ela está criando um container que abrange todo o corpo do HTML e ele também cria uma divisão para o cabeçalho;
* *Class*: este atributo tem a função de definir a qual classe aquela *tag* específica pertence, podendo ser reutilizada várias vezes;
* *id*: o atributo identificador determina uma referência, ele pode ser utilizado apenas uma vez para cada *tag;*
* Títulos e subtítulos: os títulos e subtítulos são representados pelas *tags* “<h1>”, “<h2>”, “<h3>”, “<h4>”, “<h5>”, “<h6>”, sendo o h1 a maior fonte e o h6, a menor;
* <p>: agrupa qualquer tipo de meio visual, como texto e imagens;
* <*form*>: representa uma seção do documento onde existem controles e ações que o usuário pode tomar para interagir com o formulário, neste caso a avaliação;
* <*label*>: cria um pequeno texto para indicar a categoria de um campo de texto, geralmente;
* *For*: o atributo “for” serve para referenciar à *tag* <*input*> qual é a identificação daquela *tag* <*label*> específica;
* <*input*>: os campos de texto representam espaços a serem preenchidos na interface da página, ela serve para criar os controles interativos dos formulários. Além disso, eles são categorizados através do seu tipo (*type*);
* Tipo: o atributo “type” determina de qual tipo aquela *tag “*<*input*>” pertence, podendo variar entre os tipos email, nome, senha, entre outros. Quando o tipo é definido, a *tag* consegue modificar sua formatação;
* <br>:  é utilizado para dividir os conteúdos, como textos, de uma linha para outra;
* <*textarea*>: tem a função de criar na interface da página uma caixa de texto similar ao campo de texto, porém com a finalidade de o usuário escrever um texto corrido;
* *Cols* e *Rows*: estes dois atributos definem a quantidade de colunas e linhas da *tag “*<*textarea*>”;
* <*button*>: é utilizado para criar um botão na interface da página, este botão é capaz de receber os dados digitados pelo usuário dentro da avaliação.

Figura 2 - Resultado do código HTML

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# CSS

De acordo com Quierelli (2012), o Cascading Style Sheet (CSS) edita a visualização do conteúdo de páginas web, ele adiciona cores ao fundo e estiliza textos, além de formatar imagens.

Conforme mencionado por Jobstraibizer (2009), o CSS é empregado para determinar a formatação de documentos elaborados em uma linguagem de marcação, semelhante ao HTML.

Dessa forma, Silva (2011) menciona sobre a regra CSS, disposta pelo seletor e sua declaração, e essa é formada por uma propriedade e um valor. Uma regra CSS pode conter várias declarações separadas por ponto e vírgula.

Figura 3 - Exemplo de código em CSS

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Conforme o exemplo da Figura 3, “*button*” é o elemento ou seletor referenciado e que irá receber a estilização, “*color*” é a propriedade que define a característica do seletor e, no exemplo, tem a função de definir uma cor, e “*black*” é o valor da propriedade, que neste caso é especificado como preto.

Como descreve Eis e Ferreira (2012), o *link* em HTML é um elemento que consegue importar objetos externos de um documento para outro, assim como no exemplo em que ele referencia um arquivo CSS na codificação da linguagem HTML.

Figura 4 - Conexão entre CSS e HTML

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Neste pequeno trecho da Figura 4 da *tag* “<*link*>”, deve ser repassada a rota do arquivo onde está armazenado o CSS e consequentemente a ligação entre os dois arquivos estará feita.

Figura 5 - Exemplo de código em CSS para a criação de uma estilização

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

O código exibido acima na Figura 5 resultará em uma interface de uma avaliação simples em HTML e CSS que será representado na próxima figura.

As propriedades utilizadas neste exemplo foram:

* *Font-family*: define a fonte que vai ser utilizada nos textos;
* *Margin*: altera a margem ao redor do elemento;
* *Padding*: introduz o espaço entre o conteúdo da página e as bordas;
* *Display*: determina como os elementos vão ser distribuídos na página;
* *Justify-content*: justifica que todos os itens devem ser distribuídos igualmente;
* *Align-items*: alinha os elementos verticalmente;
* *Background-color*: define a cor que vai ser utilizada no fundo dos elementos;
* *Color*: altera a cor dos textos daquele elemento;
* *Border-Radius*: serve para arredondar as bordas do elemento;
* *Box-shadow*: cria uma sombra ao redor do elemento;
* *Width*: determina o tamanho do elemento na página;
* *Text-Align*: alinha verticalmente os textos deste elemento.

Figura 6 - Resultado do código CSS

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# JavaScript

Conforme descreve Flanagan (2013), a linguagem de programação JavaScript (JS) é bastante utilizada, sendo essencial para a interatividade de elementos, possuindo uma vasta biblioteca de utilitários.

Segundo Stefanov (2011), sendo essencial para a interatividade de elementos, o JS opera entre o lado do cliente, a qual é a parte visível na página, e o lado do servidor, responsável pelo processamento de informações de um sistema.

Destacado por Groner (2019), o *GitHub*, o maior host de códigos do mundo, hospeda mais de 400 mil repositórios em JavaScript, podendo assim notar uma das linguagens mais populares e proeminentes do mundo e da internet.

Figura 7 - Código construído com a linguagem JavaScript

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

A programação exibida na Figura 7 resultará em uma interface de uma avaliação simples em HTML e CSS, porém a implementação do JavaScript fornece funcionalidade, fazendo com que um alerta apareça assim que o botão “Enviar” seja pressionado, confira nas duas próximas figuras.

As principais partes deste exemplo são:

* Função inserirDados(): esta linha cria uma função, chamada de “inserirDados”, capaz de receber dados da avaliação e inseri-los em uma mensagem de alerta. Ela abrange a maior parte da codificação;
* *Const*: é uma declaração que define uma variável com um nome específico no espaço de memória do dispositivo;
* *document.getElementById*: é autoexplicativa pelo seu nome, ela consegue buscar um conteúdo de um elemento HTML através de um identificador;
* *AddEventListener*: adiciona um manipulador de eventos para a avaliação e recebe um objeto como parâmetro, ele é utilizado todas as vezes que o botão “enviar” for pressionado;
* *event.preventDefault*(): serve para evitar o envio de um formulário padrão do navegador, o que iria conflitar com o exemplo do código quando a página fosse recarregada;
* .*value*: a finalidade desta propriedade é obter o valor do elemento HTML e armazenar o conteúdo em uma variável constante;
* *Alert*: antes de ser acionado, cria-se uma linha *string* para formatar as informações que estiverem presentes na última variável constante da codificação. Esta formatação será exibida em um alerta na página.

Figura 8 - Resultado do código com HTML, CSS e JavaScript

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Assim que o botão “Enviar” é pressionado, o seguinte alerta se apresenta na tela:

Figura 9 – Resultado de caixa de alerta com JavaScript

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# TypeScript

Parafraseando Adriano (2021), o TypeScript é um pré-processador de códigos escritos em JavaScript que é mantido e desenvolvido pela empresa Microsoft. Ele utiliza de princípios parecidos com os de JavaScript, porém a sua vantagem se concentra em prevenir erros.

Em concordância com Goldberg (2022), o TypeScript conta com um verificador de tipos que examina o código, compreende sua lógica e aponta possíveis falhas na estrutura do código.

Conforme dito por Souza, Lima e Caridade (2022), o TypeScript pode ser utilizado no desenvolvimento *back-end* de um projeto, mas sendo muito usado em frameworks *front-end* isso faz com que aumente a produtividade dos desenvolvedores *FullStack*.

# React

Em concordância com Soares (2023), o React é uma biblioteca herdada do JavaScript, tendo como propósito principal o desenvolvimento *Front-End*, trazendo consigo uma diminuição nas linhas de código em aplicações.

Conforme dito por Silva (2021), a biblioteca React possui uma facilidade e otimização no desenvolvimento de criação de interfaces interativas para o usuário e de alto desempenho.

Segundo Stefanov (2016), não há imposição de uma estrutura de diretórios fixa ao utilizar o React, possuindo a liberdade de utilizar ou renomear um diretório novo em seu desenvolvimento da aplicação em React.

Figura 10 - Exemplo de código para aplicação de um login simples em React

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Tendo em vista que o React é uma biblioteca que herda do JavaScript, dessa maneira o React também pode ser trabalhado com o layout da página com o HTML e o CSS. Com base no exemplo da Figura 10, irei dividi-lo, para uma explicação mais objetiva:

* *Function*: está criando duas funções “App” e “dados” ambas sem parâmetro, sendo “dados” uma função para armazenar os dados inseridos, já a função “App” é a função principal do código basicamente tudo que for feito no código tem que estar dentro dessa função;
* *Var*: no exemplo do código, temos as variáveis “nome”, “email” e “senha” onde serão armazenados os dados;
* *Return*: uma função destinada a retornar algum valor quando for chamada, que na situação do exemplo o “*return*” está voltando 3 variáveis do mesmo tipo sendo elas “*String*” que quer dizer que são do tipo texto;
* *Const*: no exemplo está sendo usado para chamar as variáveis da função “dados” de forma constante;
* *Export default*: está fazendo a exportação de todo esse conteúdo colocado na função “*App*”, para a execução do código.

Porém, todo esse código para ter funcionalidade dentro do React são feitas importações para conseguir manusear diversas linguagens de programação em uma só biblioteca JavaScript, que nesse exemplo são:

* *Import* React: está fazendo a importação de um objeto “React” para um módulo chamado “react” que é um pacote que contém a biblioteca React;
* *Import* “./App.css”: uma importação de um arquivo css para poder ter a estilização da aplicação. Entre as aspas é colocada a rota que esse arquivo se encontra.

Figura 11 - Resultado da aplicação de um login simples feito em React

Interface gráfica do usuário, Texto, chat ou mensagem de texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# Python

Conforme afirma Mckinney (2018), Python é uma linguagem de codificação utilizada em funções de scripts, conduzindo eficiência com sua estrutura de código simples e tipagem dinâmica.

Ressaltando o que foi dito por Borges (2014), que possui um imenso conjunto de módulos e *frameworks*, tendo uma fácil execução na compilação dos códigos, possuindo recursos de outras linguagens modernas.

De acordo com Menezes (2010), o Python está em crescimento, sendo muito utilizado na construção de *websites* e aplicações *mobile*, podendo tratar a parte do *beck-end*, isto é, nos tratamentos dos dados.

Figura 12 - Exemplo de código para aplicação de cálculo de média em Python

Texto

Descrição gerada automaticamente

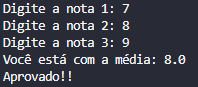
Fonte: Do próprio autor, 2024.

Dado o exemplo da Figura 12 acima, ele será desmembrado para que os comandos sejam explicados:

* *Def*: define um objeto de função, seu objetivo é efetuar uma ação que foi designada, ou seja, na situação do exemplo acima, temos uma função que é, “Calculando\_Media”;
* *Input*: essa é uma função que faz o recebimento dos valores para as três variáveis para, desse jeito, fazer o cálculo da média das três notas;
* media: uma variável que está armazenando o cálculo que será feito para tirar a média das três notas registradas em três variáveis que se chamam “nota1”, “nota2” e “nota3”;
* *If*: uma estrutura de condição para que o código seja mais eficiente podendo ampliar essa condição para outras com o comando “*Elif*”, nessa situação o “*If*” está sendo usado para dar uma condição para o programa que basicamente é, se “media” for igual ou maior que 8 mostrará uma mensagem que foi aprovado, caso contrário entra na estrutura “*Elif*” mostrando uma mensagem de recuperação, e em caso de uma nota muito baixa entra na estrutura “*Else*” informando que foi reprovado;
* *Print*: para imprimir qualquer tipo de dado ou conteúdo, seja de uma variável, até uma complexa função, sendo o comando mais utilizado não só em Python, mas em outras linguagens de programação também, no caso do exemplo acima está printando três mensagens uma de aprovado ou recuperação ou reprovado que aparecera para a pessoa dependendo da estrutura “*If*”;
* valores: uma variável criada para receber a função “Calculando\_Media” fazendo a sua chamada sem passagem de parâmetro.

Ao rodar a aplicação, a pessoa irá fornecer as notas e assim será feito o cálculo da média entre elas, conforme apresenta a Figura 13.

Figura 13 - Resultado do código do cálculo da média em Python pelo terminal



Fonte: Do próprio autor, 2024.

# NPM

Conforme define Pereira (2014), o *Node Package Manager* (NPM) significa gerenciador de pacotes do Node.js, como uma ferramenta para gerenciar módulos JS em projetos, facilitando a instalação e atualização das dependências de um módulo.

De acordo com Ihrig (2014), Node.js é um ambiente virtual para criar servidores *web* capazes de impulsionar projetos e aplicações. Sua instalação pode ser feita através do gerenciador de pacotes de um sistema operacional.

Powers (2017) destaca que, com o Node, o NPM vem juntamente instalado e os seus módulos podem ser adquiridos global ou localmente. Sendo capaz de descobrir todas as dependências presentes em um módulo e respectivamente instalar todas.

Assim, parafraseando Moraes (2021), o NPX, uma extensão do NPM, possui a função de executar módulos sem precisar de uma instalação prévia. Ele cria pastas temporárias para adicionar as dependências do pacote e logo após as remove.

Os principais comandos do NPM são:

* npm *install* nome\_do\_módulo: instala um módulo no projeto;
* npm *install* -g nome\_do\_módulo: instala um módulo global;
* npm *list*: lista todos os módulos do projeto;
* npm *list* -g: lista todos os módulos globais;
* npm *remove* nome\_do\_módulo: desinstala um módulo do projeto;
* npm *remove* -g nome\_do\_módulo: desinstala um módulo global;
* npm *update* nome\_do\_módulo: atualiza para a versão mais recente de um módulo;
* npm *update* -g nome\_do\_módulo: atualiza para a versão mais recente de um módulo global;
* npm -v: exibe a versão atual do NPM instalada na máquina.

Figura 14 - Exemplo de inicialização de um projeto NPM

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

A Figura 14 acima exemplifica a criação de um projeto em um *prompt* de comando do sistema operacional através do comando:

* mkdir ExemploNpm

Após isso, a pasta é acessada com o comando:

* cd ExemploNpm

E por fim, o prompt interativo do NPM é inicializado com o comando:

* npm *init*

Figura 15 - Exemplo de inserção dos dados exigidos pelo NPM

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Após a execução do último comando abordado na Figura 15, uma saída semelhante a essa será exibida no prompt de comando. O NPM solicitará automaticamente algumas informações do responsável pelo projeto, incluindo:

* *name*: nome do projeto;
* *version*: versão inicial do pacote Node.js;
* *description*: descrição escrita do projeto para que outras pessoas o encontrem, caso ele seja publicado na internet;
* *entry point*: arquivo principal que será a porta de entrada do projeto, que por padrão é o “*index*.js”;
* *test command*: comandos pré-definidos para testes feitos pelo NPM;
* *git repository*: se caso houver um repositório criado na plataforma de hospedagem, *github*, o NPM salvará as atualizações do projeto neste repositório automaticamente;
* *keywords*: palavras-chave do projeto para que outras pessoas o encontrem.
* *author*: através de alguns dados, como um *e-mail* e um *site,* é possível determinar um autor para o projeto;
* *license*: define as permissões e limitações legais que outras pessoas vão encontrar ao utilizar o módulo do projeto.

# JSON

Consoante a Bassett (2015), o JSON é um transportador de dados de forma simples, no qual os dados ficam entre chaves, sendo amplamente usado para intercâmbio de informações.

De acordo com Smith (2015), possuindo uma troca de dados de forma simplificada, logo não exige um conhecimento instantâneo da linguagem de programação JavaScript, Além do fato de ele ser formado e transmitido como *Unidcode*.

Segundo Júnior e Silva (2018), sendo uma estrutura de texto na forma de objeto JavaScript e sendo capaz de ser composto de subobjetos, dando escolhas de variadas interpretações de dados.

# Web Crawler

Conforme descreve Mitchell (2019), o Web Crawler, que traduzido para a língua vernácula significa rastreador da web é essencialmente um algoritmo de rastreamento com um elemento central de recursão.

Consoante Souza e Café (2018), esses capturadores são programas que coletam dados de forma contínua de fontes específicas, que podem ser desenvolvidos em Python e os dados armazenados em um banco de dados.

Assim, de acordo com Machado *et al.* (2015), o Web Crawler, também denominado *Spider* ou *Bot*, é um programa empregado para coletar informações disponíveis em páginas da web, como links e palavras-chave.

