Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Sistema *web* para comparação dos preços de supermercados *online*

Marco Antônio Brandão Carvalho

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Rafael Frederico Alexandre

Outubro, 2022
João Monlevade–MG

Marco Antônio Brandão Carvalho

Sistema *web* para comparação dos preços de supermercados *online*

Orientador: Rafael Frederico Alexandre

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Outubro de 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

C331s Carvalho, Marco Antônio Brandão.
Sistema web para comparação dos preços de supermercados online. [manuscrito] / Marco Antônio Brandão Carvalho. - 2022.
56 f.: il.: color., tab..

Orientador: Prof. Dr. Rafael Frederico Alexandre.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Aplicações Web. 2. Controle de preços. 3. Sistemas de coleta automática de dados. 4. Sistemas de informação gerencial. 5. Supermercados - Comércio eletrônico. I. Alexandre, Rafael Frederico. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.775

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Marco Antônio Brandão Carvalho

Sistema Web para Comparação dos Preços de Supermercados Online

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel

Aprovada em 27 de outubro de 2022

Membros da banca

[Doutor] - Rafael Frederico Alexandre - Orientador (Universidade Federal de Ouro Preto)
[Doutor] - George Henrique Godim da Fonseca - (Universidade Federal de Ouro Preto)
[Doutor] - Fernando Bernardes de Oliveira - (Universidade Federal de Ouro Preto)

[Rafael Frederico Alexandre], orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 12/04/2023

Documento assinado eletronicamente por **Rafael Frederico Alexandre**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 12/04/2023, às 09:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0507310** e o código CRC **9887ACAD**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado à minha família e amigos, que sempre me apoiaram durante a vida acadêmica

Agradecimentos

Agradeço primeiramente aos meus pais, sem a ajuda e apoio deles eu não teria chegado onde estou hoje.

Agradeço aos meus amigos e professores da universidade que me acompanharam durante o curso.

Agradeço ao meu orientador, Rafael Frederico Alexandre, pelas oportunidades que recebi e por me guiar durante o desenvolvimento deste trabalho.

Finalmente, agradeço à todos aqueles que tive contanto e contribuíram para a minha formação acadêmica.

"Science is more than a body of knowledge; it is a way of thinking."

— Carl Sagan (1934 – 1996),
in: The Demon-Haunted World: Science as a Candle in the Dark.

Resumo

Este trabalho propõe o desenvolvimento de uma aplicação *web* que tem como objetivo principal a comparação de preço de listas de produtos encontrados em supermercados *online*. A principal motivação é a criação de uma ferramenta na qual o usuário consiga criar uma lista de produtos, que posteriormente é utilizada pela aplicação para direcioná-lo ao melhor supermercado para fazer suas compras. Existem ferramentas consolidadas no mercado com funções semelhantes, como por exemplo o comparador de preços Zoom, contudo não é possível criar uma lista de produtos que indique o melhor local de compra. Os dados de produtos dos supermercados virtuais são obtidos de forma automatizada por um robô, que percorre os sistemas públicos dos supermercados *online*. O sistema proposto permite que o usuário procure e aplique filtros de pesquisa para encontrar produtos, além de saber o melhor preço praticado para determinado produto, junto do seu histórico de preços. A aplicação permite que o usuário adicione itens à sua lista de produtos desejados de tal forma que torna possível encontrar o melhor supermercado para realizar a compra. As tecnologias utilizadas para construção da interface *web*, API RESTful e o robô foram, respectivamente, o *framework* Angular, Express e Node.js, além do banco de dados PostgreSQL. Os resultados obtidos apontam para a viabilidade do sistema proposto, visto que o usuário é capaz de identificar os melhores supermercados para sua compra dentro da aplicação construída.

Palavras-chaves: supermercado *online*. sistema *web*. comparação de preço. *shopbot*

Abstract

This work proposes the development of a web application whose main objective is to compare the price of product lists found in online supermarkets.The main motivation is the creation of a tool in which the user can create a list of products, which is later used by the application to direct him to the best supermarket to do his shopping. There are consolidated tools on the market with similar functions, such as Zoom the price comparator, however it is not possible to create a product list that indicates the best place to buy. Product data from virtual supermarkets is automatically obtained by a robot, which runs through the public systems of online supermarkets. The proposed system allows the user to search and apply search filters to find products, in addition to knowing the best price for a particular product, along with its price history. The application allows the user to add items to their wish list in such a way that makes it possible to find the best supermarket to make the purchase. The technologies used to build the web interface, RESTful API and the robot were, respectively, the Angular, Express and Node.js framework, in addition to the PostgreSQL database. The results obtained point to the viability of the proposed system, since the user is able to identify the best supermarkets for their purchase within the built application.

Key-words: online supermarket. web application. price comparison. shopbot.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABComm *Associação Brasileira de comércio Eletrônico*

API *Application Programming Interface*

CEP Código de Endereçamento Postal

COVID-19 *Coronavirus Disease 2019*

CRUD *Create, Read, Update and Delete*

CSS *Cascading Style Sheets*

E-commerce Comércio Eletrônico

GTIN *Global Trade Item Number*

HTML *HyperText Markup Language*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hyper Text Transfer Protocol Secure*

JWT *JSON Web Token*

MPMG Ministério Público de Minas Gerais

MVC *Model View Controller*

REST *Representational State Transfer*

RFC *Request for Comments*

SDK *Software Development Kit*

SGBD Sistema de Gerenciamento de Banco de Dados

SKU *Stock Keeping Unit*

SPA *Single Page Application*

WSL Subsistema Windows para Linux

Lista de símbolos

$\leftarrow$	Recebe
$\in$	Pertence

Sumário

1 Introdução

O Comércio Eletrônico (E-commerce) é responsável por uma parcela importante e crescente do mercado brasileiro. O número de pedidos e compradores que realizaram ao menos uma compra *online* cresceram, respectivamente, de 53,7 milhões para 111,2 milhões entre os anos de 2011 e 2017, e 31,2 milhões para 55,2 milhões entre os anos de 2013 e 2017 (GUISSONI; FARINHA, 2019).

O fator de crescimento se acentuou durante o inicio da pandemia da *Coronavirus Disease 2019* (COVID-19) em 2020, no mês de abril. Foi um crescimento de 81% nas compras *online* quando comparado ao mesmo período no ano anterior (SILVA et al., 2021). Segundo Costa et al. (2021), diversas empresas e comércios passaram por uma etapa de adaptação para oferecer serviços no ambiente digital, ao mesmo tempo que surgia um novo grupo de consumidores. O uso desta abordagem pode ser beneficiária para as empresas, não apenas no sentido de abrangência, mas no corte de custos, podendo melhorar a qualidade dos produtos e serviços, além de entregá-los mais rápido ao consumidor (GOEL, 2007). O *e-commerce* deixa de ser apenas um método de compra alternativo e passa a ser, de certa maneira, uma solução que se torna utilizada amplamente.

Em diferentes seguimentos o impacto da pandemia foi sentido de forma distinta. Dentre eles destaca-se o setor de supermercados, que representa um mercado essencial para o cotidiano e que teve um crescimento de 270,16% durante o inicio da pandemia, de acordo com a *Associação Brasileira de comércio Eletrônico* (ABComm) (SILVA et al., 2021). Os serviços vitais não fecharam as portas durante este período, apesar disso, o consumidor buscou o comércio eletrônico. A mudança no comportamento do consumidor vem acontecendo desde antes da pandemia da COVID-19, como demostrando no estudo de Guissoni e Farinha (2019) publicado no início de 2019, porém no contexto da pandemia, existe um grande interesse em evitar o contato com ambientes onde existe um fluxo de pessoas elevado.

O cenário econômico do país também pode influenciar diretamente no comportamento do consumidor. Durante o período da crise econômica, entre os anos de 2014 e 2016, o perfil do consumidor passou por alterações

> ...mais seletivo e exigente, buscando a redução e corte de gastos, seja optando por produtos mais baratos ou cortando gastos considerados supérfluos. Nesse contexto, observou-se que em época de crise, o consumidor de supermercados teve que tomar atitudes como: fazer suas compras com maior cautela, não agir por impulso na decisão de compra, tomar cuidado ao usar os cartões de crédito, evitar parcelamentos, analisar qual o real valor que o produto terá em sua vida e comparar os preços e o custo-benefício antes de qualquer decisão (SALES, 2017).

Devido a crescente frente do *e-commerce*, o consumidor *online* dispõe de um aumento gradativo de opções para escolha de seus produtos na hora de realizar sua compra. Neste cenário, começam a surgir ferramentas que auxiliam na comparação de preços entre produtos. Para itens do segmento eletrônico, por exemplo, existem diversos comparadores de preços como o caso do Zoom[1], do qual este trabalho utiliza como fonte de inspiração.

Os *sites* de comparação de preço, também conhecidos como *"shopbots"*, permitem ao usuário ter um acesso rápido à comparação de preço dos produtos desejados (GAO et al., 2017). Este tipo de recurso, facilita a busca do consumidor por lojas que oferecem preços menores e estimula a competitividade de preços (BAKOS, 2001). Obter mais informações sobre os preços de produtos, aumenta a sensibilidade de preço, um princípio que determina quanto o consumidor tem interesse em pagar pelo produto (DEGERATU; RANGASWAMY; WU, 2000).