# Web Scraping

Segundo Farias, Angeluci e Passarelli (2021), o Web Scraping é o processo de criação de scripts, geralmente em Python, para extrair dados de forma automatizada e estruturada para análises ou outros fins.

Conforme demonstra Assis e Gomide (2021), essa técnica de mineração de dados online envolve buscar, extrair e organizar dados desestruturados em uma forma estruturada.

Portanto, segundo Rodrigues et al. (2021), essa coletar automática de dados pode ser útil para diversas aplicações, como na busca de documentações, relatórios e estatísticas em repositórios digitais.

# Framework

Para Gamma (2008), o *framework* é basicamente um conjunto de ferramentas que os desenvolvedores podem usar para criar e estender suas aplicações de forma mais eficiente.

Conforme expressado por Nascimento, Capanema e Pereira (2019), um *framework* pode acabar fazendo diversos processamentos, possuindo a capacidade de armazenar códigos e gerenciar recursos em uma aplicação.

Assim, de acordo com Bendoraitis e Kronika (2020), um dos aspectos importantes de um *framework* é a sua possibilidade de integrar sistemas que possam auxiliar nos projetos.

# Tailwind

Conforme descreve Abba (2023), o Tailwind é um framework CSS que simplifica a estilização de páginas web, possibilitando a escrita direta de CSS no HTML, por meio de suas classes predefinidas.

De acordo com Oliveira (2023), é possível criar componentes independentes com configurações que o próprio *framework* oferece, tornando a leitura do código mais "limpa" e de fácil compreensão para o desenvolvedor.

Portanto, Neves (2023) ressalta que a sua abordagem baseada em componentes otimiza o tempo e esforço do programador, já que poucas linhas de código precisam ser escritas para construir um *design* moderno.

Figura 16 - Exemplo de instalação do Tailwind

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

No Terminal, deve-se imprimir o seguinte comando:

* npm *install* -D tailwindcss

E logo após:

* npx tailwindcss *init*

Depois da instalação do Tailwind, será gerado um arquivo denominado de “tailwind.*config*.js”, nele deve ser inserido o caminho do seu arquivo HTML, como mostrado no exemplo da Figura 17:

Figura 17 - Exemplo de criação da conexão entre HTML e o Tailwind

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Agora, com a criação de dois arquivos em CSS, um principal e um utilitário, deve ser adicionado os diretivos do Tailwind no arquivo utilitário com os seguintes comandos descritos na Figura 18:

Figura 18 - Acrescentando diretivos do Tailwind

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Depois disso, deve existir uma conexão entre os dois arquivos CSS em seus respectivos caminhos com este comando no terminal:

* npx tailwindcss -i ./src/utilitário.css -o ./src/principal.css --watch

Figura 19 – Exemplo executando comando Tailwind

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Por fim, um link entre o CSS principal e o arquivo HTML deve ser feito e que será mostrado na Figura 20, na linha 7:

Figura 20 - Código HTML com a implementação do Tailwind

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

O código exibido acima resultará em uma interface de um login simples em HTML e Tailwind CSS que será representado na próxima figura.

As principais propriedades de Tailwind neste exemplo são:

* “bg-azul” e “bg-*white*”: conseguem definir qual cor vai ser utilizada no fundo do elemento;
* “*flex*” e “*justify-center*”: são capazes de alinhar o conteúdo da página ao centro da tela horizontalmente de maneira flexível;
* *rounded*-lg: compõe as bordas arredondadas ao elemento em um tamanho pré-definido pelo Tailwind;
* text-verde: é possível modificar a cor que o texto irá ser escrito na tela pela propriedade “text-(cor)”. E no exemplo, foi definida a cor verde;
* “m-6” e “p-5”: são responsáveis pelas margens e e preenchimento do corpo visual por um tamanho pré-definido pelo Tailwind, enquanto “mt-4” aplica uma margem para o elemento em relação aos elementos acima dele. Já “py” e “pl” são duas versões da propriedade para preenchimento, o primeiro modifica a área ocupada nos lados inferior e superior do elemento, já o segundo integra o lado esquerdo do elemento;
* “w-96”: consegue alterar o tamanho, pré-definido pelo Tailwind, da tamanho do elemento na tela;
* “*text*-4x1”: aumenta o tamanho do texto 4 vezes maior do que o padrão;
* “*text-center*”: organiza o texto de maneira centralizada;
* “*underline*”: impõe uma linha abaixo do texto, sublinhando a mesma;
* “*flex*-col”: faz com que os elementos contidos nele se organizem de maneira flexível em coluna;
* “*items*-*center­*”: permite que os elementos se organizem verticalmente dentro de um container;
* “border-2” e “border-verde”: criam uma borda e dão uma cor para ela, com um tamanho pré-definido pelo Tailwind.

Figura 21 - Resultado do código HTML com a implementação do Tailwind

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# NextJS

Conforme dito por Neto *et al.* (2022), o framework NextJS foi utilizado para ser trabalhado no visual do projeto, ou seja, no *front-end* da proposta.

Consoante a Alves (2023), o NextJS, também conhecido simplesmente como Next, possui uma escalabilidade e funcionalidades avançadas o que possibilita que seja possível criar aplicações de alta qualidade.

Segundo dito por Bolonha (2024), o benefício de utilizar o NextJS como uma ferramenta em seu projeto, é a sua leveza e segurança do seu sistema, que roda em um servidor chamado “*serverless*”, que traduzindo seria “sem servidor/menos servidor”.

# Flask

Segundo Grinberg (2018), o Flask é um microframework escrito em Python, porém ressalta que mesmo sendo um framework pequeno, não quer dizer que possua menos recursos que outros frameworks. Levando em conta a facilidade de entender o código.

De acordo com Silva (2019), como uma ferramenta auxiliar na criação de um sistema web, o Flask traz grandes benefícios quando utilizado, sendo um dos melhores frameworks para se trabalhar com Python.

Consoante Pereira (2018), por ser considerado um *microframework*, o Flask é ideal para desenvolver ferramentas que apoiam aplicações, oferecendo a flexibilidade necessária para adaptar-se a diferentes necessidades e escalas de projetos.

Figura 22 - Exemplo de código para aplicação de cálculo de média com o Flask

Texto

Descrição gerada automaticamente

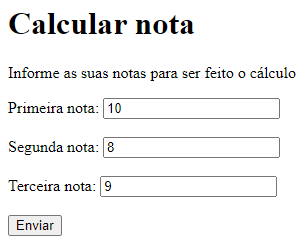
Fonte: Do próprio autor, 2024.

Dado o exemplo da Figura 22, ele será desmembrado para que os comandos sejam explicados:

* *From*: o “*From*” está indicando o *framework* que será importado;
* *Import*: um comando para importar as bibliotecas e *frameworks,* logoemseguidatemumcaracterequeéo asterisco *“\*”* que tem uma funçãode informar para o comando “*Import*” que é para importar tudo do módulo Flask;
* *App.route*: utilizado para fazer a passagem de parâmetro através da rota que for fornecida. No caso do exemplo, é o “Exemplo\_para\_TCC.html” que, por meio do método “*POST*”, passará a capturar as variáveis “nota1”, “nota2” e “nota3”, que o valor será colocado pelo usuário;
* *Render\_template*: é utilizado para renderizar o modelo HTML para *Hypertext Transfer Protocol* (HTTP), que no caso do código é o “Exemplo\_para\_TCC.html”;
* *Methods*: determina o tipo de método HTTP que será aplicado no código, na situação do exemplo é o método “*POST*”;
* *Request*: é um objeto fornecido pelo Flask que inclui os dados das notas enviados pelo usuário;
* *Redirect*: simplesmente é uma função de redirecionamento de páginas, especificada pela *Uniform Resource Locator* (URL), que no caso do código é a barra ‘/’;
* *App.run:* determina que quando a verificação for feita que o código está sendo executado direito, será realizado o método “*run*” no objeto “app”, que é uma instância do Flask, já o “*debug=True*” é opcional já que ele detalha sobre os erros e recarrega automaticamente a aplicação quando o código for alterado.

Para ser inserido os dados das notas, o usuário digitará na página web criada com HTML, tal como demonstra a Figura 23:

Figura 23 - Tela de inserimento dos dados em HTML



Fonte: Do próprio autor, 2024.

Com o desenvolvimento do Flask, quando os dados forem inseridos, o resultado da conta irá aparecer no terminal, como ilustrado na Figura 24.

Figura 24 - Resultado da codificação com o Flask no terminal



Fonte: Do próprio autor, 2024.

# Scrapy

De acordo com Duke (2018), Scrapy é um framework que extrai um conjunto de informações como textos e números para manipular e tratar os mesmos. Tudo que estiver disponível na internet pode ser obtido.

Conforme expressado por Gomes (2024), o processo de extração acontece de forma automatizada, e o framework utiliza a técnica Web Scraping juntamente ao Python, por ser uma linguagem de programação de fácil uso.

E para finalizar, Didática Tech (2022) nos mostra que as vantagens de utilizar o Scrapy são sua eficiência para lidar com abundância de coletas de dados e a fácil manipulação da codificação por um desenvolvedor.

Figura 25 - Exemplo de codificação em Python com o Scrapy

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

O código acima da Figura 25 apresenta um exemplo de codificação escrita na linguagem Python, onde podem ser obtidas informações de uma página através da técnica de *web scraping*.

As principais partes desta codificação são:

* *Import*: com este comando é possível importar classes do Scrapy, e no exemplo estão sendo importados o *Crawlspider* responsável pelas páginas *web* que terão seus dados obtidos através do *crawl*, o *Rule* que ajuda a definir limites por onde o *spider* deve percorrer *e* o *linkextractors* que permite especificar critérios para a extração de informações da página *web*;
* *Class*: determina o nome da classe que recebe o *CrawlSpider*;
* *Name*: identifica um nome para o *spider*;
* *Allowed*\_*domains*: restringe o *spider* para realizar o *crawl* apenas nas páginas da *web* permitidas;
* *Start*\_*urls*: especifica quais são as URLs das páginas *web* que o *spider* deve chegar para vasculhar as informações contidas nele;
* *Rules*: neste trecho, são armazenadas as regras como caminhos por onde o *spider* vai passar e filtrar os links que forem identificados e permitidos a ele através da URL especificada anteriormente;
* *Parse*\_*item*: esta função tem o objetivo de processar o conteúdo da página *web* e explorar tudo que foi permitido pela última regra definida, logo extraindo as informações dele. No exemplo, está sendo usado o método *css* do objeto *response*, que recebe o conteúdo da página *web*, para selecionar os títulos e preços dos livros e o método “*get()*” extrai os valores dos elementos selecionados pelo *css*;
* *Yield:* na última parte do código, a função *parse\_item* cria um dicionário que irá receber tudo que foi extraído para ser armazenado e visualizado posteriormente.

Figura 26 - Resultado da codificação com o Scrapy

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# APIs

Segundo dito por Muniz *et al.* (2023), uma *Application Program Interface* (API) possui uma execução de um sistema de forma padronizada e possibilita que duas partes de software se comuniquem.

Conforme mostrado por Stevens (2008), as APIs podem ser utilizadas em diversos ramos na tecnologia, seja para o desenvolvimento de uma aplicação até a construção de um sistema de rede.

Consoante Silva (2009), elas providenciam diversas formas de registros e finalidades, desempenhando um papel fundamental na interconexão e na integração de algum sistema.

# AwesomeAPI

Segundo Butewicz (2022), é possível criar um sistema de conversões monetárias com a linguagem de programação Python e a AwesomeAPI. Ele destaca que usar essa API é tão simples quanto qualquer outra biblioteca do Python.

Figura 27 - Exemplo de código com a AwesomeAPI

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Dado o exemplo de código da Figura 27, será separado em partes para uma explicação mais aprofundada:

* *Requests:* está importando pelo meio *“import”* o *requests* que é uma biblioteca do Python muito utilizada para fazer requisições com API*;*
* *Json:* outra importação que está sendo feita é do módulo Json para manipular os dados que a API fornece*;*
* *Link:* é uma variável que está armazenando o *link* da API de conversão monetária, quando podemos perceber que no *link* está digitado “USD-BRL” que significa que a API irá em busca da conversão de moedas do dólar americano para o real brasileiro*;*
* *req:* outra variável, mas esta variável está guardando a biblioteca *requests* que, usando o método “*get*” pegará a conversão que a API irá fornecer;
* Dumps: é uma função do módulo “Json” que converte um objeto Python em uma *String* Json formatada;
* Json: com esse método, está convertendo os dados que API está mandando para um formato Json e um objeto Python no do comando “req.json()”;
* *Indent*: está especificando que a saída Json deve ser estilizada com uma indentação de 4 espaços para tornar a visualização do resultado mais legível.

Figura 28 - Resultado da aplicação com a AwesomeAPI

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

No resultado da Figura 28, é mostrada a conversão do dólar americano para o real brasileiro, informa a quantia de ambos, além de mostrar a data e a hora em que foi feita essa execução.

# Banco de Dados

Segundo Alves (2021), desde os primórdios, o homem sentiu a necessidade de armazenar informações, assim, com o crescimento do meio computadorizado essa necessidade resultou na criação de bancos de forma bem mais prática e rápida.

Portanto, conforme descreve Zhao (2023), o banco de dados é um conjunto de dados organizados, que tem como objetivo armazenar informações em um sistema computacional.

De acordo com Date (2004), um Sistema Gerenciador de Banco de Dados (SGBD) é usado para gerenciar um ou mais bancos de dados. Ele permite a modelagem de dados, tornando os SGBDs cruciais para a organização e manipulação dos dados.

# Modelagem de Dados

Segundo Heuser (2009), a modelagem de dados é um processo que busca descrever de maneira abstrata os dados a serem armazenados, facilitando a compreensão e interpretação.

Consoante Machado (2020), construir uma modelagem de dados corretamente requer uma compreensão essencial das operações lógicas e da organização dos dados nos quais serão armazenados.

**DER**

De acordo com Cardoso e Cardoso (2012), o Diagrama Entidade Relacionamento (DER) é o primeiro passo na representação visual de um banco de dados, enfatizando sua importância para evitar erros em projetos de banco de dados.

Figura 29 - Exemplo da criação de um DER

Uma imagem contendo Polígono

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Na Figura 29 acima, é apresentado um exemplo simplificado de um DER, onde podemos visualizar os dados de um banco de forma clara. Nele, são destacadas as entidades "cliente" e "pedido", juntamente com seus atributos correspondentes.

**MER**

Conforme demonstra Sordi (2019), o Modelo Entidade Relacionamento (MER) tem como base a representação dos dados em um sistema, proporcionando uma visão clara das entidades e de seus respectivos atributos e relacionamentos.

Figura 30 - Exemplo da criação de um MER

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024

Já na Figura 30 acima, é destacado um exemplo de MER, onde inclui duas entidades, que são: “cliente” e “pedido”, demonstrando cada entidade com seus atributos específicos, como nome e email para cliente, e campos adicionais como pedido\_id, data\_pedido e valor para pedido. O relacionamento é estabelecido através do campo cliente\_id, indicando que cada pedido está associado a um cliente específico.

# MySQL

Conforme caracteriza Milani (2006), O MySQL é um SGBD, que supre de maneira vantajosa o armazenamento de dados e as informações de um meio que hoje está cada vez mais popularizado.

De acordo com Lobo (2008), há vários bancos de dados disponíveis, mas é essencial escolher o software apropriado para o sistema no qual será desenvolvido. O MySQL é uma opção competitiva, robusta e vantajosa em termos de licenciamento.

Dessa forma, segundo Alves (2017), esse gerenciador de banco de dados relacional e de código aberto é considerado muito recorrente entre os desenvolvedores para aplicações *web*.

Figura 31 - Exemplo de um código de banco de dados MySQL

Texto

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Na Figura 31 é configurado um código de criação de um banco de dados no MySQL e sua respectiva tabela como exemplo, logo em seguida são apresentados os comandos principais de uma aplicação MySQL, que são:

* *Create*: em português significa criar, seu uso é feito para a criação de um banco que é acompanhado do *Database* e para a criação de uma tabela que é acompanhado do *Table*.
* *Insert*: traduzindo para a língua nacional significa inserir, é usado para inserir dados em uma tabela, junto ao comando *Into*, que vem antes do nome da tabela e o *Values* que passa os valores que vão ser inseridos na tabela.
* *Select*: na língua vernácula quer dizer selecionar, é usado para verificar vários ou um dado que é acompanhado do *From* que determina em qual tabela ele irá selecionar e do *Where* caso queira selecionar de alguma informação específica.
* *Update*: é usado para atualizar um dado que já foi registrado anteriormente é seguido pelo *set* que determina o que vai ser mudado e pelo *Where* que define de qual lugar é esse dado que vai ser transformado.
* *Delete*: transpondo para o português significa deletar, empregado para excluir dados que foram registrados anteriormente.

Figura 32 - Resultado de uma consulta de uma tabela MySQL

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# UX

Conforme descreve Furtado (2023), *User Experience (*UX), que traduzindo significa Experiência do Usuário, refere-se à toda usabilidade que engloba as reações e ações de uma pessoa ao experimentar ou interagir com um produto.

Parafraseando Grant (2019), é evidente que os profissionais dessa área detêm percepções valiosas por meio de estudos e práticas, contudo, é necessário empatia e objetividade como principais habilidades para uma boa criação UX.

Segundo Podmajersky (2019), existe a redação UX, a qual é empregada como diálogo entre a experiência por meio de elementos visuais e palavras e o usuário, que responde interagindo com esses recursos.

Assim, de acordo com Levy (2021), quando implementada uma estratégia de UX adequada, isso traz com que possamos revolucionar o mercado com um produto que seja único ou até uma melhor alternativa dos que existem, por meio da inovação.

# Wireframe

De acordo com Teixeira (2014), o *Wireframe* é crucial para o desenvolvimento de uma aplicação, pois desenha protótipos que representam como ﻿uma interface visual vai ser e ajuda na organização do conteúdo dentro do projeto.

Conforme dito por Memória (2006), o desenvolvimento de um *Wireframe* sempre começa ﻿com baixa fidelidade e vai ganhando forma conforme a proposta avança. Dessa maneira, os detalhes da interface vão se tornando cada vez mais refinados.

Em concordância com Pereira (2018), acrescenta e traz que tal é um esboço mais detalhado, possuindo um propósito de comunicar as melhores decisões de uma interface de alguma aplicação, seja *web* ou *mobile*.

Figura 33 - Exemplo de Wireframe de baixa fidelidade.

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Na figura 33 anterior temos o wireframe de baixa fidelidade onde foi colocado o escopo visual inicial de uma página web. Já a seguir a Figura 34 é um *wireframe* de média fidelidade, possuindo mais informações do que o de baixa fidelidade.

Figura 34 - Exemplo de Wireframe de média fidelidade.

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

E por fim, na Figura 35 temos o *wireframe* de alta fidelidade mais detalhado que reflete o escopo final do visual da respectiva página web, possuindo cores, textos mais trabalhados e entre outros elementos.

Figura 35 - Exemplo de Wireframe de alta fidelidade.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# UML

De acordo com Guedes (2018), A *Unified Modeling Language* (UML) é uma linguagem visual e de modelagem baseada no paradigma de orientação a objetos, composta por diagramas com o objetivo de definir características de um sistema a ser aplicado.

Conforme afirmam Pressman e Maxim (2021), essa linguagem se tornou o padrão internacional para a criação de *software* de todos os tipos. Assim, compreender seus elementos permite especificar e entender um sistema de forma fácil.

Segundo Fowler (2005), ao usar a UML, as equipes de desenvolvimento podem melhorar a comunicação e a compreensão entre os membros, tornando o processo de desenvolvimento mais eficiente.

Portanto, consoante Booch, Rumbaugh e Jacobson (2006), a UML é amplamente utilizada em diversos contextos, abrangendo itens estruturais, comportamentais, de agrupamento, descritivos, relacionamentos e diagramas.

**Diagrama de Casos de Uso**

Conforme descreve Fowler (2005), os diagramas de caso de uso são representações gráficas que têm por objetivo fornecer uma visão das funcionalidades de um sistema, identificando os atores e os serviços que ele oferece.

Figura 36 - Exemplo de Diagrama de Casos de Uso

**Diagrama

Descrição gerada automaticamente**

Fonte: (Guedes, 2018)

De acordo com Guedes (2018), esse é um exemplo de diagrama de casos de uso de um sistema de controle bancário, que define as funcionalidades, bem como seus atores caracterizando interações como abertura e encerramento de conta.

**Documentação de Caso de Uso**

Segundo Pressman e Maxim (2021), as documentações de caso de uso são usadas para descrições formais de um caso de uso específico, que inclui elementos como atores, resumo do escopo, precondição, cenário e exceções.

Figura 37 - Exemplo de documentação de Caso de Uso

Interface gráfica do usuário, Texto, Tabela

Descrição gerada automaticamente

Fonte: (Guedes, 2018)

Consoante Guedes (2018), a documentação de caso de uso do processo de emissão de saldo descreve o resumo, as ações do sistema, exceções e os atores que interagem com esse caso de uso específico.

**Diagrama de Sequência**

De acordo com Booch, Rumbaugh e Jacobson (2006), um diagrama de sequência foca na ordem temporal das mensagens entre papéis. Ele exibe os papéis e as mensagens trocadas entre suas instâncias, destacando a visão dinâmica do sistema.

Figura 38 - Exemplo de Diagrama de Sequência

Diagrama

Descrição gerada automaticamente

Fonte: (Guedes, 2018)

Evidenciado por Guedes (2018), o exemplo do diagrama de sequência do sistema de controle bancário revela as sequências para a emissão de saldo por meio suas respectivas instâncias.

**Diagrama de Atividade**

Conforme aponta Fowler (2005), um diagrama de atividades demonstra o fluxo de trabalho entre ações em um sistema, sendo representado através de nós de ação, setas e outros elementos visuais.

Figura 39 - Exemplo de Diagrama de Atividade

Diagrama

Descrição gerada automaticamente

Fonte: (Guedes, 2018)

Segundo Guedes (2018), nesse diagrama de atividade de emissão de saldo temos todas as ações que moldam essa atividade que vão desde receber o número da conta, nó de ação e nó de decisão até de fato apresentar o saldo.

**Diagrama de Máquina de Estados**

Consoante Booch, Rumbaugh e Jacobson (2006), o diagrama de Máquina de Estados representa a visão dinâmica de um sistema, mostrando uma máquina de estados composta por estados, transições, atividades e eventos.

Figura 40 - Exemplo de Diagrama Máquina de Estados

Diagrama

Descrição gerada automaticamente

Fonte: (Guedes, 2018)

Conforme demonstra Guedes (2018), nesse exemplo de diagrama de máquina de estados, é utilizado o caso de uso de emissão de saldo, representando os estados desse processo, bem como suas transições e eventos.

# DESENVOLVIMENTO

Nesta seção, serão detalhados todos os processos de criação do sistema web Navigate Buy, no qual a documentação será apresentada utilizando UML, acompanhada de *wireframes* de baixa e alta fidelidade que expressam as telas do sistema web.

# Diagrama de Casos de Uso

Primeiramente, serão descritos os requisitos funcionais, requisitos não funcionais e as regras de negócio, e logo após será fornecido o diagrama e casos de uso da aplicação.

Requisitos Funcionais:

* RF01: O Consumidor poderá efetuar o cadastro de sua conta.
* RF03: O Consumidor poderá efetuar o login de sua conta.
* RF04: O Consumidor poderá consultar dados do perfil de sua conta.
* RF05: O Consumidor poderá editar dados de sua conta.
* RF06: O Consumidor poderá buscar produto.
* RF07: O Scrapy poderá mapear os produtos.
* RF08: O Consumidor poderá buscar comentários de avaliação.
* RF09: O Scrapy poderá mapear os comentários de avaliação.
* RF10: O Consumidor poderá favoritar algum produto pelo qual foi buscado.
* RF11: O Consumidor poderá adicionar a opção de receber notificações sobre os produtos favoritados.
* RF12: O Consumidor poderá consultar os produtos que foram favoritados.
* RF13: O Consumidor poderá editar os produtos que foram favoritados.
* O Consumidor poderá comparar preços de produtos internacionais.
* RF14: A AwesomeAPI poderá fazer conversões monetárias.

Requisitos Não Funcionais:

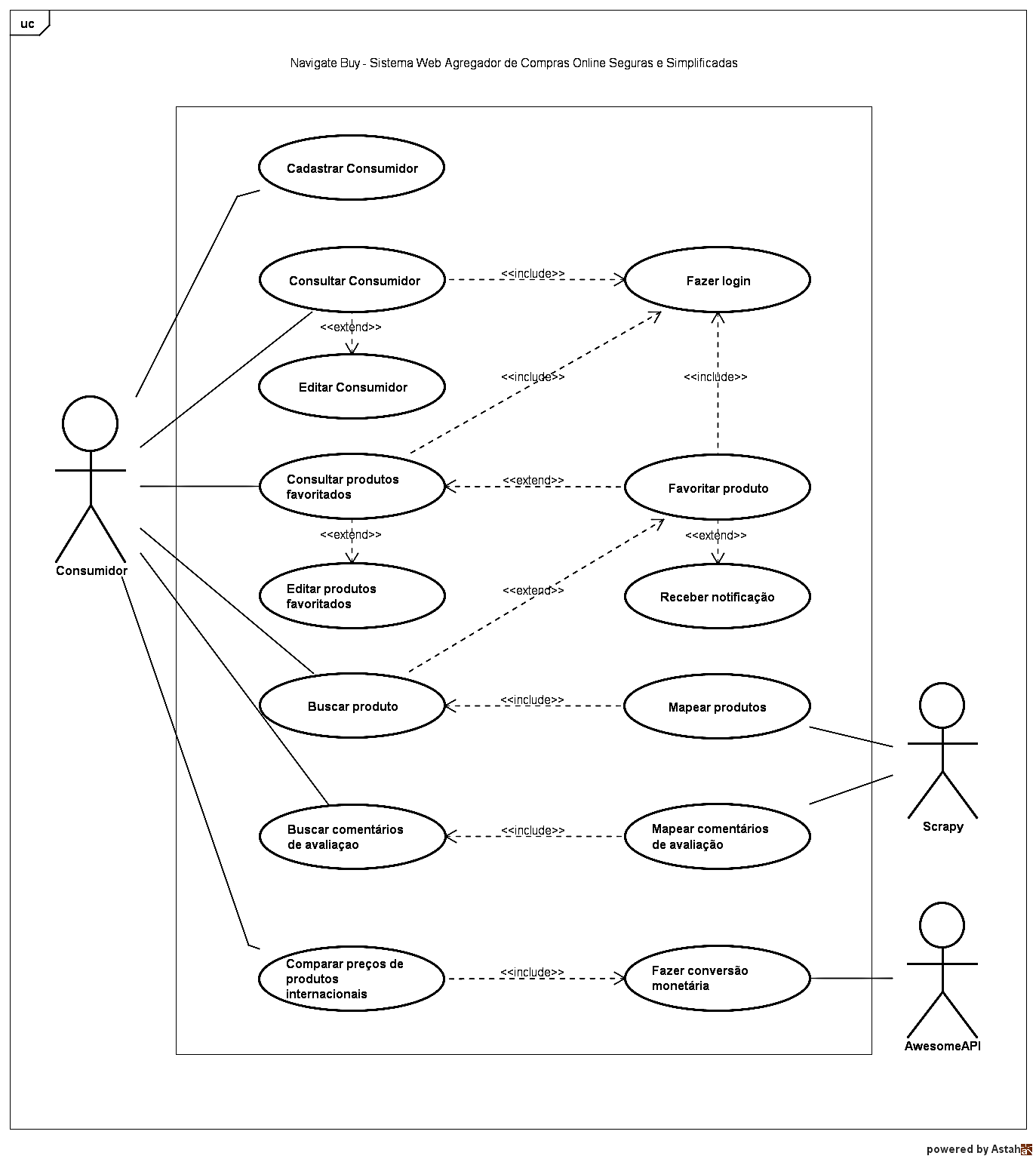
* RNF01: Os cards dos produtos deverão ter botões/links que os direcionam para as respectivas ofertas em sua loja online de origem.
* RNF02: Os resultados de busca dos comentários de avaliação serão retirados e indexados de sites como o Reclame aqui ou outros motores de pesquisa.
* RNF03: Todas as notificações serão recebidas pelo consumidor via Email.
* RNF04: Todos os códigos de verificação para cadastro serão enviados via Email.
* RNF05: Os dados deverão ser salvos no banco junto a senha com o protocolo criptográfico de segurança MD5.
* RNF06: As conversões monetárias deverão ser feitas através do AwesomeAPI com uma taxa de atualização de 30 segundos.

Regras de Negócio:

* RN01: As filtragens serão realizadas exclusivamente por relevância, preço ou avaliações, com a de relevância definida como padrão.
* RN02: Apenas lojas reconhecidas e com hosts verificados, serão incluídas para maior segurança na pesquisa e análise do consumidor.

Abaixo na Figura 41, segue o diagrama de casos de uso desenvolvido, que demonstra como os atores Consumidor, Scrapy e AwesomeAPI, interagem com o sistema web e as funcionalidades que o descrevem.

Figura 41 - Diagrama de Casos de uso do sistema web



Fonte: Do próprio autor, 2024.

# Documentações de Caso de Uso

Dando continuidade, serão apresentadas as tabelas das documentações de cada caso de uso, na mesma ordem em que foram listados os Requisitos Funcionais (RF).

Quadro 1 - Documentação do caso de uso "Cadastrar Consumidor"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Cadastrar Consumidor |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator Consumidor cadastrar sua conta no sistema |
| Pré-Condições | Validar código de verificação |
| Pós-Condições | Fazer o login da sua conta |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Inserir suas informações pessoais e clica no botão para se cadastrar. |  |
|  | 1. Verificar se os dados inseridos são válidos. |
|  | 1. Enviar código de verificação via email. |
| 1. Inserir código de verificação. |  |
|  | 1. Validar código de verificação. |
|  | 1. Todos os dados são recebidos e armazenados no banco de dados, com a senha criptografada. |
| Restrições/Validações | 1. A senha precisa ter no minímo 8 caracteres. |

Fonte: Do próprio autor, 2024.

Quadro 2 - Documentação do caso de uso "Fazer login"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Fazer login |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a etapa de fazer o login, ou seja, acessar sua conta |
| Pré-Condições | Cadastrar Consumidor |
| Pós-Condições | O cliente está autenticado e pode acessar as funcionalidades do sistema de acordo com  suas permissões |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Inserir suas informações pessoais já cadastradas. |  |
|  | 1. Verificar se esses dados existem no banco. |
|  | 1. Autentificar login, caso a conta esteja registrada. |
| **Fluxo Alternativo I** | |
| Ações do ator | Ações do sistema |
|  | 1. Após a verificação, a conta/perfil não for encontrada no banco. |
|  | 1. Solicitar a realização do cadatro ou que revise os dados digitados. |
| **Fluxo Alternativo II** | |
| Ações do ator | Ações do sistema |
|  | 1. Após a verificação, a conta/perfil for encontrada no banco mas a senha estiver incorreta. |
| 1. Selecionar esqueci senha. |  |
|  | 1. Enviar email para o editar a senha. |
| 1. Fazer login com sua senha nova. |  |

Fonte: Do próprio autor, 2024.

Quadro 3 - Documentação do caso de uso "Consultar Consumidor"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Consultar Consumidor |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator consultar os dados do perfil da sua conta |
| Pré-Condições | Fazer login |
| Pós-Condições | Editar Consumidor |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Entrar no seu perfil. |  |
|  | 1. Fornecer os dados do perfil conforme o login da conta feito com sucesso. |

Fonte: Do próprio autor, 2024.

Quadro 4 - Documentação do caso de uso "Editar Consumidor"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Editar Consumidor |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator caso ele queira editar os dados da sua conta |
| Pré-Condições | Consultar Consumidor |
| Pós-Condições | Atualizar os dados cadastrados |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Selecionar o campo que deseja editar. |  |
| 1. Inserir os novos dados e apertar no botão para editar. |  |
|  | 1. Todos os dados são recebidos e atualizados no banco de dados. |
| Restrições/Validações | 1. Não fornecer a possibilidade de editar os dados do email. |

Fonte: Do próprio autor, 2024.

Quadro 5 - Documentação do caso de uso "Buscar produto"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Buscar Produto |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários | Scrapy |
| Resumo | Este caso de uso descreve a ação do ator ao buscar um produto para pesquisa e análise de melhores ofertas. |
| Pré-Condições | Acessar a barra de busca de um produto |
| Pós-Condições | Mapear produtos |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Inserir o nome do produto pelo qual deseja buscar. |  |
| 1. Apertar no botão de busca. |  |
|  | 1. Fornecer produtos de diferentes lojas relacionados a busca a partir da fdo mapeamento feito pelo scrapy. |
| **Fluxo Alternativo I** | |
|  | 1. Caso após qualquer busca, não tenha encontrado nenhum resultado, informar ao ator consumidor. |

Fonte: Do próprio autor, 2024.