Visto os fatores pontuados anteriormente, este trabalho propõe o desenvolvimento de um sistema *web* que busca explorar a comparação dos preços, com um foco no seguimento de supermercados *online*, criando uma ferramenta que possa ser utilizada para pesquisar preços e ajudar na tomada de decisão do consumidor. Nas próximas seções, serão discutidos o problema de pesquisa, os objetivos e a metodologia utilizada para realização do trabalho.

1.1 Objetivos

Existem diversas ferramentas que realizam a comparação de preços entre produtos de diferentes *sites* da Internet. Ao restringir o escopo apenas para os que realizam a comparação de produtos de supermercados, o número de opções diminui drasticamente. Um outro fator importante é a disponibilidade de dados dos produtos sobre os supermercados de uma determinada região ou cidade.

No contexto da cidade de Belo Horizonte, Minas Gerais, foi notado que não existe um *site* ou aplicativo que auxilie na comparação de preço das compras *online* sobre os supermercados da cidade. Páginas *web* como Submarino[2] e Americanas[3], que não têm um foco em produtos consumíveis, chegam a oferecê-los por meio de lojas parceiras que possuem restrições por região e a entrega pode demorar vários dias. Esses registros são inseridos em comparadores preços, porém não refletem a situação dos supermercados em localidades especificas.

O presente trabalho consiste em desenvolver uma aplicação *web* que permita ao usuário final criar sua própria lista de compras, e fazer a comparação de diversos produtos disponíveis nos supermercados da região de Belo Horizonte. É possível listar os seguintes

[1] Disponível em: <https://www.zoom.com.br/>
[2] Disponível em: <https://www.submarino.com.br/>
[3] Disponível em: <https://www.americanas.com.br/>

objetivos específicos:

- Desenvolver uma aplicação web que liste e compare os produtos de cada supermercado da região.
- Consumir *Application Programming Interface*s (APIs) para obter dados periodicamente.
- Armazenar e identificar produtos em bancos de dados.
- Gerar históricos de preço para cada produto.
- Indicar ao usuário final, qual o melhor supermercado para comprar os itens de uma lista de compras.

1.2 Metodologia

Os passos para execução deste trabalho são assim definidos:

- Revisão da literatura: estudar recursos similares e trabalhos correlatos para entender as principais implicações e necessidades.
- Definir o ambiente: definição da linguagem de programação, plataforma de banco de dados e arquitetura geral do projeto.
- Estudar as tecnologias: instalação e entendimento do funcionamento, bem como os seus limites, com objetivo de facilitar o desenvolvimento.
- Desenvolver o *software*: desenvolvimento do código e do banco de dados, conforme as necessidades observadas inicialmente.
- Analisar os resultados: verificar se o que foi desenvolvido cumpre a proposta definida, além de observar possíveis melhorias.

1.3 Organização do trabalho

O restante deste trabalho é organizado como se segue. O Capítulo 2 apresenta o estudo de trabalhos e ferramentas existentes, relacionados a este artigo. No Capítulo 3 é detalhado como foi o processo de desenvolvimento do *software*. Já no Capítulo 4 são apresentados os resultados obtidos e, por fim, o Capítulo 5 traz as considerações finais e propostas para trabalhos futuros.

2 Trabalhos Correlatos

Nesta seção são apresentados trabalhos e ferramentas existentes, que comparam preços ou listam produtos, para analisar as principais características destas aplicações, afim de refinar a visão sobre o *software* que será desenvolvido.

2.1 Revisão bibliográfica

Os trabalhos presentes na literatura, dentro da área de comparadores de preço, abordam uma série de cenários e características, que tem uma forte correlação com o tema deste artigo. Esta seção é destinada a um estudo aprofundado desses trabalhos, para que seja possível ligar os principais pontos em comum, e construir uma ideia mais refinada para guiar o projeto.

Os comparadores de preços podem ser definidos como um serviço *online* utilizado por compradores, quando estão a procura de informações sobre um produto, originadas de vários sites e vendedores diferentes (WAN, 2009). De maneira similar, Laghmari et al. (2019) definem como um serviço *online* centralizado em um *website*, que com base na pesquisa por um produto ou serviço, estabelece uma lista, fazendo a identificação de determinados fatores dos produtos ou serviços, ofertados por várias plataformas de *e-commerce*.

Em um estudo recente, Fernandes (2022) faz a caracterização da tendência de funcionamento dos *shopbots*: possuem uma caixa de pesquisa, onde o usuário pode inserir as palavras-chaves referentes ao produto desejado, e em seguida, o site retorna uma lista dos produtos que se encaixam nos termos pesquisados. Além disso, o usuário também pode aplicar filtros que julgar pertinentes. Segundo a autora, esse fluxo de funcionamento se mostra presente em quase todos os 163 *websites* abordados em seu trabalho.

No trabalho de Paula (2018) é mostrado o desenvolvimento de uma aplicação móvel que auxilia o usuário à realizar compras de supermercado. Foi construído um sistema cujo objetivo era permitir aos usuários criar listas de produtos e, a partir delas, serem direcionados ao melhor estabelecimento para realizar suas compras. O trabalho da autora compartilha de um objetivo principal similar ao apresentado por este texto. Um dos destaques levantados por Paula (2018) está no modelo colaborativo do seu sistema, isto é, na inserção de dados dos produtos feita pelos próprios usuários.

A dinâmica introduzida pelos comparadores depende, de maneira intrínseca, de uma constante entrada de dados, para que as informações dispostas sejam úteis. Quando os preços apresentados pelo *shopbot*, diferem dos preços reais no site do vendedor, ocorre

o fenômeno da dispersão de preços (BRYNJOLFSSON; SMITH, 2000). A fidelidade dos dados está relacionada à frequência de atualização de preços, tanto dos sites contemplados pelos rastreadores, quanto pela frequência de execução do serviço responsável pela raspagem dos dados. No sistema projetado por Rathod, Pavate e Patil (2018), o processo de coleta ocorre à cada hora, enquanto Alam et al. (2020) programou esta tarefa para ocorrer a cada doze horas. Essa definição, tem implicações voltadas para o poder de processamento da máquina e abrangência das coletas, além de impactar na credibilidade da informação mostrada ao usuário.

Os comércios *online*, que serão alvo da coleta de dados, são escolhidos com base em fatores como diversidade de produtos, popularidade e confiança (ALAM et al., 2020). A confiança, é um dos componentes mais importantes para qualquer mercado *online* segundo Brynjolfsson e Smith (2000), estando diretamente ligada à imagem da empresa. Esses princípios são empregados por comparadores como o Zoom[1].

No estudo de Degeratu, Rangaswamy e Wu (2000), sobre o comportamento do consumidor entre lojas *onlines* e físicas, os atributos de pesquisa são divididos em quatro categorias: nome da marca, preço, atributos sensoriais e não-sensoriais. Segundo os autores, atributos sensoriais, isto é, características do produto que podem ser percebidas pelos sentidos do ser humano como visão, tato ou olfato, possuem menos impacto na escolha de produtos *online*. Um outro aspecto trabalhado por Smith e Brynjolfsson (2001), diz sobre o impacto de características não relacionadas ao produto. O serviço de compra e entrega, pode contar com custos extras ou ser mais rápido, dependendo da empresa prestadora. Com exceção dos atributos sensoriais, que são direcionados ao comércio físico, é possível categorizar e quantificar esses dados, como possíveis termos de pesquisa para o usuário.

O processo de coleta ou raspagem de dados pode ocorrer de diferentes formas. No trabalho de Laghmari et al. (2019), são apontadas quatro técnicas que possuem esta finalidade: *Data Feeding*, *Data Wrapping*, *Affiliate Feeds* e recuperação de dados por meio de *meta-engines*. A primeira e segunda abordagem, são bem detalhadas por Wan (2009) e, consistem respectivamente de, o próprio vendedor inserir os dados do seu catálogo no sistema comparador, enquanto a outra, é sobre extrair os dados de forma manual ou automática, diretamente de uma página da *web*, transformando-os em um formato consistente. A terceira maneira, resume-se em obter os dados em tempo real, por meio de uma API fornecida pelo vendedor, e por fim, a utilização de *meta-engines* corresponde a obter dados por meio dos próprios comparadores e motores de pesquisa (LAGHMARI et al., 2019).

Dentre as estratégias aplicadas na coleta de dados, se destaca o uso de *web crawling* e *web scraping* em conjunto, como aplicado no trabalho de (MEHAK et al., 2019). Este procedimento pode ser dividido em duas etapas, adquirir os recursos e extrair as informações

[1] Disponível em: <https://www.zoom.com.br/conheca-o-zoom/>

úteis (ZHAO, 2017). O processo de *crawling* é responsável por encontrar as páginas da *web*, enquanto o *scraping*, ou raspagem, ocorre quando é feita a extração de dados das páginas encontradas.

Nos trabalhos de Alam et al. (2020) e Rathod, Pavate e Patil (2018), foram construídos *web crawlers*, que percorrem os sites e fazem a raspagem de dados, posteriormente armazenando-os em um banco de dados. Essa forma de obtenção de dados se encaixa no modelo de *Data Wrapping* discutido anteriormente, fazendo o uso de técnicas de *web crawling* e *web scraping*. O processamento pode contar com etapas adicionais, como o caso de Rathod, Pavate e Patil (2018), onde foi implementado um algoritmo para classificar produtos com base na popularidade e menor preço, permitindo a listagem dos resultados mais relevantes primeiro.