Quadro 6 - Documentação do caso de uso "Mapear produtos"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Mapear produtos |
| Caso de Uso Principal |  |
| Ator Principal | Scrapy |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator ao mapear produtos referente a busca |
| Pré-Condições | Buscar produto |
| Pós-Condições | Fornecer produtos |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Fazer o mapeamento do produto pelo qual foi buscado |  |
|  | 1. Fornecer produtos após filtragem. |

Fonte: Do próprio autor, 2024.

Quadro 7 - Documentação do caso de uso "Buscar comentários de avaliação"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Buscar comentários de avaliação |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários | Scrapy |
| Resumo | Este caso de uso descreve a ação do ator de buscar comentários de avaliação para melhor análise de ofertas online |
| Pré-Condições | Acessar a barra de busca para cometários de avaliação |
| Pós-Condições | Mapear comentários de avaliação |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Inserir o nome do produto ou o nome da loja pelo qual deseja encontrar avaliações e comentários. |  |
|  | 1. Fornecer comentários de avaliação de outros consumidores referente aquele produto ou loja pesquisada, caso encontre. |
| **Fluxo Alternativo I** | |
| 1. Inserir o nome do produto e o nome da loja pela qual deseja pesquisar reclamações pelo qual deseja encontrar comentários de avaliação |  |
|  | 1. Fornecer comentários de avaliação referente aquele produto e loja pesquisada, caso encontre. |
| **Fluxo Alternativo II** | |
| 1. Caso o consumidor queira ele pode a partir de um produto buscado apertar em um link que está de baixo de cada produto. |  |
|  | 1. Direcionar o ator para buscar com o nome do produto pelo qual o link estava em baixo. |
| 1. Apertar o botão da barra de pesquisa para buscar aquele nome de produto. |  |
|  | 1. Fornecer comentários de avaliação referentes a busca feita. |
| **Fluxo Alternativo III** | |
|  | 1. Caso após qualquer busca, não tenha encontrado nenhum resultado, informar ao ator consumidor. |

Fonte: Do próprio autor, 2024.

Quadro 8 - Documentação do caso de uso "Mapear comentários de avaliação"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Mapear comentários de avaliação |
| Caso de Uso Principal |  |
| Ator Principal | Scrapy |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator ao mapear comentários de avaliação referente a busca |
| Pré-Condições | Buscar comentários de avaliação |
| Pós-Condições | Fornecer comentários de avaliação |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Fazer o mapeamento do nome do produto e/ou loja pelo qual foi buscado |  |
|  | 1. Fornecer comentários de avaliação |

Fonte: Do próprio autor, 2024.

Quadro 9 - Documentação do caso de uso "Favoritar produto"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Favoritar produto |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator ao favoritar um produto no sistema |
| Pré-Condições | Fazer login |
| Pós-Condições | Receber notificações |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Buscar o nome de um produto. |  |
|  | 1. Fornecer produtos de diferentes lojas relacionados a busca a partir da filtragem. |
| 1. Favoritar um ou mais produtos. |  |
|  | 1. Perguntar se deseja receber notificações e armazenar produto favoritado na conta do ator Consumidor. |

Fonte: Do próprio autor, 2024.

Quadro 10 - Documentação do caso de uso "Receber notificações"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Receber notificações |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator de escolher receber notificações sobre algum produto via email |
| Pré-Condições | Favoritar produto |
| Pós-Condições | Editar produtos favoritados |
| **Fluxo Principal** | |
| Ações do ator: | Ações do sistema |
| 1. Favoritar um produto. |  |
|  | 1. Perguntar se deseja receber notificações e armazenar produto favoritado na conta do ator Consumidor. |
| 1. Responder que deseja receber notificações. |  |
|  | 1. Enviar lembretes de notificação sobre o produto. |
|  | 1. Possibilitar que essa funcionalidade seja editada depois no editar produtos favoritados. |
| **Fluxo Alternativo I** | |
| 1. Favoritar um produto. |  |
|  | 1. Perguntar se deseja receber notificações e armazenar produto favoritado na conta do ator Consumidor. |
| 1. Responder que não deseja receber notificações. |  |
|  | 1. Possibilitar que essa funcionalidade seja editada depois no editar produtos favoritados. |

Fonte: Do próprio autor, 2024.

Quadro 11 - Documentação do caso de uso "Consultar produtos favoritados"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Consultar produtos favoritados |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator consultar os produtos que foram favoritados em sua conta |
| Pré-Condições | Fazer login |
| Pós-Condições | Editar produtos favoritados |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Entrar na aba dos produtos favoritados. |  |
|  | 1. Fornecer os dados dos produtos favoritados conforme o login da conta feito com sucesso. |

Fonte: Do próprio autor, 2024.

Quadro 12 - Documentação do caso de uso "Editar produtos favoritados"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Editar produtos favoritados |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários |  |
| Resumo | Este caso de uso descreve a ação do ator caso ele queira editar os produtos que foram favoritados na sua conta |
| Pré-Condições | Consultar produtos favoritados |
| Pós-Condições | Atualizar os dados dos produtos favoritados |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Selecionar o produto favoritado que deseja editar. |  |
| 1. Fazer a mudança desejada, seja desfavoritar ou editar a configuração de recebimento denotificações. |  |
|  | 1. Todos os dados são recebidos e atualizados. |

Fonte: Do próprio autor, 2024.

Quadro 13 - Documentação do caso de uso "Comparar preços de produtos internacionais"

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Comparar preços de produtos internacionais |
| Caso de Uso Principal |  |
| Ator Principal | Consumidor |
| Atores Secundários | AwesomeAPI |
| Resumo | Este caso de uso descreve a ação do ator Consumidor ao fazer uma comparação de preços de produtos internacionais |
| Pré-Condições | Acessar a interface para comparar preços internacionais |
| Pós-Condições | Fazer conversão monetária |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Inserir quantia a ser convertida. |  |
| 1. Inserir moeda que será convertida e para qual será convertida. |  |
|  | 1. Fornecer os dados para a AwesomeAPI. |

Fonte: Do próprio autor, 2024.

Quadro 14 - Documentação do caso de uso "Fazer conversão monetária"

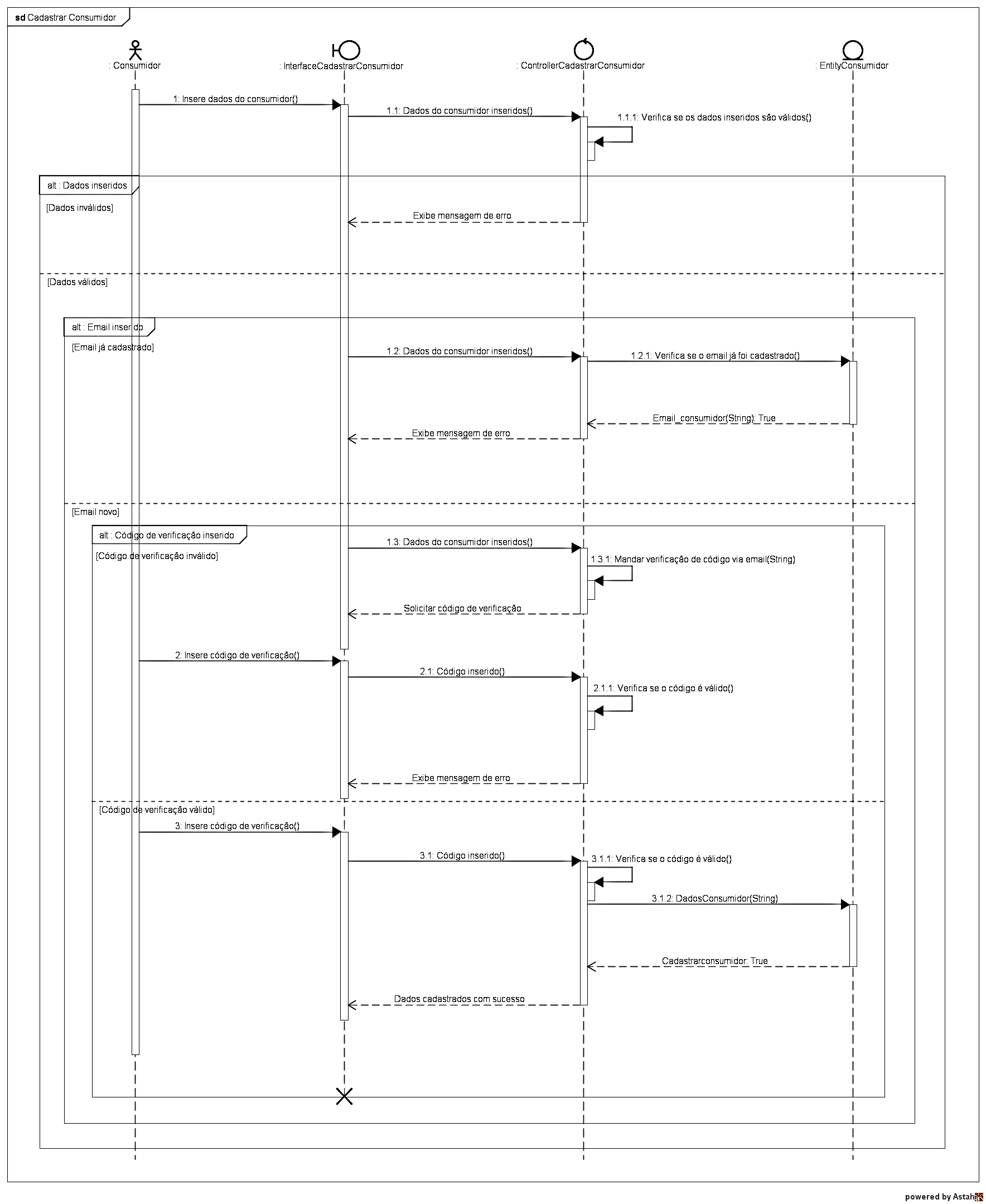
|  |  |
| --- | --- |
| **Nome do Caso de Uso** | Fazer conversão monetária |
| Caso de Uso Principal |  |
| Ator Principal | AwesomeAPI |
| Atores Secundários | Consumidor |
| Resumo | Este caso de uso descreve a ação do ator AwesomeAPI ao fazer uma conversão monetária |
| Pré-Condições | Adicionar valor e moedas para conversão |
| Pós-Condições | Fornecer conversão monetária para análise |
| **Fluxo Principal** | |
| Ações do ator | Ações do sistema |
| 1. Fazer a conversão das taxas de quanto custa as moedas escolhidas |  |
| . | 1. Calcular taxas de importação |
|  | 1. Fornecer resultado da conversão monetária para o ator Consumidor. |

Fonte: Do próprio autor, 2024.

# Diagramas de Sequência

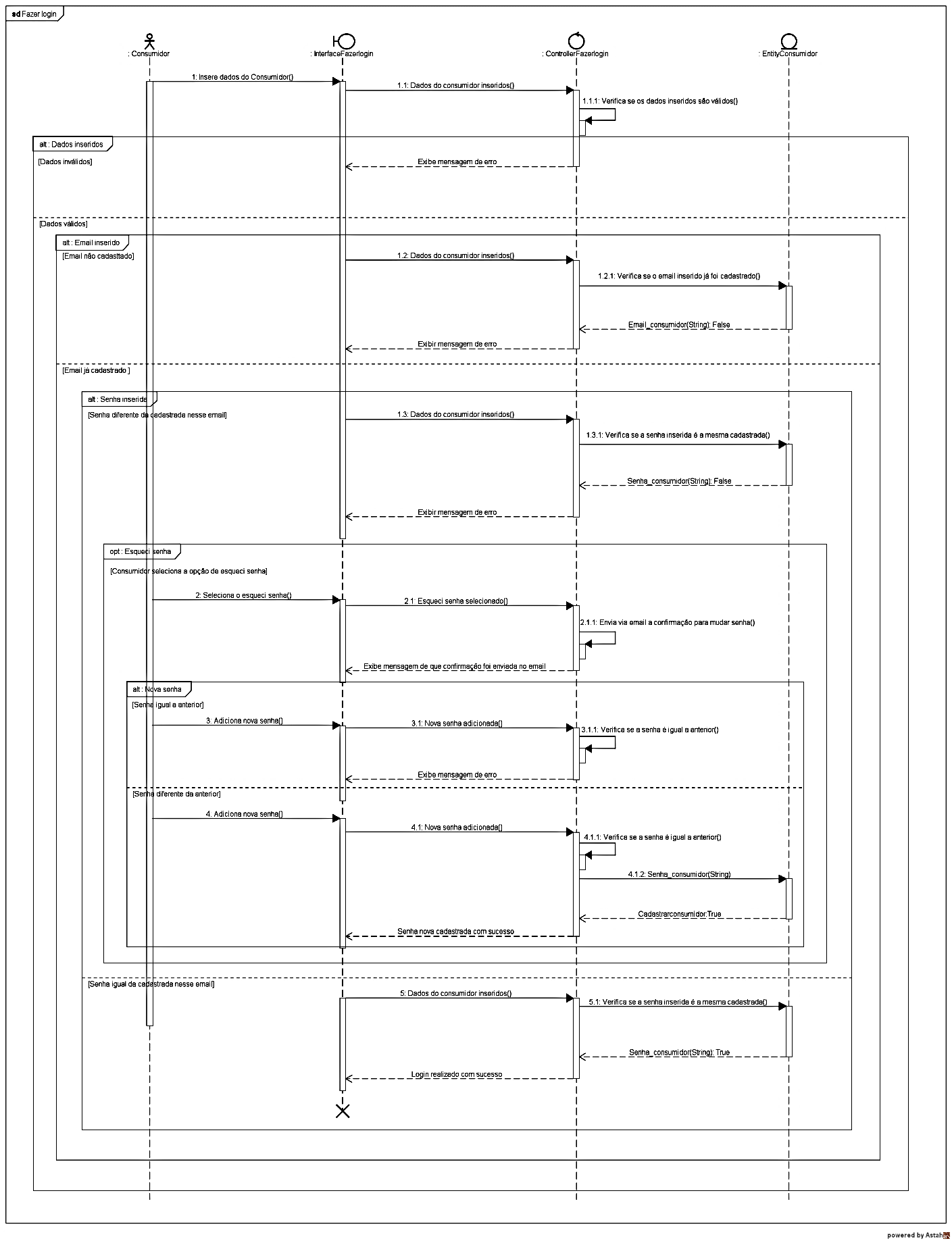
Neste capítulo, serão demonstrados os diagramas de sequência da aplicação, que ilustram detalhadamente o fluxo de mensagens e interações entre os atores, interfaces, controladores e entidades nos casos de uso, que vão desde a Figura 42 até a Figura 51.

Figura 42 - Diagrama de sequência "Cadastrar Consumidor"



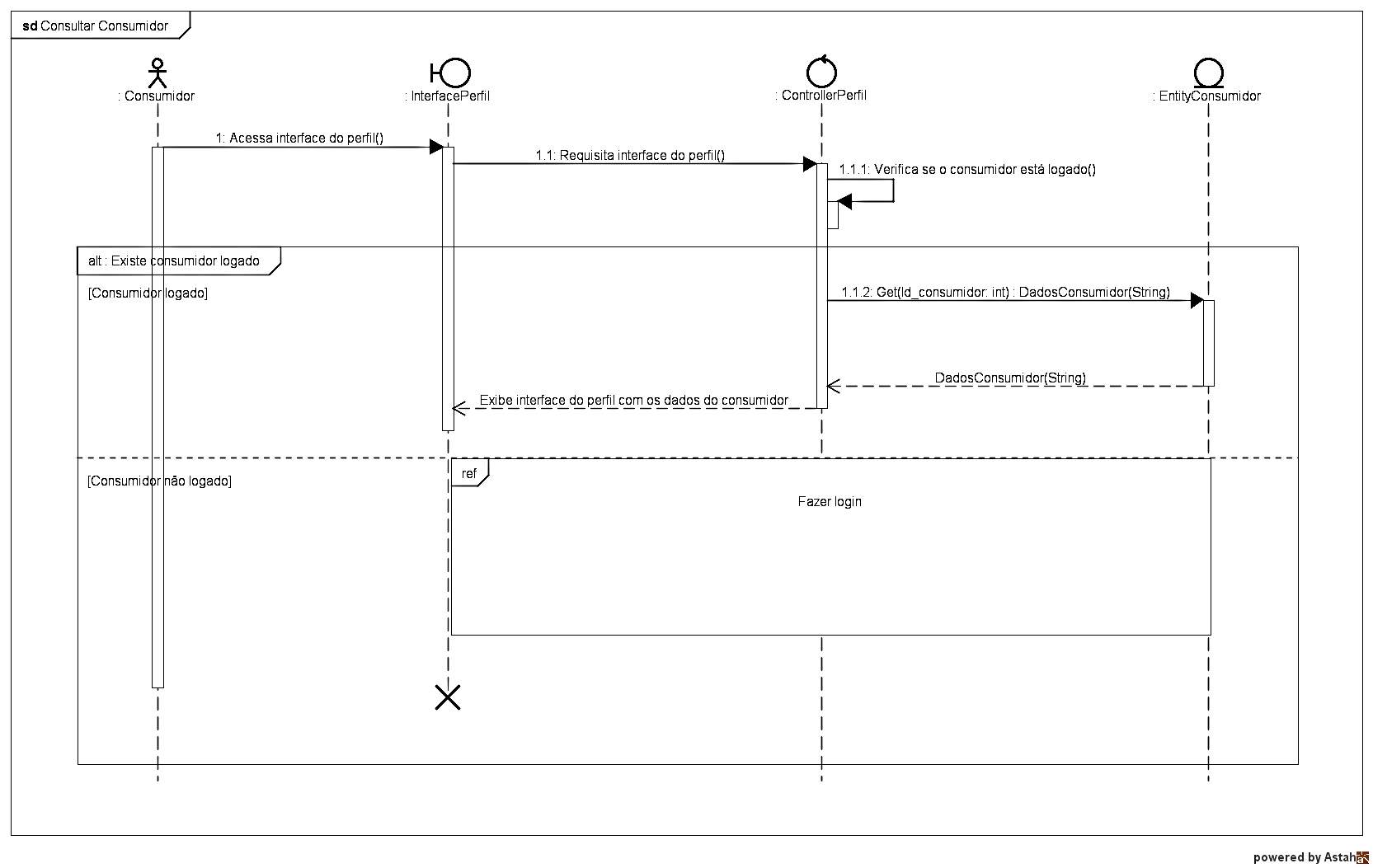
Fonte: Do próprio autor, 2024.

Figura 43 - Diagrama de sequência "Fazer login"



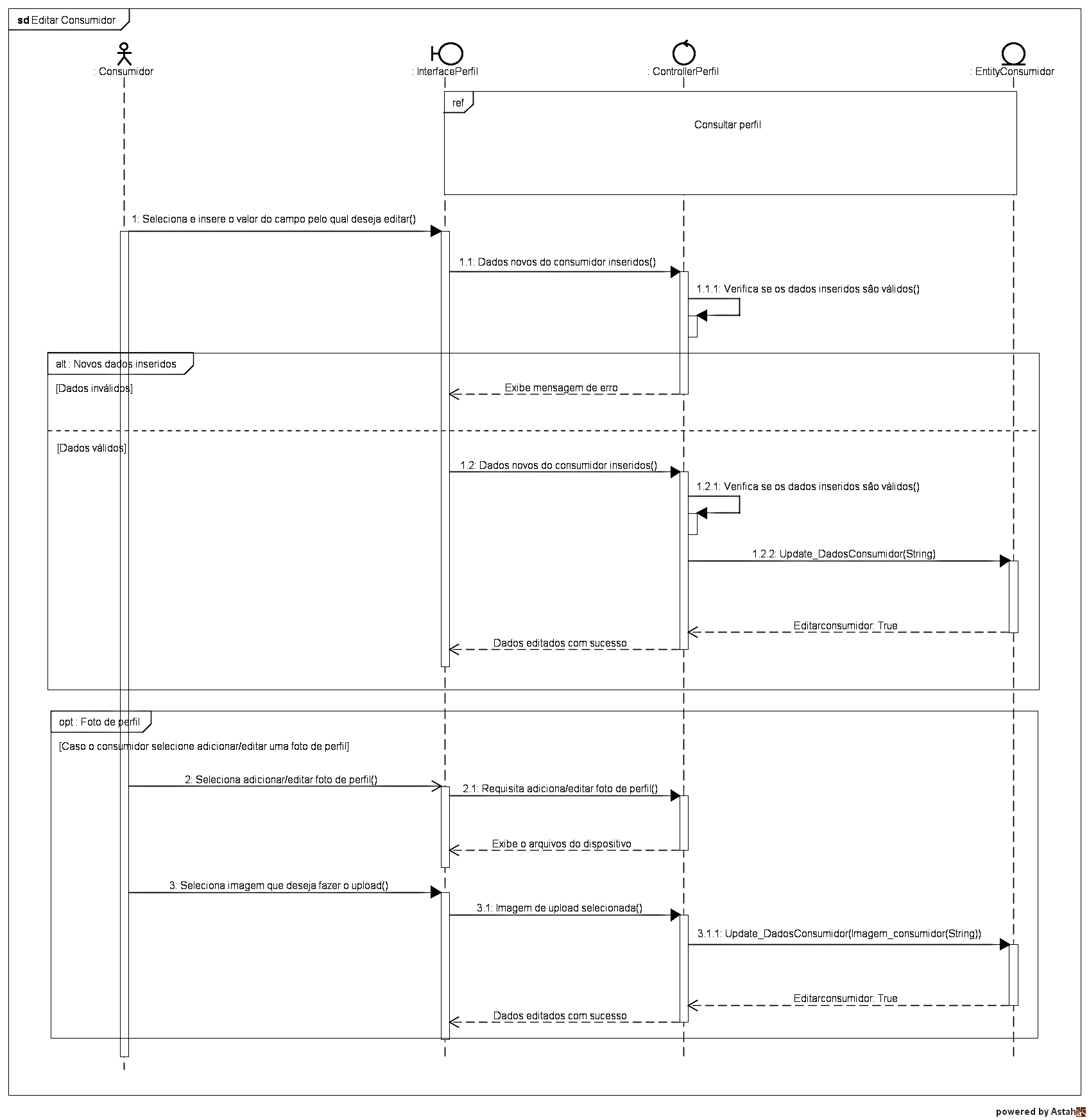
Fonte: Do próprio autor, 2024.

Figura 44 - Diagrama de sequência "Consultar Consumidor"



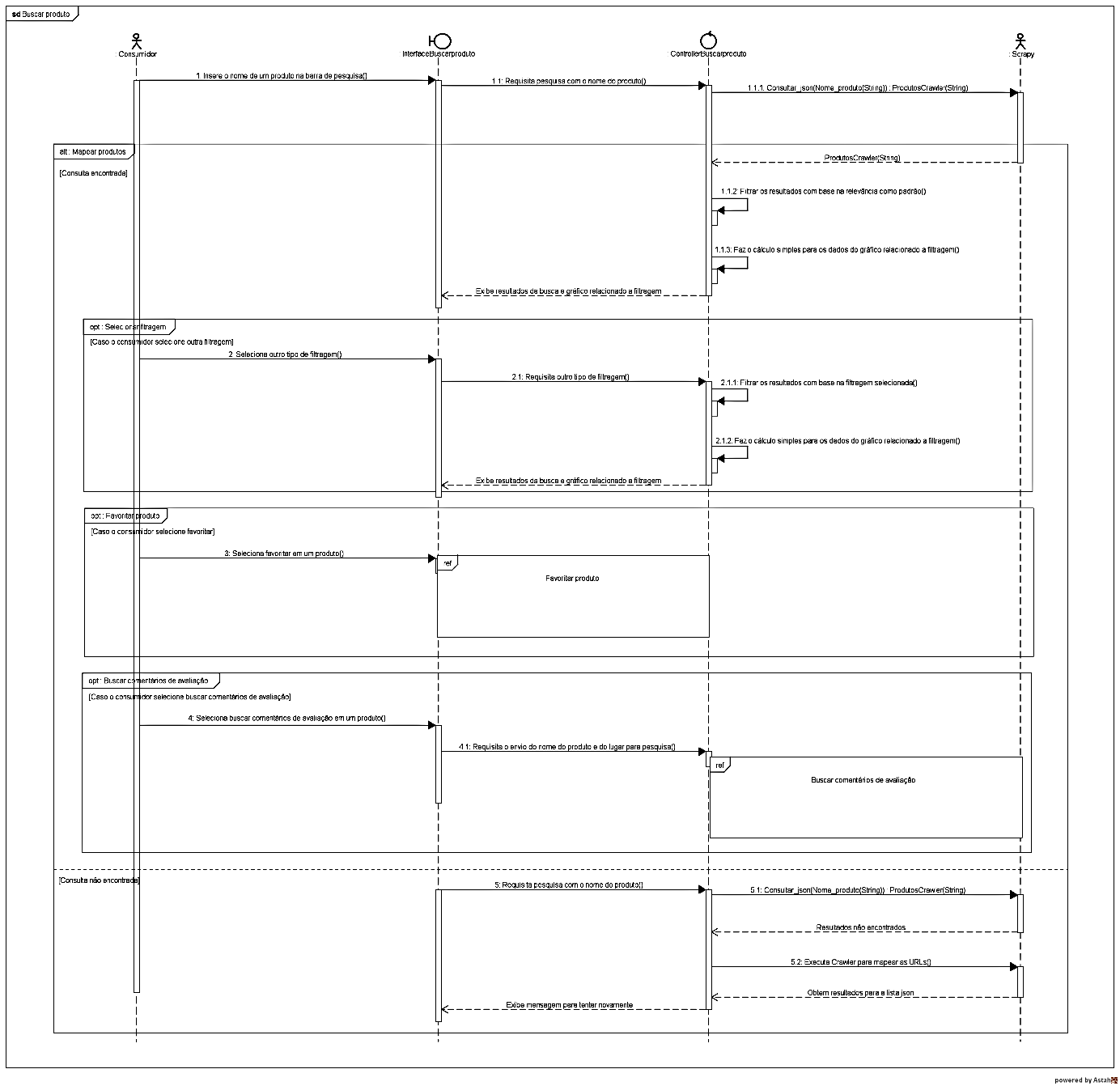
Fonte: Do próprio autor, 2024.

Figura 45 - Diagrama de sequência "Editar Consumidor”



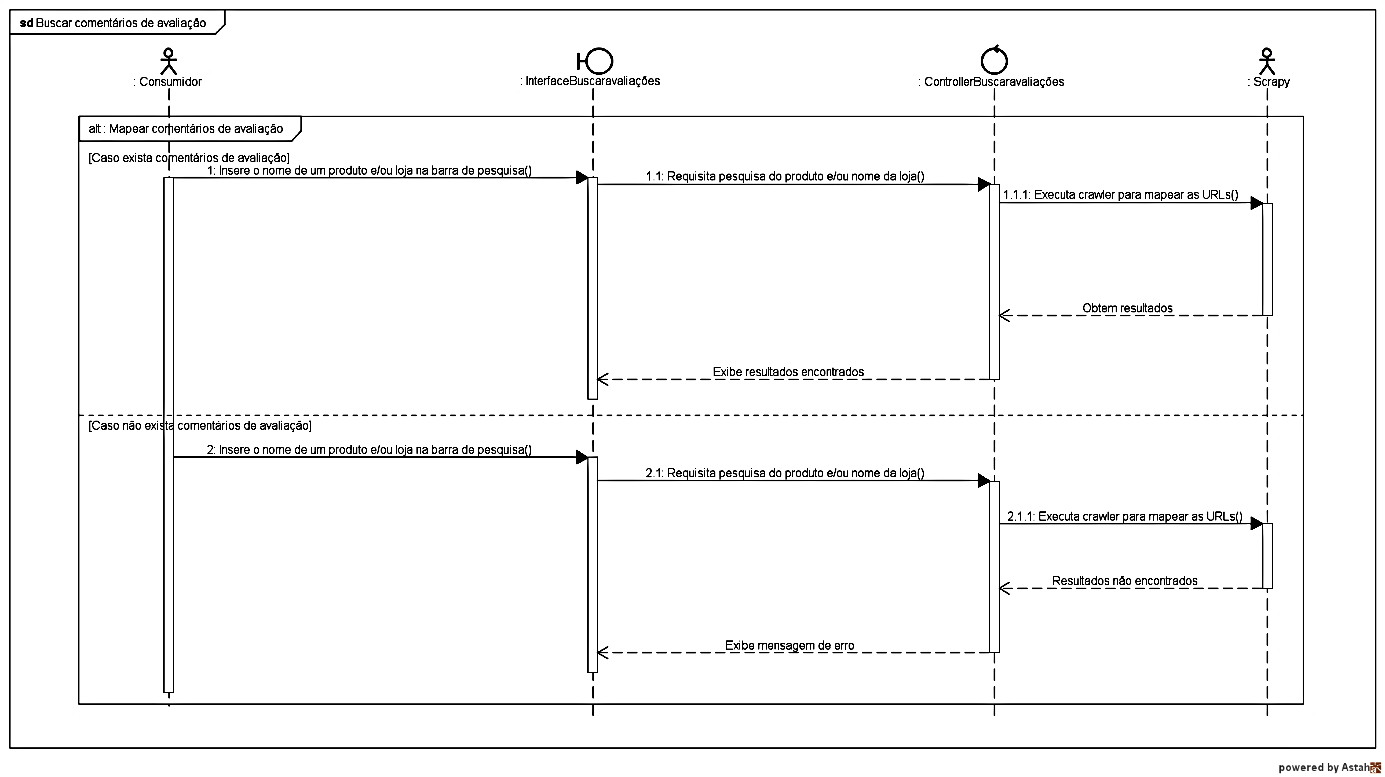
Fonte: Do próprio autor, 2024.

Figura 46 - Diagrama de sequência "Buscar produto”



Fonte: Do próprio autor, 2024.

Figura 47 - Diagrama de sequência "Buscar comentários de avaliação”



Fonte: Do próprio autor, 2024.

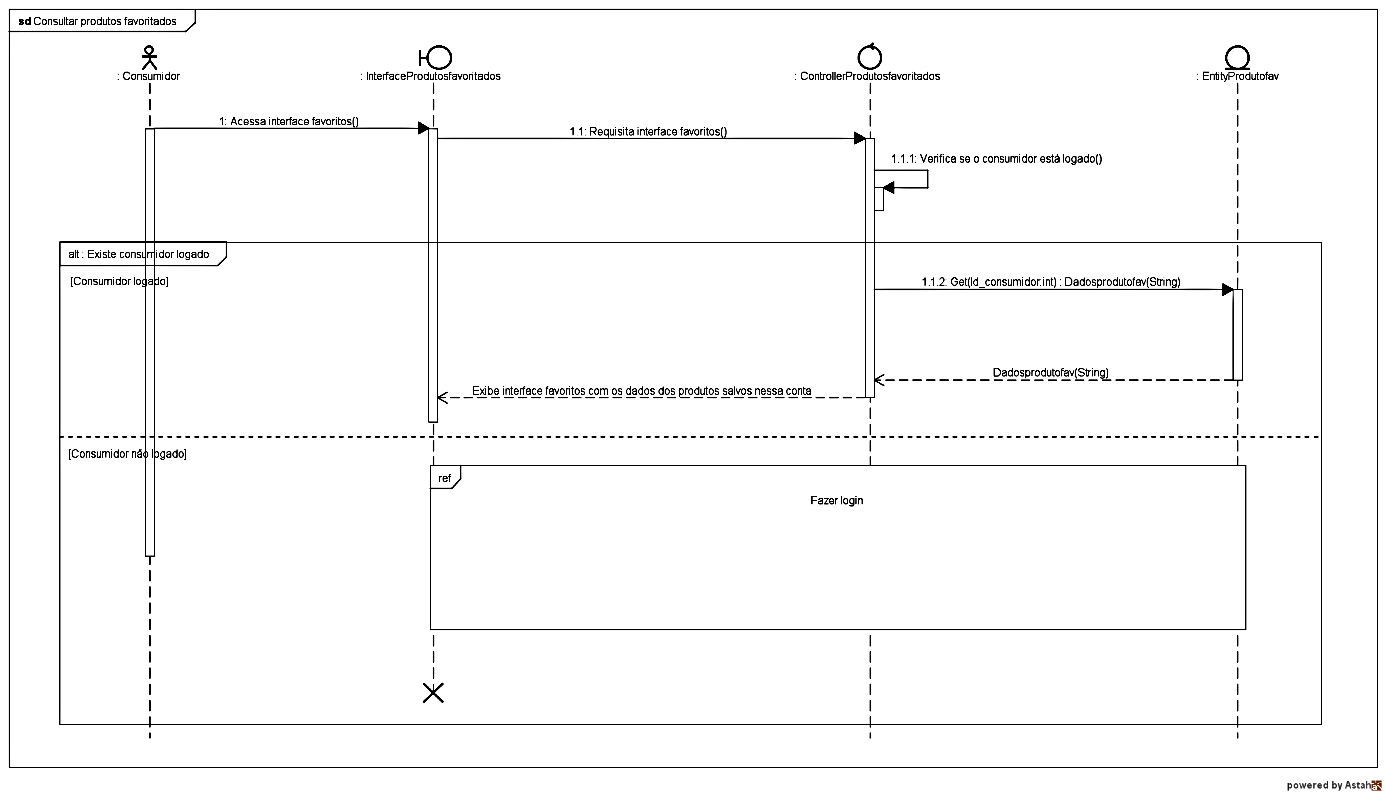
Figura 48 - Diagrama de sequência "Favoritar produto”

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 49 - Diagrama de sequência "Consultar produtos favoritados”



Fonte: Do próprio autor, 2024.