2.2 Ferramentas e aplicativos

É possível encontrar uma grande variedade de aplicativos e *sites*, que permitem a listagem e comparação de produtos das mais diversas categorias. Dentro desse contexto, esta seção apresenta a análise de cinco *sites* e aplicativos móveis, inseridos dentro do cenário brasileiro, a fim de entender e comparar suas principais características e funcionalidades.

2.2.1 Cornershop

O Cornershop[2] é um serviço de *delivery* controlado pela Uber, que conta com uma aplicação *web* e *mobile*. Na plataforma, os usuários podem montar uma lista de compras, com produtos em vários estabelecimentos cadastrados. Ao finalizar o pedido, um dos funcionários da empresa realiza a sua compra fisicamente nas lojas correspondentes, fazendo a entrega dos produtos no mesmo dia.

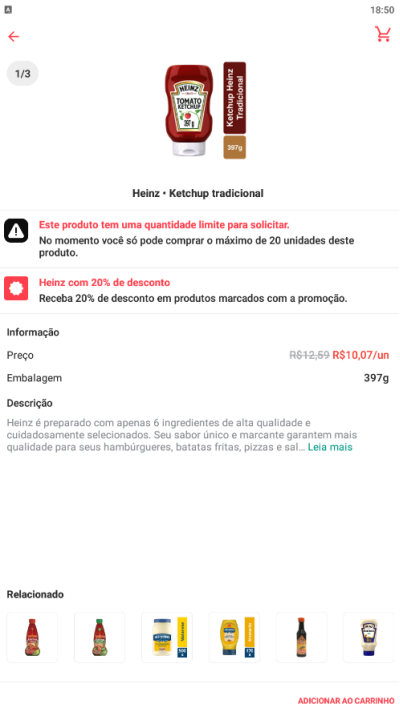
Por contar com diversos estabelecimentos físicos, o catálogo da plataforma é bastante extenso. Tanto o aplicativo quando o *site*, permitem visualizar dados como preço, descrição, disponibilidade, promoções e características dos produtos, como mostrado na Figura 1. Apesar de contar com bastante informações sobre lojas e supermercados, a plataforma apenas expõe os dados dos produtos, não contando com ferramentas para comparar propriamente os preços entre lojas.

2.2.2 Mercado Mineiro

O *site* Mercado Mineiro[3] oferece uma ferramenta de comparação de preços entre produtos na região metropolitana de Belo Horizonte. Segundo eles, a fonte de dados é

[2] Disponível em: <https://cornershopapp.com/>
[3] Disponível em: <https://mercadomineiro.com.br/>

Figura 1 – Cornershop



Fonte: Elaborado pelo autor

por meio de pesquisas *in loco*, ou seja, no próprio local, e também pelo fornecimento dos próprios estabelecimentos. Além de produtos consumíveis, também são listadas categorias como combustíveis e serviços variados.

Os usuários podem pesquisar por nomes de produtos ou por categorias pré-definidas. A Figura 2 ilustra a tela apresentada ao realizar uma pesquisa. Existe uma seção de ofertas que exibe dados cadastrados por usuários por meio de um aplicativo parceiro detalhado na Seção 2.2.3, contando com imagens do local da oferta. O *site* também conta com o apoio do Ministério Público de Minas Gerais (MPMG) para realizar suas operações.

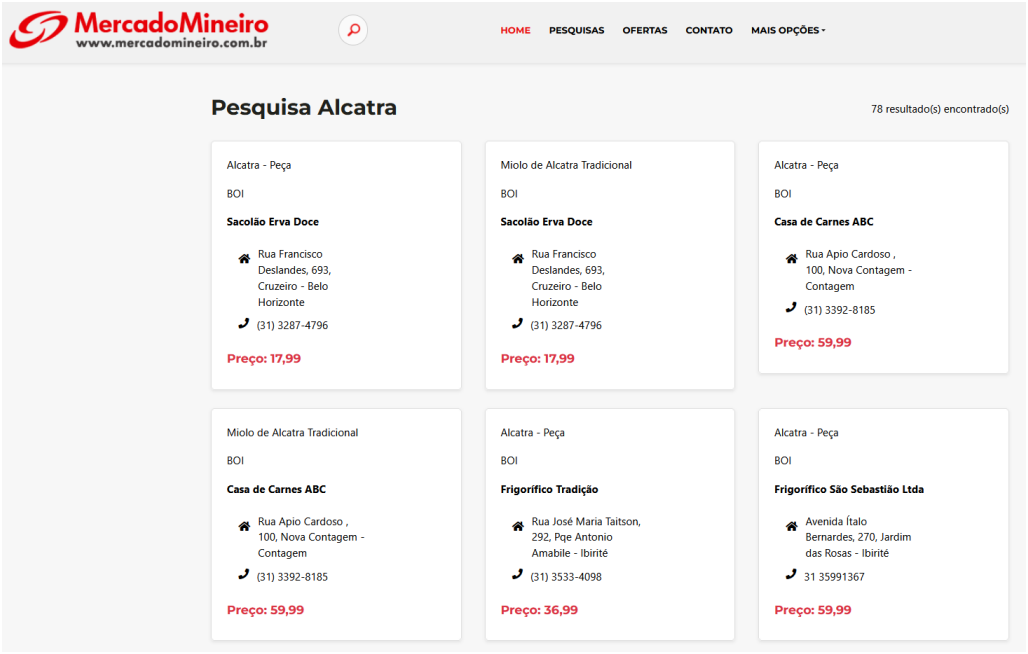
2.2.3 comOferta

O aplicativo comOferta[4], disponível para dispositivos Android e iOS, permite que usuários cadastrem ofertas que encontram nos estabelecimentos. Todos os usuários conseguem visualizar as ofertas cadastradas para sua região.

Para acessar o sistema é necessário realizar cadastro, também é requisitada a localização do usuário para oferecerem ofertas relevantes. A entrada de dados é exclusivamente em *in loco*, contando com a imagem do produto mostrando a oferta no local, bem como

[4] Disponível em: <https://play.google.com/store/apps/details?id=br.com.comoferta>

Figura 2 – Mercado Mineiro



Fonte: Elaborado pelo autor

endereço e informações de contato. Além disso, possui um histórico de preço para cada produto listado, como mostrado na Figura 3.

2.2.4 Zoom

A plataforma do Zoom[5] engloba diversos produtos de lojas e revendedores selecionados, podendo servir como vitrine ou mediar a compra de produtos *online*. Apesar de não ter foco em produtos consumíveis, é possível encontra-los no *site* dentre as vastas categorias.

Uma das principais características dessa ferramenta é a abrangência e o histórico de preço, como mostrado na Figura 4. O usuário consegue visualizar a diferença de preço entre várias lojas e as flutuações dos valores anteriores, em um período de até seis meses. Juntamente do sistema web, O Zoom conta com um aplicativo móvel próprio que dispõe das mesmas funcionalidades.

> Além de poder analisar preços a partir das buscas e comparações, o aplicativo permite que o usuário salve produtos pelos quais apresentou algum interesse e crie alertas baseados no histórico de preços. Ainda é possível acessar descrições mais especificas do item, avaliações de outros clientes e opinião de especialistas sobre as qualidades e vantagens técnicas. (ARAÚJO et al., 2016).

[5] Disponível em: <https://www.zoom.com.br/>

Figura 3 – comOferta

Fonte: Elaborado pelo autor

Em 2019, a empresa adquiriu uma de suas principais concorrentes, a Buscapé, mas que ainda manteve a sua marca G1 (2022).

2.2.5 Google Shopping

O Google Shopping[6], é uma ferramenta da empresa Google, que oferece uma visualização do preço de produtos, em diversas lojas *online*. Por estar integrada no buscador da Google, faz um uso prático de seus recursos durante uma pesquisa.