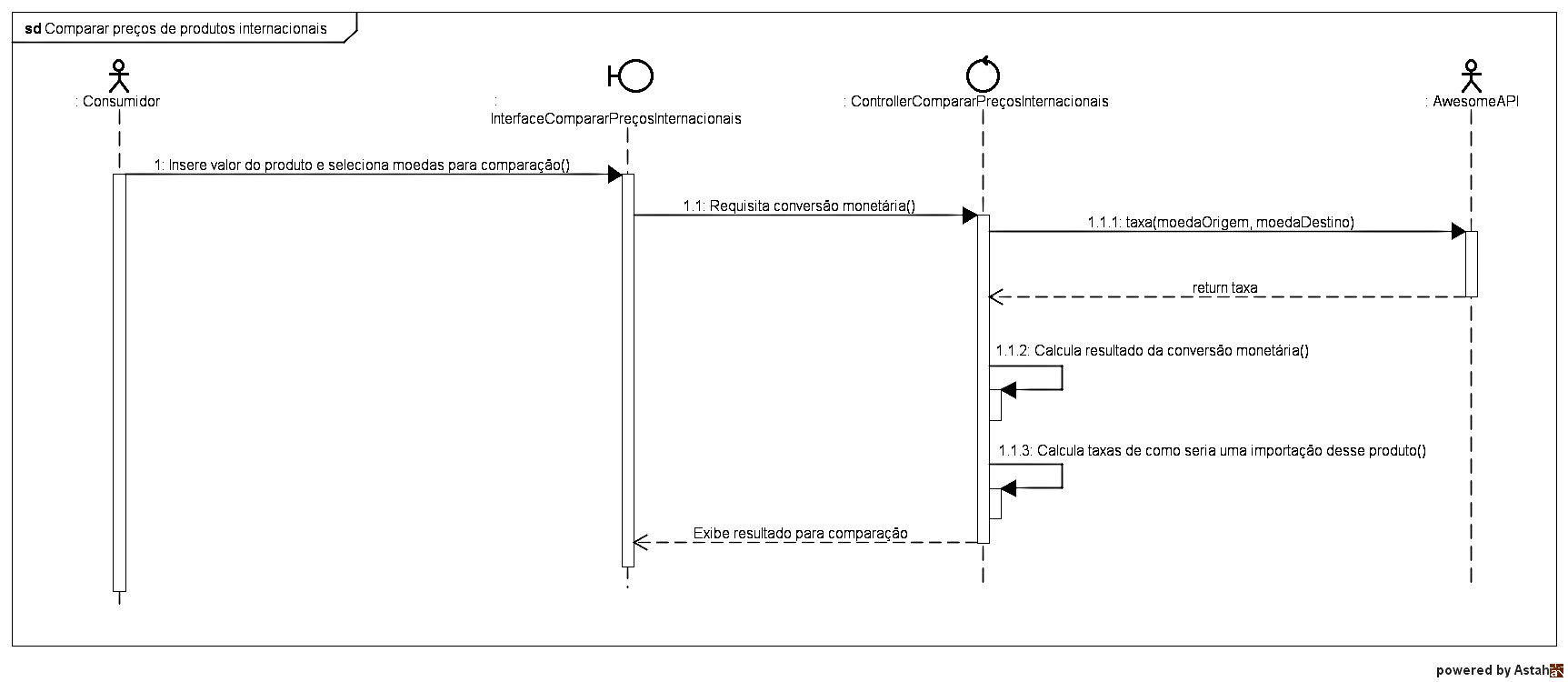
Figura 50 - Diagrama de sequência "Editar produtos favoritados”

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 51 - Diagrama de sequência "Comparar preços de produtos internacionais”



Fonte: Do próprio autor, 2024.

# Diagramas de Atividade

Portanto, nesse momento serão esclarecidos os diagramas de atividade da aplicação que abrangem da Figura 52 à Figura 61, apresentando as atividades do Sistema *Web*, Navigate Buy.

Figura 52 - Diagrama de Atividade "Cadastrar Consumidor"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 53 - Diagrama de Atividade "Fazer login"



Fonte: Do próprio autor, 2024.

Figura 54 - Diagrama de Atividade "Consultar Consumidor"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 55 - Diagrama de Atividade "Editar Consumidor"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 56 - Diagrama de Atividade "Buscar produto"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 57 - Diagrama de Atividade "Buscar comentários de avaliação"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 58 - Diagrama de Atividade "Favoritar produto"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 59 - Diagrama de Atividade "Consultar produtos favoritados"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 60 - Diagrama de Atividade "Editar produtos favoritados"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 61 - Diagrama de Atividade "Comparar preços de produtos internacionais"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

# Diagramas de Máquina de Estados

# Prototipação das páginas da aplicação

Este capítulo apresenta as páginas do sistema web, desenvolvidas a partir dos *wireframes* de baixa e alta fidelidade, incluindo suas respectivas descrições, fornecendo uma visão clara do layout e da estrutura de cada uma.

Nas Figuras 62 e 63, podemos observar a primeira interface, que corresponde ao cadastro dos consumidores.

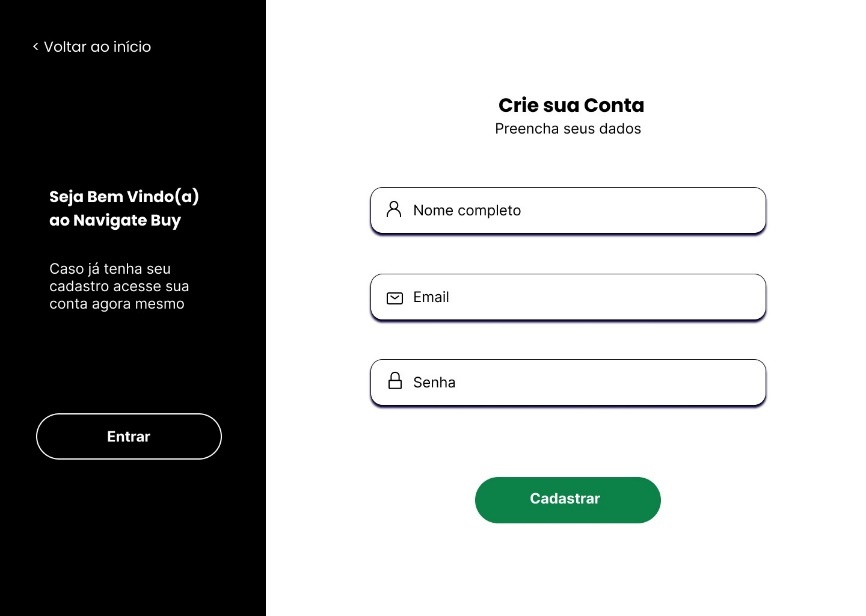
Figura 62 - Wireframe de baixa fidelidade da interface “Cadastro”

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 63 – Wireframe de alta fidelidade da interface "Cadastro"



Fonte: Do próprio autor, 2024.

Em seguida, após o consumidor completar o cadastro, ele pode realizar o login na tela apresentada na Figura 64 e Figura 65, para acessar funcionalidades específicas que serão descritas adiante.

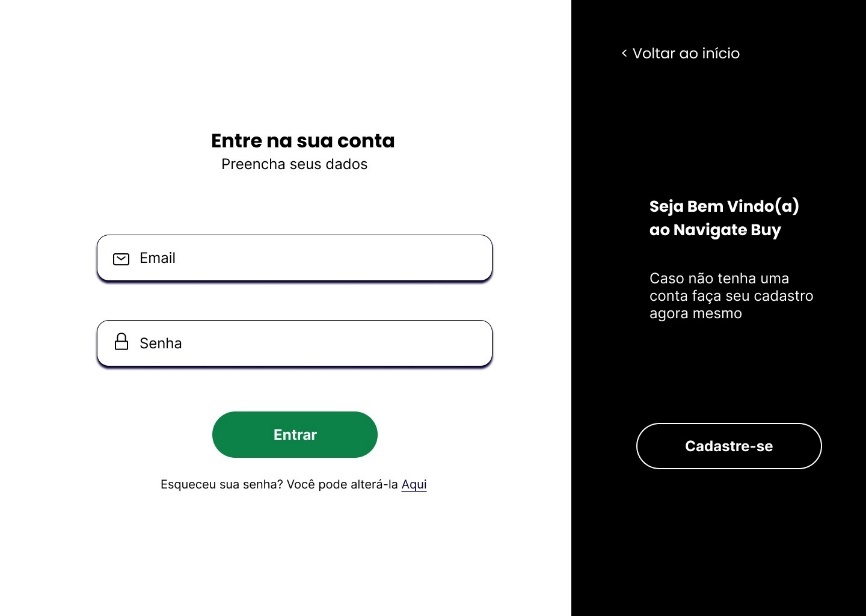
Figura 64 - Wireframe de baixa fidelidade da interface “Login”

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 65 - Wireframe de alta fidelidade da interface "Login"



Fonte: Do próprio autor, 2024.

Na interface representada nas Figuras 66 e 67, o consumidor pode visualizar e editar seu perfil. Essa interface permite ao usuário atualizar suas informações pessoais, como nome, senha e foto de perfil.

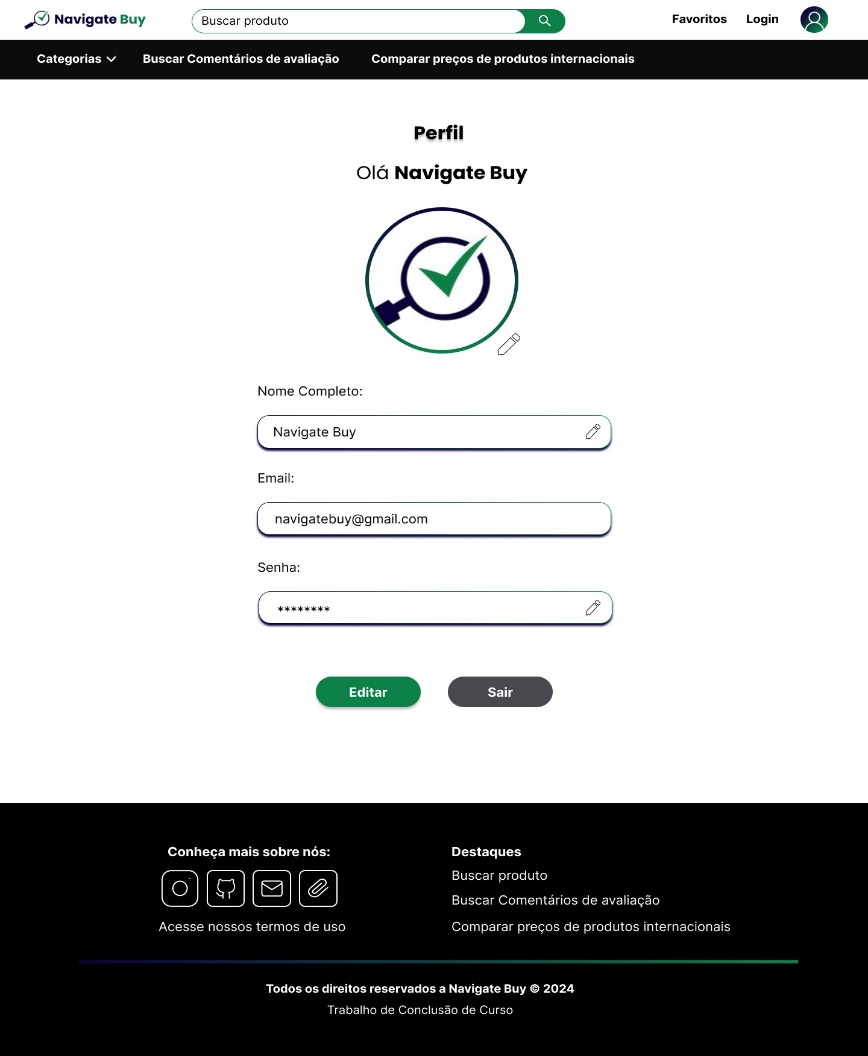
Figura 66 - Wireframe de baixa fidelidade da interface "Perfil"

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 67 - Wireframe de alta fidelidade da interface "Perfil"



Fonte: Do próprio autor, 2024.

Na Figura 68 e 69, é apresentada a tela inicial do Navigate Buy que não é necessário o login para acessá-la, onde são descritas as principais lojas nacionais que foram feitos os *scrapys*, exemplos de principais categorias e recomendações de segurança como a lista “Evite esses sites” do Procon SP e verificar se o link de sites são seguros com o *Google Transparency Report*, para melhores compras online.

Figura 68 - Wireframe de baixa fidelidade da interface "Home"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 69 - Wireframe de alta fidelidade da interface "Home"



Fonte: Do próprio autor, 2024.

Já na Figura 70 e 71, temos uma das principais funcionalidades do sistema que também não é necessário o login, onde o consumidor pode buscar produtos através de uma categoria ou digitar um nome específico na barra de pesquisa, dessa forma serão apresentados os resultados daquele respectivo produto em diferentes lojas nacionais mais populares e com *hosts* de URLs seguros, a partir disso o consumidor pode filtrar esses produtos por ordem de maior relevância, menor preço, maior preço e maior avaliação, podendo visualizar gráfico da filtragem escolhida que fornece o maior e menor valor encontrado da filtragem selecionada.

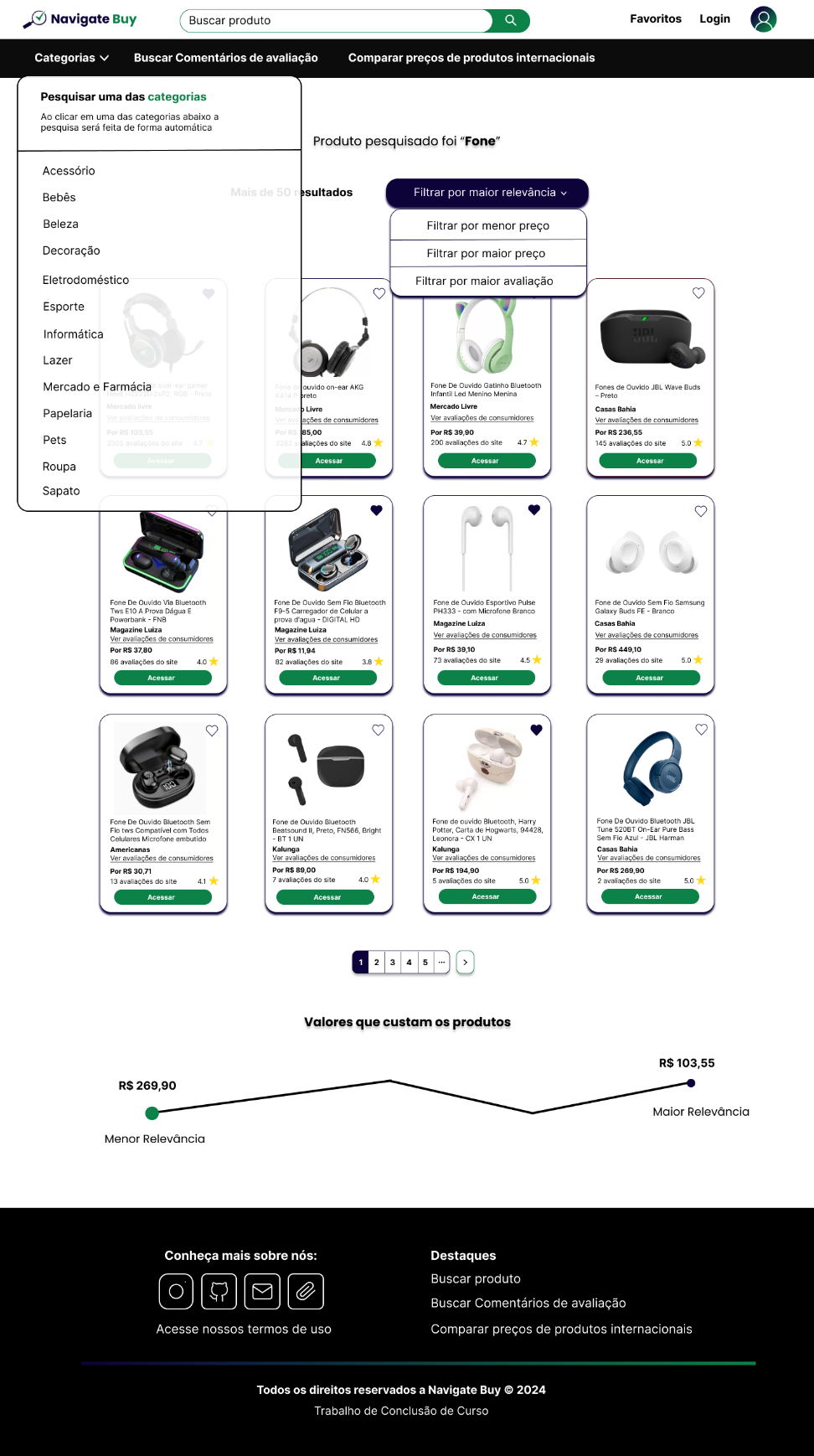
Figura 70 - Wireframe de baixa fidelidade da interface "Buscar produto"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 71 - Wireframe de alta fidelidade da interface "Buscar produto"



Fonte: Do próprio autor (2024)

Caso o consumidor deseje favoritar algum dos produtos que pesquisou, será necessário que ele realize o cadastro e login de sua conta, conforme demonstrado nas Figuras 62, 63, 64 e 65. Já na Figura 72, a seguir, é apresentada a interface de produtos favoritados, onde o consumidor pode não apenas visualizar os itens que foram adicionados à sua lista de favoritos, mas também editar essas preferências. Além disso, ele terá a opção de escolher receber alertas por e-mail referente a esses produtos, adicionando lembretes que há algum produto favoritado na conta dela.

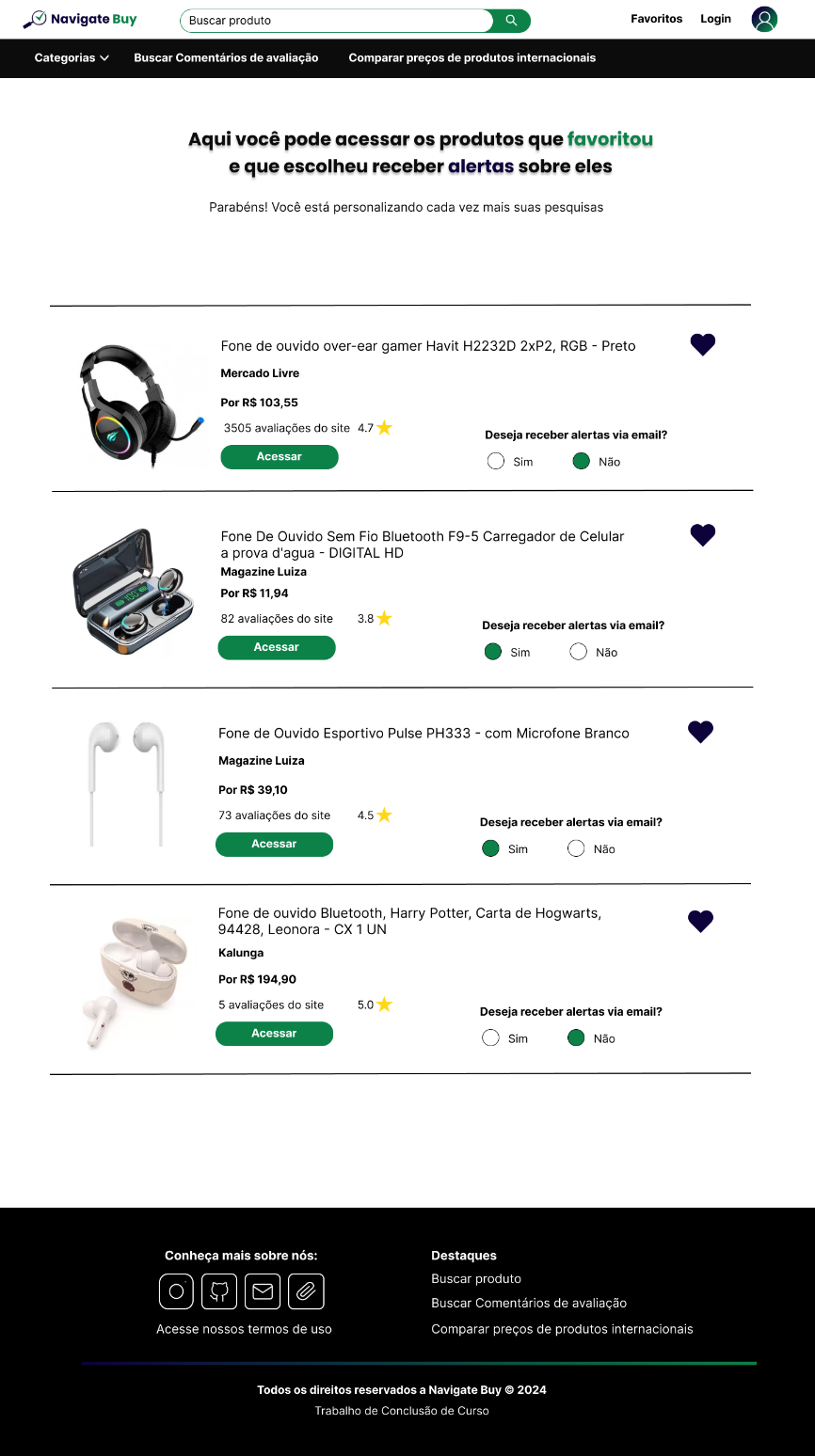
Figura 72 - Wireframe de baixa fidelidade da interface "Produtos favoritados"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 73 - Wireframe de alta fidelidade da interface "Produtos favoritados"



Fonte: Do próprio autor, 2024.

Nas Figuras 74 e 75, está representada outra funcionalidade principal do sistema web sendo ela buscar comentários de avaliação, que permite ao consumidor pesquisar um produto e/ou o nome da loja da qual deseja buscar comentários de avaliação no qual fizemos os *scrapys* e foram indexados do Reclame Aqui. Dessa forma, ele pode ser direcionado à página do Reclame Aqui, onde poderá analisar se a reclamação correspondente teve solução ou se ainda persiste, causando algum tipo de prejuízo para o comprador, sendo fundamental para auxiliar os consumidores na tomada de decisões mais informadas antes de efetuar uma compra online.

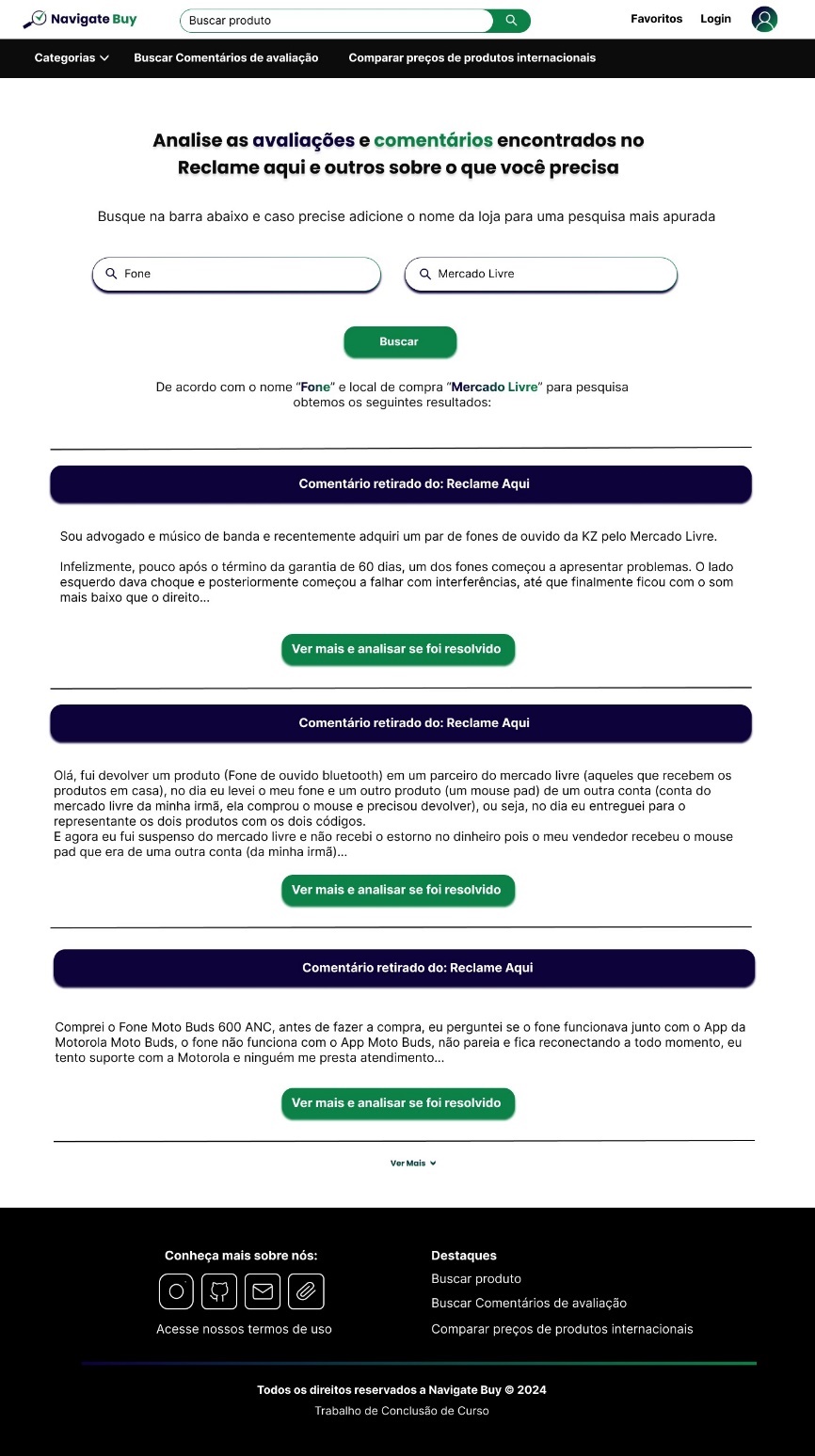
Figura 74 - Wireframe de baixa fidelidade da interface "Buscar Comentários de avaliação"

Texto

Descrição gerada automaticamente com confiança média

Fonte: Do próprio autor, 2024.

Figura 75 - Wireframe de alta fidelidade da interface "Buscar Comentários de avaliação"



Fonte: Do próprio autor, 2024.

Como última interface, as Figuras 76 e 77 retratam a tela de comparação de preços de produtos internacionais, que fornece a conversão monetária entre moedas em tempo real. Essa funcionalidade permite a análise de diferentes preços em diversos países, bem como o cálculo das taxas de importação e o caminho para entender como esses cálculos são realizados e o que são as taxas de importação. Além disso, a interface apresenta ideias de lojas internacionais, informando seus continentes e as moedas disponíveis em cada uma delas.

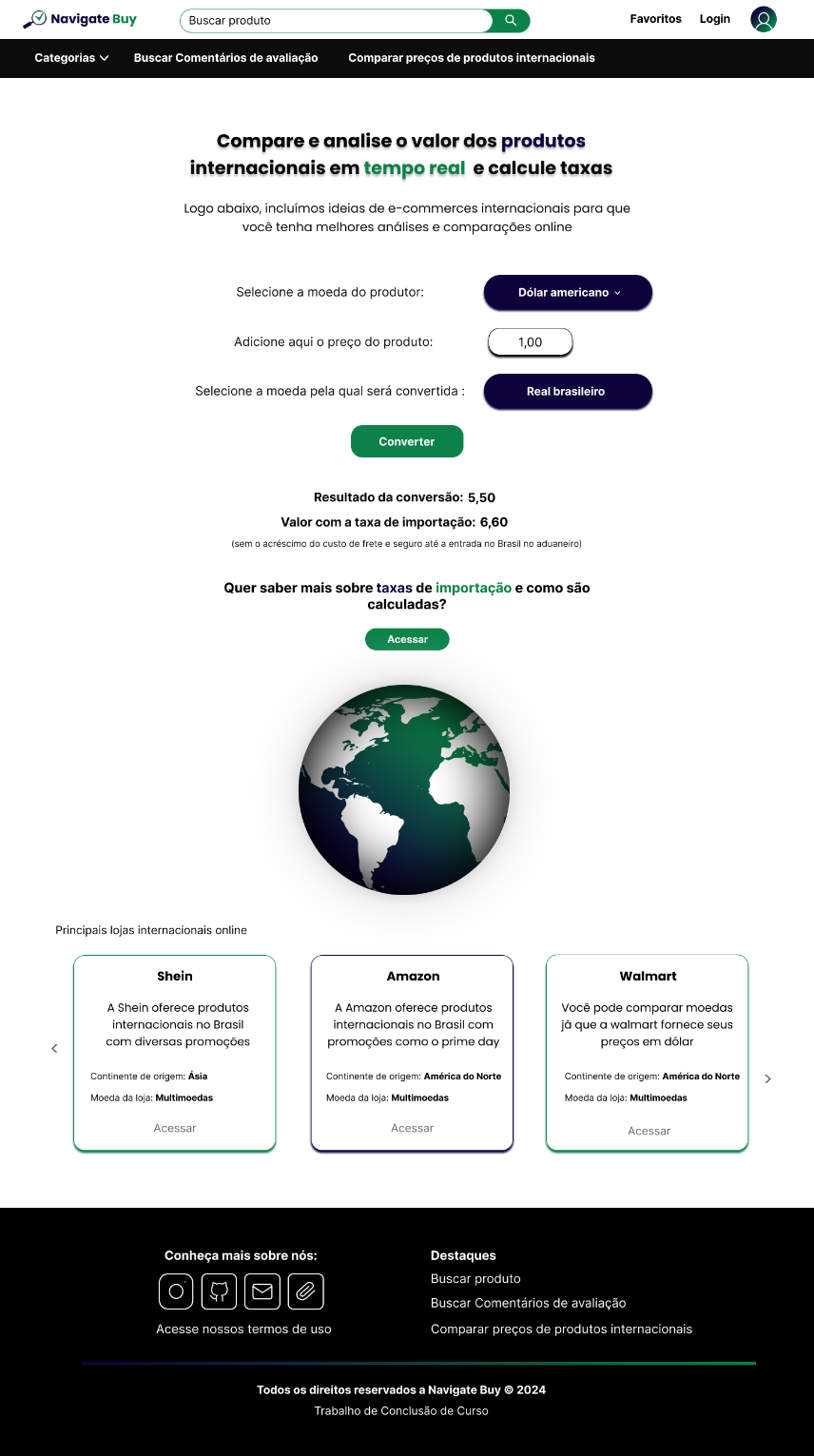
Figura 76 - Wireframe de baixa fidelidade da interface "Comparar preços de produtos internacionais"

Diagrama

Descrição gerada automaticamente

Fonte: Do próprio autor, 2024.

Figura 77 - Wireframe de alta fidelidade da interface "Comparar preços de produtos internacionais"



Fonte: Do próprio autor (2024)

# CONSIDERAÇÕES FINAIS

Após o desenvolvimento do projeto, ficou evidente a relevância do sistema web no cotidiano dos consumidores online, especialmente em um cenário onde o e-commerce cresce rapidamente, no qual os objetivos estabelecidos no começo desse estudo foram concluídos com êxito, em que sob esse viés oferece uma solução aos resultados da problemática encontrada, onde pesquisas mostraram que ainda há desafios na procura e comparação de melhores ofertas de diversas lojas verificadas, destacando a importância da implementação do sistema web como uma ferramenta crucial não apenas para projeções futuras, mas também para atender às demandas atuais dos consumidores, promovendo um crescimento contínuo no ambiente virtual.

Além disso, este projeto traz consigo diversas consequências positivas, como a otimização da experiência de compra e a simplificação na busca por preços mais competitivos, afetando diretamente a rotina dos consumidores. No entanto, é essencial considerar alguns obstáculos, como a necessidade de ampliar a variedade de lojas incluídas na análise e a rápida evolução das tecnologias disponíveis.