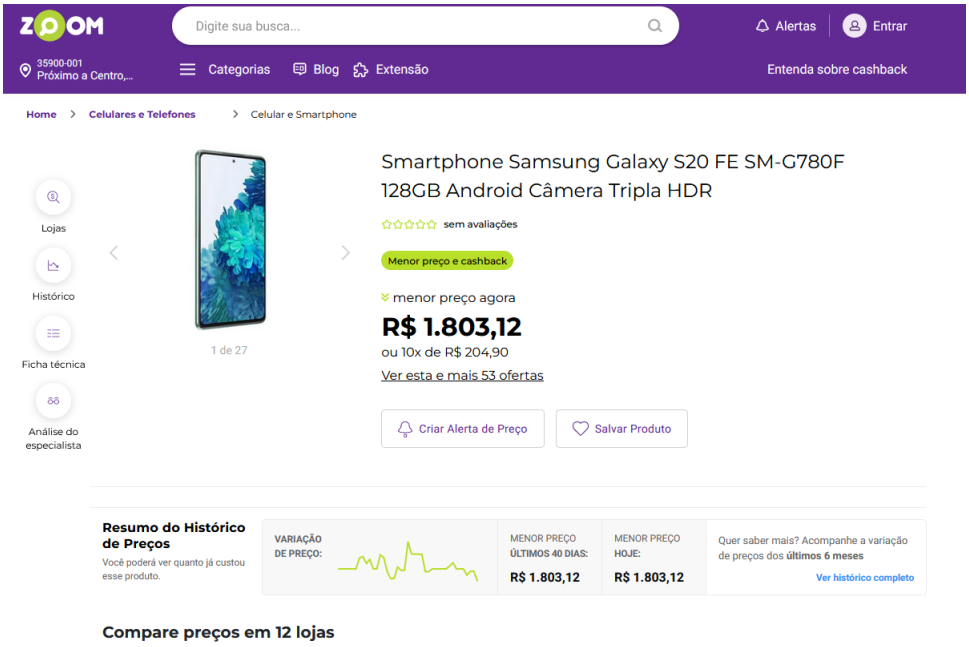
A plataforma permite uma pesquisa com filtros, mostrando várias opções de produtos. Na página do item, mostrada na Figura 5, são informados os detalhes do produto e os preços em cada *website*, além de também exibir as avaliações contidas nestes *sites*. A entrada principal de dados vem diretamente dos vendedores e de outros comparadores de preço.

2.3 Síntese

As características e aspectos discutidos, durante a revisão da literatura na Seção 2.1, podem ser observadas nos mais diversos *sites* de comparação e venda de produtos.

[6] Disponível em: <https://shopping.google.com.br/>

Figura 4 – Zoom



Fonte: Elaborado pelo autor

Neste sentido, foram elaboradas a Tabela 1 e Tabela 2, que fazem a listagem dessas características, para cada um dos aplicativos examinados na Seção 2.2.

Tabela 1 – Características dos sistemas estudados.

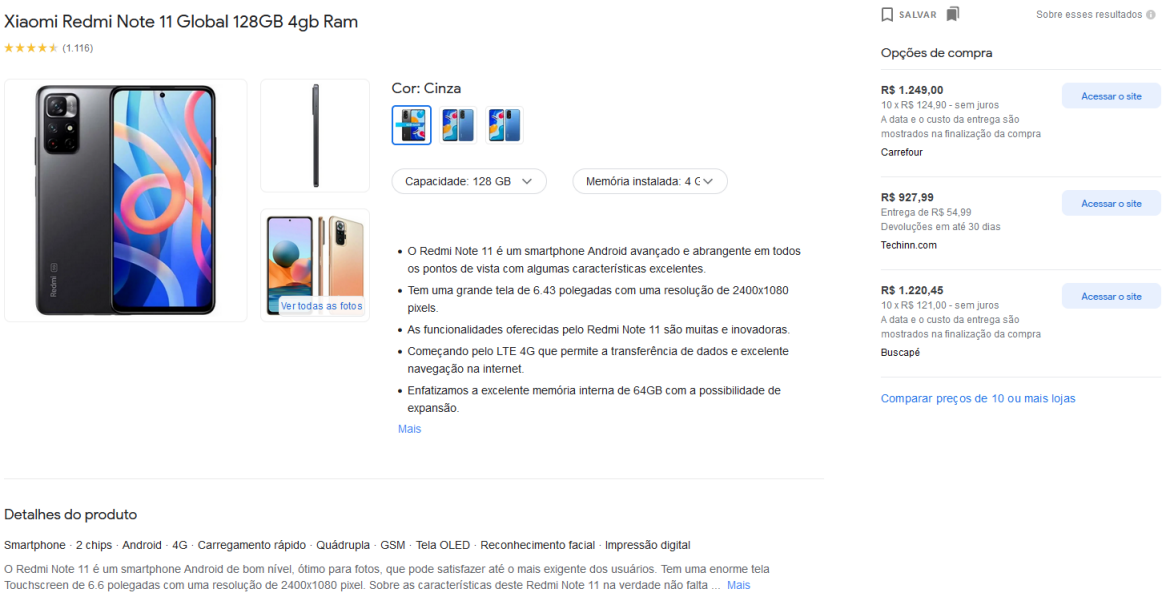
	Comparação de preço	Plataforma	Especialização	Promoções	Avaliações
Cornershop	Não	Híbrida	Supermercado	Sim	Não
Mercado Mineiro	Sim	*Web*	Supermercado	Sim	Não
comOferta	Sim	*Mobile*	Supermercado	Sim	Não
Zoom	Sim	Híbrida	Geral	Sim	Sim
Google Shopping	Sim	*Web*	Geral	Sim	Sim
Este trabalho	Sim	*Web*	Supermercado	Não	Não

Fonte: Elaborado pelo autor

Durante a análise, foram considerados casos onde o sistema abrangia vários produtos de diversas origens. Por causa desse fator, o aplicativo analisado não precisa necessariamente, ser um comparador de preço, como é o caso do Cornershop. Uma outra característica observada, foi a presença de abordagens hibridas, onde o sistema possui uma versão *web* para *desktop*, e também conta com um aplicativo móvel para a mesma aplicação.

Foram ainda, destacadas características que representam ações disponíveis para o usuário, dentro do ambiente do aplicativo que aumentam a praticidade de uso. A possibilidade de aplicar filtros na pesquisa, criar e salvar uma lista de compras própria,

Figura 5 – Google Shopping



Fonte: Elaborado pelo autor

Tabela 2 – Características dos sistemas estudados (continuação).

	Negócio	Venda de produtos	Filtros	Criar lista	Alertas de preço
Cornershop	Loja física	Sim	Sim	Sim	Não
Mercado Mineiro	Loja física	Não	Sim	Não	Não
comOferta	Loja física	Não	Sim	Não	Não
Zoom	Loja virtual	Sim	Sim	Não	Sim
Google Shopping	Loja virtual	Não	Sim	Sim	Não
Este trabalho	Loja virtual	Não	Sim	Sim	Não

Fonte: Elaborado pelo autor

criar um alerta de preço, para quando determinado produto ficar abaixo de um preço pré-estabelecido, são recursos interessantes para o tipo de aplicação analisado.

Devido ao foco voltado para supermercados deste trabalho, buscou-se explorar soluções que se especializassem no mesmo setor, porém isso não foi colocado como um fator limitante. Além disso, também se fez a separação de aplicativos que trabalham com o comércio em loja física e em loja virtual, visto que existem diferentes implicações para cada tipo de negócio.

Algumas ferramentas realizam apenas a listagem dos produtos, encaminhando o usuário para o *site* do vendedor, enquanto outras são capazes de fazer a mediação da compra com a loja original. No caso do comparador Zoom, foi observado que apenas certos

artigos contavam com a opção de compra dentro do *site*.

Não foram incluídas na tabela, informações a respeito dos métodos de obtenção de dados, discutidos na Seção 2.1, pois em diversos casos, não foi possível confirmar a metodologia utilizada. Apenas pela observação das funcionalidades de um sistema, não é possível determinar se a coleta é feita por *Data Wrapping* ou *Affiliate Feeds*. Visto a proposta e limitações do presente trabalho, uma das abordagens mais interessantes para ser explorada, é a baseada em *Data Wrapping* para coleta automática de dados.

Em relação à aplicação proposta por este trabalho, se buscou cobrir as principais funcionalidades para garantir o funcionamento do sistema. Recursos como mostrar promoções e criar alertas de preço, podem ser adicionadas em trabalhos futuros. O principal diferencial está na capacidade de comparar preços, juntamente de permitir criar uma lista de produtos.

3 Desenvolvimento

Este capítulo descreve o desenvolvimento do trabalho seguindo as etapas de arquitetura da aplicação, levantamento de requisitos, pesquisa de supermercados da região, bem como características da coleta de dados e a modelagem do banco de dados. Em seguida, se segue para o desenvolvimento do robô, que será responsável por obter os dados de entrada do sistema, o desenvolvimento da API de *backend* e do *frontend web* da aplicação.

3.1 Arquitetura da aplicação proposta

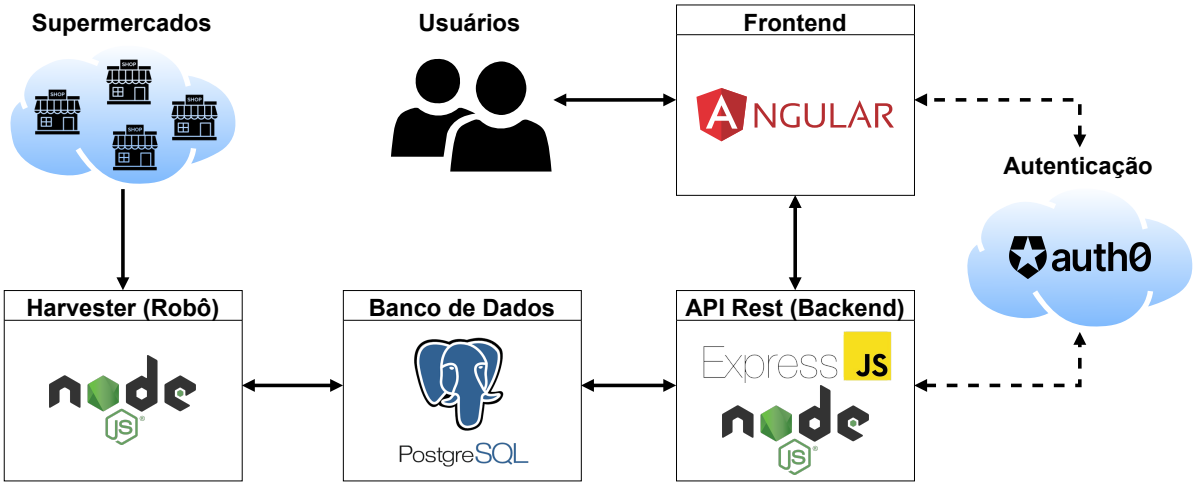
A revisão bibliográfica e análise de ferramentas comparadoras, feitas no Capítulo 2, permitiu um maior refinamento do projeto proposto por este trabalho. Com base no material estudado, foi escolhido seguir a abordagem para a construção de um *site*, onde o usuário poderá pesquisar por produtos e comparar preços.

A arquitetura proposta é composta pela implementação de sistema *web*, que segue o modelo cliente-servidor. Segundo Sommerville (2019), esta arquitetura é representada por um conjunto de serviços, onde cada serviço é fornecido por um servidor separado, a comunicação entre clientes e o servidor é feita por protocolos de requisição-resposta, como por exemplo o *Hypertext Transfer Protocol* (HTTP). Essa divisão permite ao sistema ser facilmente escalável.

O desenvolvimento do sistema foi feito utilizando as tecnologias descritas no Apêndice A. As ferramentas listadas compõem os recursos necessários para a construção de um ambiente *web*, capaz de fornecer os serviços e as funcionalidades exigidas, para a operação plena do *site*. É proposta uma coleta de dados periódica, junto com um armazenamento de dados robusto e a construção de uma API *RESTful*, isto é, que segue os conceitos do modelo *Representational State Transfer* (REST).

A Figura 6 ilustra a arquitetura geral do projeto, levando em consideração as tecnologias mencionadas anteriormente. O *frontend* é o *site* onde o usuário interage com o sistema, enquanto a API do *backend* trata e responde as requisições. O serviço de autenticação, cria a sessão do usuário no *site*, e valida a autenticidade das requisições rebebidas pela API. O robô é responsável pela coleta de dados dos supermercados, percorrendo *sites* e consumindo APIs da Internet. O banco de dados fornece ao robô informações como lista de empresas, e ao *backend* o histórico dos produtos, além de armazenar os dados enviados por ambos.

Figura 6 – Arquitetura geral da aplicação



Fonte: Elaborado pelo autor

3.2 Levantamento de Requisitos

A análise dos requisitos de um sistema são o ponto de partida para descrever as necessidades dos usuários e funcionalidades que o sistema deve possuir, para Sommerville (2019), a engenharia de requisitos é o primeiro estágio no desenvolvimento de software. Logo, o principal objetivo desta análise é nortear o que precisa ser implementado, tanto em questão de código quanto na construção do banco de dados.

Inicialmente, se fez uma análise das principais funcionalidades utilizadas pelas aplicações citadas no Capítulo 2, além disso, se idealizou quais outras funcionalidades são interessantes de serem inseridas neste contexto. A Tabela 3 lista as funcionalidades disponíveis para todos os usuários que acessam o sistema *web*.

Tabela 3 – Funcionalidades do usuário *guest*.

Funcionalidade	**Descrição**
Visualizar produtos	Ver informações sobre os produtos registrados no sistema
Pesquisar produtos	Pesquisar, ordenar e filtrar produtos
Visualizar histórico de grupo	Ver as alterações de preço de um grupo de produtos
Visualizar supermercados	Visualizar os supermercados que determinado produto é encontrado
Visualizar histórico de supermercado	Ver as alterações de preço de um determinado produto em cada supermercado
Criar conta	Fazer cadastro no sistema
Realizar *login*	Entrar na sessão com suas credenciais

Fonte: Elaborado pelo autor

A Tabela 4 comporta as funcionalidades disponíveis apenas para os usuários autenticados, após realizarem o cadastro e autenticação no sistema. Esses usuários, também possuem acesso a todas as funcionalidades listadas na Tabela 3.

Tabela 4 – Funcionalidades do usuário autenticado.

Funcionalidade	**Descrição**
Adicionar à lista	Adicionar produtos à lista de compra pessoal
Remover da lista	Remover produtos da lista de compra pessoal
Visualizar lista	Ver a lista de compras e o somatório de preços dos produtos
Visualizar lista por supermercado	Ver a disponibilidade e somatório do preço de produtos em cada supermercado
Realizar *logout*	Sair da sessão de usuário

Fonte: Elaborado pelo autor

3.3 Pesquisa de supermercados e análise de coleta

Para buscar os supermercados alvos na região de Belo Horizonte foi utilizado o serviço Openrouteservice[1], que fornece uma API com informações geográficas como pontos de interesse dentro de uma área delimitada. Na pesquisa conduzida, se buscou por estabelecimentos que se encaixavam na categoria de mercearia ou supermercado. Após a configuração dos parâmetros desejados, a requisição feita à API retorna uma lista com vários locais, contendo uma série de informações a respeito deles. Não se limitando apenas à essa ferramenta, também é possível buscar pelos supermercados no Google Maps[2] ou no próprio mapa interativo[3] oferecido pelo Openrouteservice.

Durante a pesquisa, dentre as informações mais interessantes na busca dos supermercados, estão: a presença de um *site*, que pode conter uma listagem dos produtos vendidos, nome da organização, além do endereço, Código de Endereçamento Postal (CEP) e coordenadas geográficas de suas filiais. Apenas os estabelecimentos que possuem um catálogo *online* completo dos seus produtos, conseguem compor a base de dados do sistema. Devido a este motivo, *sites* de estabelecimentos populares na região como o Supermercados BH[4] não foram contemplados, visto que na data na qual este trabalho está sendo desenvolvido, não são cumpridos os requisitos propostos. Por meio desse processo, foram escolhidos os seguintes supermercados, dentro da região delimitada, que se encaixaram em todos os requisitos:

[1] Disponível em: <https://openrouteservice.org/>
[2] Disponível em: <https://www.google.com.br/maps/preview/>
[3] Disponível em: <https://maps.openrouteservice.org/>
[4] Disponível em: <https://www.supermercadosbh.com.br/belo-horizonte/>

- Carrefour[5]
- Super Nosso[6]
- Verdemar[7]

Após serem determinados os supermercados, é preciso fazer uma análise da forma como as informações dos seus respectivos *sites*, chegam até o usuário, para que sejam possível realizar a coleta de dados. Cada empresa possui uma implementação diferente que se encaixa às necessidades dela. Um fator importante é a localização, visto que uma mesma empresa pode possuir diversas filais e cada uma dessas pode ter um preço diferente para um mesmo produto.

Foram utilizadas as ferramentas de inspeção, comumente presente em navegadores *web* modernos, para observar o fluxo de rede nas páginas. Uma característica em comum de todos o sistemas é o carregamento dinâmico da lista de produtos. Ao descobrir o endereço dos *endpoints*, ou rotas, responsáveis por trazer a lista de produtos, se passa a estudar as variáveis empregadas nas requisições, para ser possível manipular e obter uma lista completa de produtos de todas as filiais.

3.4 *Matching* de produtos

Entre os desafios encontrados na coleta de dados, se destacam a categorização e associação. Para cada *site*, é preciso conhecer quais as categorias existentes, e posteriormente dentro da mesma categoria, conseguir identificar produtos iguais em meio a todos os supermercados. Esses são os principais passos para viabilizar a comparação de produtos.

3.4.1 Categorias

A categorização de produtos não possui um único padrão. Os supermercados podem agrupar seus produtos em categorias mais especificas ou mais genéricas. De maneira simplificada, um supermercado pode utilizar somente a categoria “bebidas”, para se referir à todos os consumíveis líquidos, enquanto um segundo, pode fazer a separação entre “bebidas” e “bebidas alcoólicas”. Por este motivo, se optou por utilizar uma representação mais genérica como caso base na categorização.

As principais categorias encontradas foram: mercearia; carnes; bebidas; frios e laticínios; congelados; hortifrúti; infantil; higiene e beleza; limpeza e *pet*. Categorias que fogem do escopo ou são muito específicas, foram desconsideradas ou integradas em uma mais genérica, durante a coleta de dados.

[5] Disponível em: <https://mercado.carrefour.com.br/>
[6] Disponível em: <https://www.supernossoemcasa.com.br/>
[7] Disponível em: <https://www.loja.verdemaratevoce.com.br/>

Além da categorização, é preciso fazer a identificação dos produtos, para que seja possível compará-los em diferentes supermercados. Neste cenário, busca-se por um atributo que seja relativamente consistente, mantendo um valor igual ou próximo, para produtos idênticos de supermercados distintos.

3.4.2 *Matching*

O *matching*, ou o ato de corresponder, equiparar, associar produtos, é uma parte crucial da regra de negócio do sistema proposto. Para que a comparação entre supermercados seja viável, é preciso identificar cada produto individualmente.