Portanto, este projeto contribui tanto para a esfera acadêmica quanto para o avanço nas compras online, facilitando a busca por melhores preços e promovendo inovação e infraestrutura digital, dessa forma, para investigações e implementações futuras, é recomendável explorar inovações tecnológicas que possam aprimorar a personalização das ofertas e a compreensão das preferências dos clientes na internet.

# REFERÊNCIAS

ADRIANO, Thiago Silva. **Guia prático de TypeScript:** Melhore suas aplicações JavaScript. São Paulo: Casa do Código Editora, 2021. E-book.

ALMEIDA, Pedro. **React hooks:** Um guia didático e prático para desenvolvimento web eficiente. 1 ed. Estados Unidos: Pedro de Almeida Alves Editora, 2023. E-book.

ALVES, William Pereira**. Construindo uma aplicação web completa com PHP e MySQL**. 1. ed. São Paulo: Novatec Editora, 2017. E-book.

ALVES, William Pereira. **Banco de dados**: Teoria e Desenvolvimento. 2. ed. São Paulo: Erica, 2021.

BASSETT, Lindsay. **Introdução ao JSON:** Um guia para JSON que vai direto ao ponto. 1 ed. São Paulo: Novatec Editora, 2015. E-book.

BENDORAITIS, Aidas; KRONIKA, Jake. **Desenvolvimento Web com Django 3 Cookbook:** Soluções práticas para problemas comuns no desenvolvimento Web com Python. 1 ed. São Paulo: Novatec Editora, 2020. E-book.

BOLONHA, Milton. **Introdução à programação:** Para iniciantes e agências web modernas. 1 ed. Brasil: Editora BOOK4DEV, 2024. E-book.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**: guia do usuário. 1. ed. Rio de Janeiro: Elsevier Editora, 2006.

BORGES, Luiz Eduardo. **Python para desenvolvedores**. 1 ed. São Paulo: Novatec Editora. 2014. E-book.

CARDOSO, Carlos. **HTML 4 - Série Curso Básico & Rápido**. 1 ed. Rio de Janeiro: Axcel Books, 1999.

CARDOSO, Virgínia; CARDOSO, Giselle. **Sistemas de banco de dados**: uma abordagem introdutória e aplicada. 1. ed. São Paulo: Saraiva, 2012.

DATE, C. J. **Introdução a sistemas de bancos de dados**. 8. ed. Rio de Janeiro: Elsevier Editora, 2004.

EIS, Davi.; FERREIRA, Elcio. **HTML5 e CSS3 com Farinha e Pimenta.** 1 ed. São Paulo: Tableless, 2012.

FLANAGAN, David. **JavaScript:** O guia definitivo. 6 ed. Porto Alegre: Bookman, 2013.

FOWLER, Martin. **UML Essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3. ed. [S.l]: Bookman Editora, 2005. E-book.

FREEMAN, Elisabeth; FREEMAN, Eric. **Use a Cabeça! HTML com CSS e XHTML**. 2. ed. Rio de Janeiro: Alta Books, 2006.

FURTADO, Daniel. **Princípios de UX**: Entendendo o Design Centrado no Usuário. 1. ed. Santa Catarina, Joinville: Brauer, 2023.

GAMMA, Erich. **Padrões de projeto:** Soluções reutilizáveis de software orientado a objetos. São Paulo: Bookman Editora, 2008. E-book

GRANT, Will. **UX Design**: Guia definitivo com as melhores práticas de UX. 1. ed. São Paulo: Novatec Editora, 2019.

GOLDBERG, Josh. **Aprendendo TypeScript:** Melhore suas habilidades de desenvolvimento web usando JavaScript Type-Safe. 1 ed. São Paulo: Novatec Editora, 2022. E-book.

GRINBERG, Miguel. **Desenvolvimento web com Flask:** Desenvolvendo aplicações web com Python. 1 ed. São Paulo: Novatec Editora. 2018. E-book.

GRONER, Loiane. **Estruturas de Dados e Algoritmos com JavaScript:** Escreva um Código JavaScript Complexo e Eficaz Usando a Mais Recente ECMAScript**.** 2 ed. São Paulo: Novatec Editora, 2019.

GUEDES, Gilleanes T. A. **UML 2 - Uma Abordagem Prática**. 3. ed. São Paulo: Novatec Editora, 2018.

HEUSER, Carlos Alberto**. Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009. E-book.

IHRIG, Colin J. **Pro Node.js para Desenvolvedores**. 1 ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2014.

JOBSTRAIBIZER, Flávia. **Criação de sites com CSS**. 1 ed. São Paulo: Digerati Books, 2009.

LEVY, Jaime. **Estratégia de UX**: Técnicas de estratégia de produto para criar soluções digitais inovadoras. 2. ed. São Paulo: Novatec Editora, 2021.

LOBO, Edson Junior Rodrigues. **Curso prático de MySQL**. 1. ed. São Paulo: Universo dos Livros, 2008. E-book.

MACHADO, Felipe Nery Rodrigues. **Banco de dados**: Projeto e Implementação. 4. ed. São Paulo: Erica, 2020.

MCKINNEY, Wes. **Python para análise de dados:** Tratamento de dados com Pandas, NumPy e IPython. 1 ed. São Paulo: Novatec Editora. 2018. E-book.

MEMÓRIA, Felipe. **Design para a internet:** Projetando a experiência perfeita. São Paulo: Elsevier Editora, 2006. E-book.

MENEZES, Nilo Ney Coutinho. **Introdução à programação com Python:** Algoritmos e lógicas de programação para iniciantes. 1 ed. São Paulo: Novatec Editora. 2010. E-book.

MILANI, André. **MySQL - Guia do Programador**. 1. ed. São Paulo: Novatec Editora, 2006.

MILETTO, Evandro Manara.; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de Software II: Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP**. Porto Alegre: Bookman, 2014.

MITCHELL, Ryan. **Web Scraping com Python**: Coletando mais dados da web moderna. 2. ed. Brasil: Novatec Editora, 2019.

MORAES, William B. **Construindo aplicações com NodeJS**. 3 ed. São Paulo: Novatec Editora, 2021.

MUNIZ, Antonio *et al.* **Jornada API na prática:** Unindo conceitos e experiências do Brasil para acelerar negócios com a tecnologia. 1 ed. Rio de Janeiro: Brasport Editora, 2023. E-Book.

PEREIRA, Caio R. **Node.js:** Aplicações web real-time com Node.js. 1 ed. São Paulo: Casa do Código. 2014. E-book.

PEREIRA, Eduardo. **Trilhas Python:** Programação multiparadigma e desenvolvimento Web com Flask. São Paulo: Casa do Código Editora, 2018. E-book.

PEREIRA, Rogério. **User experience design**. São Paulo: Casa do Código Editora, 2018. E-book.

PODMAJERSKY, Torrey. **Redação Estratégica para UX**: Aumente engajamento, conversão e retenção com cada palavra. 1. ed. São Paulo: Novatec, 2019.

POWERS, Shelley. **Aprendendo Node:** Usando JavaScript no servidor. 1 ed. São Paulo: Novatec Editora, 2017.

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

QUIERELLI, Davi A. **Criando Sites com Html-css-php:** Construindo um projeto iniciante. 1 ed. São Paulo: Clube de Autores, 2012.

SILVA, Maurício Samy. **Ajax com jQuery:** Requisições Ajax com a simplicidade de jQuery. 1 ed. São Paulo: Novatec Editora, 2009. E-book.

SILVA, Maurício Samy. **CSS3:** Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3. 1 ed. São Paulo: Novatec Editora, 2011.

SILVA, Tiago. **Flask de A a Z:** Crie aplicações web mais completas e robustas em Python. São Paulo: Casa do código Editora, 2019. E-book.

SILVA, Mauricio Samy. **React Aprenda Praticando:** Desenvolva aplicações web reais com uso da biblioteca React e de seus módulos auxiliares.1 ed. São Paulo: Novatec Editora, 2021. E-book.

SMITH, Bem. **JSON Básico:** Conheça o formato de dados preferido da web. 1 ed. São Paulo: Novatec Editora, 2015. E-book.

SOARES, Adriano Mesquita (org.). **Tópicos especiais em engenharia:** Inovações e avanços tecnológicos. 1 ed. São Paulo: AYA Editora, 2023. E-book.

SORDI, José Osvaldo de. **Modelagem de dados**: estudos de casos abrangentes da concepção lógica à implementação. 1. ed. São Paulo: Erica, 2019.

STEFANOV, Stoyan. **Padrões JavaScript**. 1 ed. São Paulo: Novatec Editora, 2011.

STEFANOV, Stoyan. **Primeiros passos com React:** Construindo aplicações web. 1 ed. São Paulo: Novatec Editora, 2016. E-book.

STEVENS, W. Richard. **Programação de rede Unix:** API para soquetes de rede. 3 ed. Porto Alegre: Bookman Editora, 2008. E-Book.

TEIXEIRA, Fabricio. **Introdução e boas práticas em UX Design**. São Paulo: Casa do Código Editora, 2014. E-book.

ZHAO, Alice. **SQL - Guia Prático**: um guia para o uso de SQL. 4. ed. São Paulo: Novatec Editora, 2023.

ASSIS, Wendel Vilaça de; GOMIDE, João Victor Boechat. Web scraping em dados públicos: método para extração de dados dos gastos públicos dos vereadores da Câmara Municipal de Belo Horizonte. **Informação & Informação**, Londrina, v. 26, n. 4, p. 319-341, out./dez. 2021. Disponível em: https://ojs.uel.br/revistas/uel/index.php/informacao/article/view/44123. Acesso em: 30 mai. 2024.

DIAS, Juan Pablo da Silva; HEMAIS, Marcus Wilcox. Consumidores de baixa renda e compras on-line: Receios em consumir pela internet. **REGE - Revista de Gestão**, São Paulo, v. 22, n. 2, p. 115-132, jan./mar. 2015. Disponível em: https://www.sciencedirect.com/science/article/pii/S1809227616300960. Acesso em: 03 mai. 2024.

FARIAS, Marcello Tenorio de; ANGELUCI, Alan César Belo; PASSARELLI, Brasilina. Web scraping e ciência de dados na pesquisa aplicada em comunicação: um estudo sobre avaliações online. **Revista Observatório**, Palmas, v. 7, n. 3, p. 1-22, jul./set. 2021. Disponível em: https://repositorio.usp.br/item/003093096. Acesso em: 29 mai. 2024.

JÚNIOR, João Batista da Silva; SILVA, Paulo Caetano. Análise da representação semântica de modelos de dados do formato JSON. **Revista de Sistemas e Computação**, Salvador, v.8, n.1, p. 196-209, jan./jun. 2018. Disponível em: https://www.researchgate.net/profile/Joao-Junior-43/publication/329525269\_Analise\_da\_representacao\_semantica\_de\_modelos\_de\_dados\_do\_formato\_JSON/links/5c0dbe3a4585157ac1b6ba05/Analise-da-representacao-semantica-de-modelos-de-dados-do-formato-JSON.pdf. Acesso em: 23 out. 2024.

MACHADO, Cristian Cleder *et al.* Um Web Crawler para Projeções e Análise de Vulnerabilidades de Segurança e Consistência Estrutural de Páginas Web. **Revista de Empreendedorismo, Inovação e Tecnologia**, [S.L.], v. 2, n. 2, p. 3-12, 2015. Disponível em: https://seer.atitus.edu.br/index.php/revistasi/article/view/869. Acesso em: 18 mai. 2024.

NASCIMENTO, João P. B.; CAPANEMA, Daniel de O.; PEREIRA, Adriano C.M. Projeto e análise de desempenho de um algoritmo iterativo para grandes grafos em um ambiente distribuído. **RBCA** **– Revista Brasileira de Computação Aplicada**, Minas Gerais, v. 11, n. 1, p. 36-47, mar. 2019. Disponível em: https://seer.upf.br/index.php/rbca/article/view/8738/114114519. Acesso em: 31 mai. 2024.

NETO, Henrique Carivaldo de Mirando *et al*. **UNIPAM -** **Anais COMINE**, Minas Gerais, v.2, p.7-80, out. 2022. Disponível em: https://anais.unipam.edu.br/index.php/comine/issue/view/50/57. Acesso em: 23 out. 2024.

RODRIGUES, Quemuel Baruque de Freitas et al*.* Webscraping em R: uma abordagem para investigação em ciências sociais. **Simbiótica**, Vitória, v. 8, n. 4, p. 191-215, set./dez. 2021. Disponível em: https://periodicos.ufes.br/simbiotica/article/view/37351. Acesso em: 31 mai. 2024.

SOUZA, Renato Rocha; CAFÉ, Lígia Maria Arruda. Análise de sentimento aplicada ao estudo de letras de música. **Informação & Sociedade**: Estudos, João Pessoa, v. 28, n. 3, p. 275-286, set./dez. 2018. Disponível em: https://periodicos.ufpb.br/index.php/ies/article/view/34884. Acesso em: 30 mai. 2024.

SOUZA, Francisco Moreira Calado; LIMA, Edilson Carlos Silva; CARIDADE, Elda Regina de Sena. Criando sistema escalável de agendamentos utilizando TypeScript com NestJS no BackEnd e NextJS no FrontEnd. **Revista Ibero - Americana de Humanidades, Ciências e Educação**, São Paulo, v.8, n.12, p.43-57, dez. 2022. Disponível em: https://periodicorease.pro.br/rease/article/view/7986/3136. Acesso em: 23 out. 2024.

ABBA, Ihechikara. **Como Usar o Tailwind CSS para Desenvolver Rapidamente os Sites da Snazzy**. 2023. Disponível em: https://kinsta.com/pt/blog/tailwind-css/. Acesso em: 23 abr. 2024.

BOLZANI, Isabela. **61% dos brasileiros compram mais pela internet do que em lojas físicas, aponta estudo**. G1globo, 2022. Disponível em: https://g1.globo.com/economia/noticia/2022/12/14/61percent-dos-brasileiros-compram-mais-pela-internet-do-que-em-lojas-fisicas-aponta-estudo.ghtml. Acesso em: 24 mar. 2024.

BUTEWICZ, Thanael. **Utilidades Python:** Obtendo conversão de moedas. Casa do desenvolvedor. 2022. Disponível em: https://forum.casadodesenvolvedor.com.br/topic/45430-utilidades-python-obtendo-conversão-de-moedas. Acesso em: 23 mai. 2024.

DIDÁTICA TECH. **Tutorial de Scrapy para iniciantes em Python**. 2022. Disponível em: https://didatica.tech/tutorial-de-scrapy-para-iniciantes-em-python/. Acesso em: 31 mai. 2024.

DUKE, Justin. **Como Fazer Crawling em uma Página Web com Scrapy e Python 3**. 2018. Disponível em: https://www.digitalocean.com/community/tutorials/como-fazer-crawling-em-uma-pagina-web-com-scrapy-e-python-3-pt. Acesso em: 26 mai. 2024.

E-COMMERCE BRASIL. **O impulso do e-commerce no Brasil**: perspectivas até 2027. 2024. Disponível em: https://www.ecommercebrasil.com.br/artigos/o-impulso-do-e-commerce-no-brasil-perspectivas-ate-2027. Acesso em: 04 jun. 2024.

FAUSTINO, Marco; LOBATO, Gisele. **Golpes virtuais crescem 65% com brasileiros passando mais tempo na internet**. Terra, 2023. Disponível em: https://www.terra.com.br/noticias/checamos/golpes-virtuais-crescem-65-com-brasileiros-passando-mais-tempo-na-internet,5b5a67f228cff03faf6084e2fc39eaa6oq7kkt7m.html. Acesso em: 28 mar. 2024.

GOMES, Ana Maria. **O que é Web Scraping e Como Utilizar com Python**. 2024. Disponível em: https://hub.asimov.academy/tutorial/o-que-e-web-scraping-e-como-utilizar-com-python/. Acesso em: 31 mai. 2024.

NEVES, Vinicios. **Composição elegante: Descubra o poder do Tailwind CSS**. 2023. Disponível em: https://marcosviniciosneves.medium.com/composi%C3%A7%C3%A3o-elegante-descubra-o-poder-do-tailwind-css-4aae5306bf91. Acesso em: 23 abr. 2024.

OLIVEIRA, Rafael. **Criando componentes com Tailwind CSS**.2023. Disponível em: https://www.treinaweb.com.br/blog/criando-componentes-com-tailwind-css. Acesso em: 23 abr. 2024.

SÉ, Letícia. **Nove** **em dez brasileiros pesquisam online antes de comprar, diz Google**. Globo, 2023. Disponível em: https://revistapegn.globo.com/negocios/noticia/2023/03/nove-em-dez-brasileiros-pesquisam-online-antes-de-comprar-diz-google.ghtml. Acesso em: 27 mar. 2024.