Em uma abordagem convencional, seria possível fazer essa comparação pelo *Global Trade Item Number* (GTIN), um identificador global único muito utilizado em estabelecimentos físicos. A disponibilidade deste número em lojas *online* não é comum, tornando difícil utilizá-lo como referencial. Uma outra alternativa para identificar produtos, seria a utilização da *Stock Keeping Unit* (SKU). Este valor é uma identificação única para cada item, dentro do estoque de um determinado estabelecimento. Um dos problemas com essa abordagem, é que cada supermercado implementa seu próprio valor, de maneira que, um mesmo produto muito dificilmente terá o mesmo SKU em sistemas diferentes.

Feita a análise das alternativas, se decidiu fazer o uso do nome do produto como identificador. Este atributo é um dos primeiros que o usuário analisa, pois contém uma série de informações importantes a respeito do item, como o nome, marca, quantidade, dentre outras características. Por não possuir uma padronização, cada *website* pode incluir ou remover propriedades do nome, além de sofrer alterações no espaçamento, ordenação, letras maiúsculas e minúsculas, porém se mantém moderadamente consistente.

Na sequência, é apresentada uma lista contendo cinco nomes de produtos da Coca-Cola, todos do tipo lata, contendo um volume de 350ml. As descrições possuem variação na quantidade e ordenação das palavras.

1. Coca-Cola 350ml
2. Coca-Cola 350 ml
3. Coca-Cola Zero Açúcar 350ml
4. Coca-Cola 350ml Zero Açúcar

Os itens 1 e 2, são descrições para um mesmo produto, enquanto os itens 3 e 4 representam outro. O desafio está em fazer a comparação dos nomes, para conseguir identificar cada grupo.

Afim de expressar a lógica de comparação proposta, a Tabela 5, mostra o comportamento esperado do algoritmo, para duas entradas de texto. Inicialmente, todas letras são convertidas para letras minúsculas, e na sequência, é removida a acentuação de forma que a comparação não seja impactada. Em seguida, são realizados dois tipos de verificação. A primeira compara as palavras ignorando o espaçamento, isto é, removendo os espaços e considerado o nome como uma única linha de letras. A segunda verificação, transforma cada nome em um conjunto de palavras, e na sequência, é utilizado uma função para testar se os conjuntos são iguais.

Tabela 5 – Lógica da comparação de texto.

Entrada 1	**Entrada 2**	**Resultado**
A B	A B	Verdadeiro
A B	B A	Verdadeiro
AB	A B	Verdadeiro
AB	B A	Falso

Fonte: Elaborado pelo autor

Aplicando esta lógica à listagem feita anteriormente, os produtos são divididos em dois grupos como esperado. Os itens 1 e 2 são considerados o mesmo produto, pois ao ignorar o espaçamento, as letras seguem a mesma ordem. Já os produtos 3 e 4, formam outro grupo, pois possuem o mesmo conjunto de palavras, apesar de terem ordenações diferentes. Entretanto, existem casos onde o algoritmo não consegue identificar corretamente o produto:

- Coca-Cola Lata 350ml

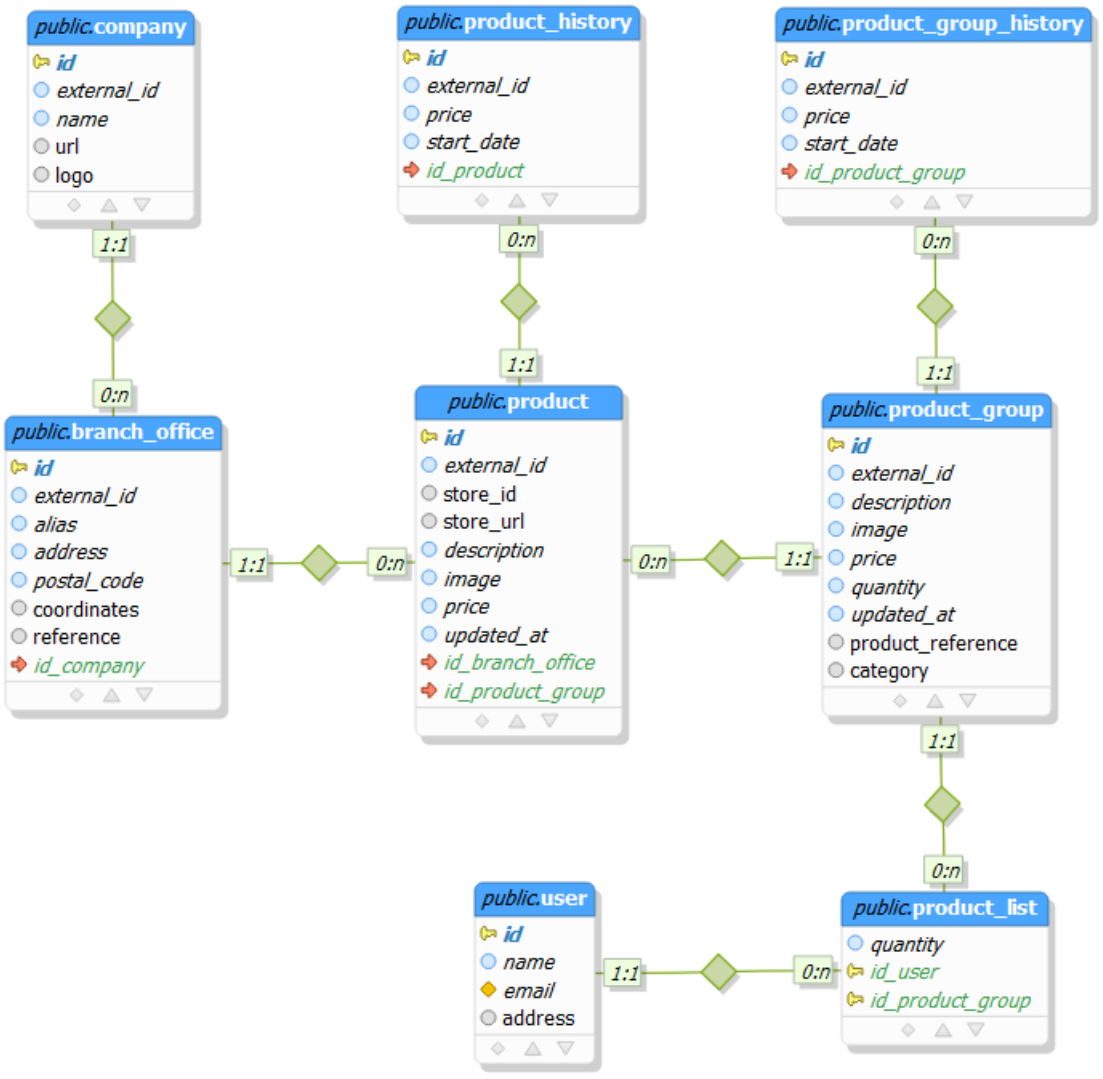
Neste exemplo, o item deveria ser inserido no mesmo grupo dos itens 1 e 2. Devido à palavra ‘Lata‘, um atributo que descreve uma das características do produto, o algoritmo entende esta descrição como um grupo próprio. A primeira verificação falha, pois as letras não seguem a mesma ordem, enquanto a segunda falha devido aos conjuntos não serem mais idênticos.

3.5 Modelo de dados proposto

Concluindo o levantamento de requisitos e o estudo dos dados disponibilizados pelos supermercados, é construído o Modelo Entidade e Relacionamento, que representa as tabelas, atributos e relacionamentos que serão criados no banco de dados. É importante que essa representação seja clara e satisfaça as necessidades do sistema, caso contrário, podem surgir problemas durante o desenvolvimento relacionados à modelagem dos dados.

A seguir, a Figura 7 ilustra a modelagem do banco. As tabelas de *company* e *branch_office*, representam a empresa e suas filiais respectivamente. Os produtos são armazenados na tabela *product*, enquanto o seu histórico de preço é guardado na *product_history*. Para cada produto, existe um grupo de produtos representado pela tabela *product_group*, responsável por associar mercadorias iguais de diferentes filiais ou empresas, além de contar com um histórico próprio na tabela *product_group_history*. A tabela *user* representa o usuário do sistema, enquanto a *product_list* guarda a sua lista de compras.

Figura 7 – Diagrama Entidade Relacionamento



Fonte: Elaborado pelo autor

O grupo de produtos mostrado na representação, é o objeto chave da lógica do sistema, sendo responsável por guardar as informações obtidas pelo *matching*. Além disso, armazena os dados do produto com menor preço, entre todos os produtos pertencentes a ele. A condição para um produto ser incluído em um grupo, é ter um nome que seja

equivalente à sua descrição, mediante a lógica de comparação discutida anteriormente.

A principal ideia do grupo de produtos, é facilitar a implementação. Ao representar produtos idênticos, por um grupo, busca-se simplificar a maneira como o sistema manipula a informação. Quando o usuário realiza uma pesquisa, ou adiciona um item à sua lista de compras, se trabalharia apenas com o grupo responsável por identificar o produto em questão. É imprescindível que os dados do grupo sempre se mantenham atualizados, a fim de refletir fielmente os produtos que ele representa.

3.6 Desenvolvimento do robô (Harvester)

A primeira etapa de desenvolvimento, tem o foco em criar o serviço que irá percorrer os sistemas públicos dos supermercados, para obter os dados de entrada que compõem a maioria das funcionalidades da aplicação. Nesta parte foi empregado um sistema em Node.js, que possui ligação com o banco de dados, para requisitar e inserir dados referentes aos supermercados. O robô responsável por esta tarefa foi apelidado de Harvester.

Na Figura 8, é mostrado um pseudo código que representa a lógica construída para obter os dados. O algoritmo começa recuperando a lista de empresas e categorias, presentes no banco de dados. Para cada empresa, são buscadas as suas filiais, e para cada filial, itera-se sobre todos as categorias. No laço mais aninhado, é chamada a função responsável pela raspagem de dados. O retorno desta função é uma lista de produtos, que é então enviada para o banco de dados.

Figura 8 – Pseudo código da lógica do robô

Algoritmo 1: Fluxo de operações do robô

Entrada: Uma lista de *empresas* e uma lista de *categorias* do banco de dados

```
1  início
2      para cada empresa ∈ empresas faça
3          filiais ← EncontrarFiliais(empresa);
4          para cada filial ∈ filiais faça
5              para cada categoria ∈ categorias faça
6                  produtos ← RaspagemDeDados(categoria, filial);
7                  PersistirProdutos(produtos, categoria, filial);
8              fim
9          fim
10     fim
11 fim
```

Fonte: Elaborado pelo autor

A maneira escolhida para percorrer os *websites*, foi por meio da reconstrução das chamadas de API. O processo funciona de maneira análoga a um usuário acessando o *site*, informando sua localização, selecionando uma categoria e navegando pela lista de produtos. Este comportamento foi refeito, ignorando o ambiente do navegador, fazendo requisições diretamente para as APIs envolvidas.

Devido à singularidade de cada supermercado, foi preciso desenvolver um *script* especifico para cada um. Os *scripts*, são códigos responsáveis por implementar a lógica das APIs e retornar os produtos encontrados. Suas principais funções são realizar requisições HTTP e fazer o tratamento dos dados. Dentre os sistemas analisados, foi notado o uso da plataforma VTEX IO[8], que tem o foco em soluções de comércio digital para grandes empresas.

A inserção no banco de dados foi feita com a chamada de um procedimento, que recebe os dados do produto, horário da coleta, categoria e a qual filial os dados pertencem. Este procedimento contém a lógica que identifica o produto individualmente, avaliando se será criado um novo produto ou atualizado um já existente, além de encontrar um grupo para inseri-lo, ou caso necessário, criar um novo grupo.

Para simplificar o processo de guardar o histórico de preços, foram criados gatilhos, ou *triggers*, no banco de dados. Este recurso é acionado quando ocorre um evento de inserção ou atualização, em ambas as tabela de produtos e de grupo de produtos. Caso o produto tenha mantido o mesmo preço, nenhuma operação é realizada. Se o produto de menor preço dentro de um grupo, sofrer alterações, o procedimento consegue atualizar a informação na tabela de *product_groups*.

Um dos problemas encontrados durante o desenvolvimento, foi o tratamento dos erros gerados pelas APIs dos supermercados. A abordagem escolhida foi a seguinte: quando um produto possui algum atributo ausente ou estranho, ele é ignorado; caso a API devolva algum erro interno, se espera por um determinado intervalo, antes de re-enviar a requisição. Foi implementado um sistema de *log* para mostrar a situação do robô em tempo real.

3.7 Desenvolvimento do *backend*

O *backend* do trabalho foi desenvolvido utilizando o *framework* Express, contextualizado na Seção A.1.2. O objetivo é a criação de uma API REST, uma arquitetura onde tudo é representado por recursos, que possuem um identificador único (SOMMERVILLE, 2019). As extremidades da API que se expõem para receber requisições, são chamadas de *endpoint*.

Por meio do protocolo HTTP, são utilizados os verbos GET, POST, PUT, PATCH

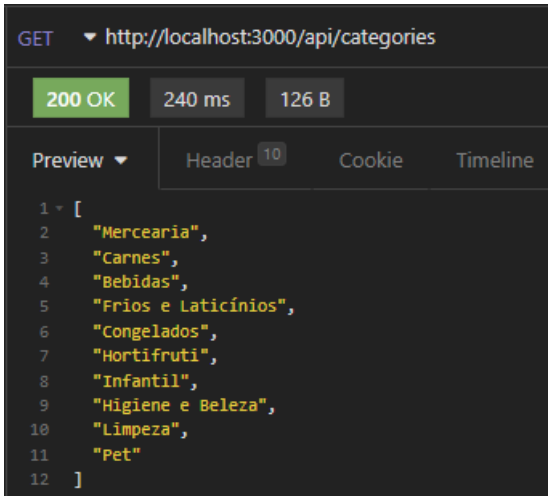
[8] Disponível em: <https://vtex.com/br-pt/vtex-io/>

e DELETE, para se comunicar com o *backend*. Devido a entrada de dados automatizada fornecida pelo robô, neste momento não é necessária a criação das quatro operações básicas *Create, Read, Update and Delete* (CRUD) para todos os registros, pois os produtos e empresas não precisam ser modificados diretamente por usuários. Logo, se construiu principalmente requisições do tipo GET, com exceção da remoção de produtos da lista de compras do usuário. A documentação de todas rotas criada para a API REST, pode ser encontrada no Apêndice B.

A autenticação no *backend*, consistiu na criação de um *middleware*, que é uma camada para verificar a autenticidade das requisições, aplicada à todas as rotas do usuário. O sistema recebe um *JSON Web Token* (JWT), juntamente da requisição *Hyper Text Transfer Protocol Secure* (HTTPS). Este *token*, definido pela *Request for Comments* (RFC)-7519[9], é gerado pela plataforma Auth0, descrita na Seção A.3, do apêndice de tecnologias. Em seguida, ele é enviado à aplicação do *frontend*, quando se é iniciado uma sessão de usuário. Quando este *token* chega ao *backend*, ele é validado pela API de autenticação da Auth0, que verifica a assinatura e garante que nada foi alterado.

A Figura 9, mostra um exemplo de requisição que pode ser efetuada para a o sistema. No caso, foi enviada uma chamada do tipo GET para o endereço local da API, com o alvo na rota de categorias, representado pelo trecho *api/categories*. Esta rota, tem como função retornar a lista de categorias de produtos, que está armazenada dentro do banco de dados. O resultado da requisição pode ser observado na aba de *preview*, que contém um vetor com todas as categorias presentes no sistema.

Figura 9 – Requisição da lista de categorias



Fonte: Elaborado pelo autor

Com o servidor *backend* conectado ao banco de dados, e provendo serviços de

[9] Disponível em: <https://datatracker.ietf.org/doc/html/rfc7519>

maneira RESTful, isto é, um sistema que implementa uma arquitetura REST, é possível começar o desenvolvimento da interface do usuário que irá consumir esses serviços, sendo o assunto da próxima seção.

3.8 Desenvolvimento do *frontend*

O *framework* Angular, utilizado para construção do *frontend web*, permite a iniciação rápida de uma aplicação base, gerando toda a estrutura de arquivos necessária. A partir desse ponto, se começa a criar e customizar as telas, de acordo com a demanda da aplicação que está sendo desenvolvida.

Cada componente criado no Angular possui um arquivo exclusivo para o código de estilização no formato (*Cascading Style Sheets* (CSS)), outro para organização dos elementos na página seguindo o modelo (*HyperText Markup Language* (HTML)), e por fim, um arquivo TypeScript para controle da entrada e saída de dados do componente. Um dos desafios nesta etapa do desenvolvimento, foi a organização de *layout* dos objetos. Pelo uso de componentes já documentados, fornecidos pela biblioteca do Angular Material[10], foi possível fazer a construção das telas, utilizando recursos estilizados e funcionais.

A página inicial da aplicação, foi construída com base nas características observadas durante a Seção 2.1. O objetivo é apresentar uma lista de produtos, que possuí filtros e ações, para ajudar o usuário na pesquisa. Estes recursos são referentes à ordenação dos elementos, filtro de categorias, barra de pesquisa e paginação, isto é, permitir ao usuário navegar pelas páginas de itens. Para os produtos individuais, foi construída uma pagina que mostra todas as suas informações, como o nome, melhor preço, preço em cada filial de supermercado, bem como o histórico de alterações no valor.

Para mostrar os históricos de preços na interface do usuário, foi escolhido utilizar a ferramenta gráfica do ApexCharts[11], que possui uma biblioteca de integração com o *framework* utilizado. Neste cenário, cada gráfico é visto como um componente dentro do sistema. Os dados de entrada para popular os gráficos, são obtidos pelos *endpoints* criados no *backend*. Ainda na página do produto, o usuário tem a opção de adicionar itens à sua lista de compras.

Foi desenvolvida uma tela de perfil, onde será mostrada a lista de produtos do usuário. Dentro dela, são mostrados os menores preços para cada produto, além de permitir que o usuário veja qual o preço da sua lista, em diferentes supermercados. O usuário deve ser capaz de encontrar o melhor preço, no estabelecimento que desejar. Oferecer este recurso é um dos pontos mais importantes do sistema.

A autenticação é integrada diretamente ao *frontend*, por meio de um *Software*

[10] Disponível em: <https://material.angular.io/>
[11] Disponível em: <https://apexcharts.com/>

Development Kit (SDK) que conecta o sistema à aplicação do Auth0, provendo os métodos necessários para tratar as sessões de usuário. Além de criar o estado da sessão, é por meio desta ferramenta que se torna possível obter o *token* do usuário, que é necessário para utilizar os *endpoints* protegidos pelo *middleware* de autenticação.

Para se ter acesso aos *endpoints* criados na seção anterior dentro do Angular, são criados serviços, que fazem as requisições HTTP, e disponibilizam os dados da resposta para a aplicação. A fim de facilitar a utilização dos *endpoints* em casos mais complexos, são criados modelos de interface que representam o objeto devolvido na resposta da requisição. Dessa maneira, é possível traduzir as rotas do *backend* em funções, que podem ser chamadas dentro do código quando necessário.

Chegando ao fim das etapas de desenvolvimento, é obtido um sistema totalmente funcional. Dentre suas principais funcionalidades, são destacadas a capacidade de obter dados de produtos, categorizá-los e armazená-los em um banco de dados, acessar o banco para servir os dados em uma API REST, além de entregar estes dados para o usuário em um *website*. Parte-se agora, para a análise dos resultados obtidos, pelo sistema desenvolvido.

4 Resultados

Este capítulo busca mostrar os resultados obtidos no desenvolvimento do sistema, tais como a coleta de dados feita pelo robô, as telas construídas e as visualizações do usuário.

4.1 Dados coletados

O robô implementado para coletar os dados dos supermercados *online*, é a principal fonte de entrada de dados do sistema. Esta seção tem o objetivo de mostrar o que foi coletado, pela execução dos *scripts* desenvolvidos.

A Tabela 6 mostra o volume de dados coletados pelo robô. São listados a quantidade de produtos de cada supermercado e os tempos de execuções, por categoria. Foram analisadas quatro filiais para o supermercado Carrefour, uma para o Supernosso e oito para Verdemar. O volume de produtos representa a soma dos produtos em todas as suas filiais. Para o Carrefour, a coleta de dados levou 11 minutos e 47 segundos, já para o Supernosso foi de 4 minutos e 33 segundos, e por fim, Verdemar com 20 minutos e 31 segundos. As categorias de mercearia, bebidas e higiene, foram as que apresentaram maior quantidade de produtos, o que também eleva o tempo de processamento.

Tabela 6 – Métricas da coleta de dados.

Categorias	**Carrefour**	**Supernosso**	**Verdemar**	**Produtos**	**Tempo**
Mercearia	11254	2987	13133	27374	9m19s
Carnes	1890	344	2934	5168	1m51s
Bebidas	5570	2027	12382	19979	6m48s
Frios e Laticínios	2206	489	6549	9244	3m34s
Congelados	1661	309	3798	5768	2m3s
Hortifruti	780	204	2566	3550	1m25s
Infantil	951	-	-	951	27s
Higiene e Beleza	7974	1703	10395	20072	6m14s
Limpeza	3805	794	6096	10695	3m33s
Pet	668	102	2311	3081	1m37s
Total	36759	8959	60164	**105882**	**36m51s**

Fonte: Elaborado pelo autor

Por meio da execução dos *scripts*, foi possível obter dados de 105.000 produtos, em cerca de 36 minutos. Destes, foram criados 30.000 grupos de produtos, aplicando o

algoritmo discutido na Seção 3.4.2. É esperado que existam grupos de produtos iguais que estejam repetidos, devido às diferenças na descrição utilizada por cada supermercado. O serviço de coleta deve ser executado diariamente para ser possível gerar históricos precisos.

É importante mencionar que, foi considerada somente uma filial durante o processo de coleta do estabelecimento Supernosso, pois à API principal do supermercado, não diferenciava os preços dos produtos baseado na localização. Isso significa que não é possível distinguir a coleta por filial. Dentro do banco de dados do projeto, este supermercado apresenta 28 filiais na região de Belo Horizonte. Foi feito um caso especial dentro do robô, para evitar executar a coleta de dados várias vezes para este estabelecimento.

4.2 Aplicação *web*

A interface criada dentro do *framework* Angular, é a forma como o usuário consegue interagir com o sistema, para visualizar os dados e fazer comparações. É importante que a aplicação entregue as funcionalidades propostas para o sistema. Esta seção busca mostrar o resultado visual do que foi desenvolvido.

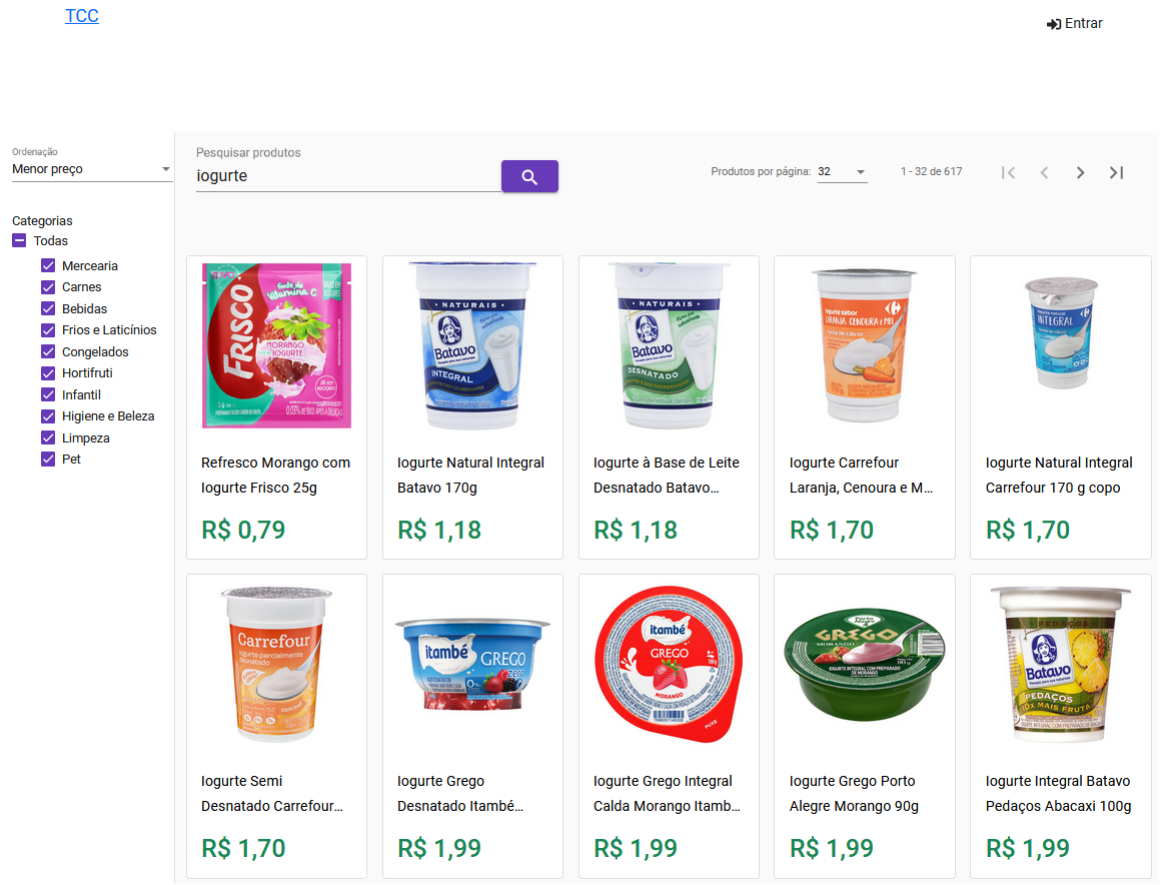
Na página inicial apresentada pela Figura 10, é mostrado a listagem dos produtos. Na esquerda da página, pode-se selecionar a ordenação por menor ou maior preço, além de escolher quais categorias se deseja ver. Em cima da lista de produtos, se encontram o campo de pesquisa, onde um produto pode ser pesquisado por nome, e ao lado, as ações de paginação, permitindo selecionar a quantidade de produtos mostrada, além de navegar pelas páginas. Ao clicar em um produto da lista, o usuário é redirecionado à página de produto.

A página do produto, ilustrada pela Figura 11, é mostrado o menor preço atual, bem como os preços individuais dos supermercados que possuem o produto especificado. Um gráfico mostra as flutuações do menor preço, entre os estabelecimentos. O usuário pode adicionar um produto à sua lista, ao clicar no botão “Adicionar à lista“. A ação anterior requer uma sessão, logo, caso o usuário não esteja autenticado, ele irá ser redirecionado para a página de *login*. O usuário pode escolher entrar com um *login* social como da Google, em uma página auto-gerada pela ferramenta de autenticação demonstrada na Figura 12,

No fim da página de produto, existe um botão para mostrar o histórico completo do preço em todos os supermercados. O gráfico resultante é visualizado na Figura 13. Essa visualização permite entender as diferenças de preço entre filiais, durante o tempo.

A lista de produtos do usuários é apresentada na Figura 14. Por ela, é possível ter o controle de quais produtos estão incluídos na lista pessoal, além de poder ir para a página do produto visualizá-lo. No topo da lista, é mostrado o botão ‘Encontrar supermercados‘, que encaminha o usuário para a página de preços por filial.

Figura 10 – Página inicial da aplicação

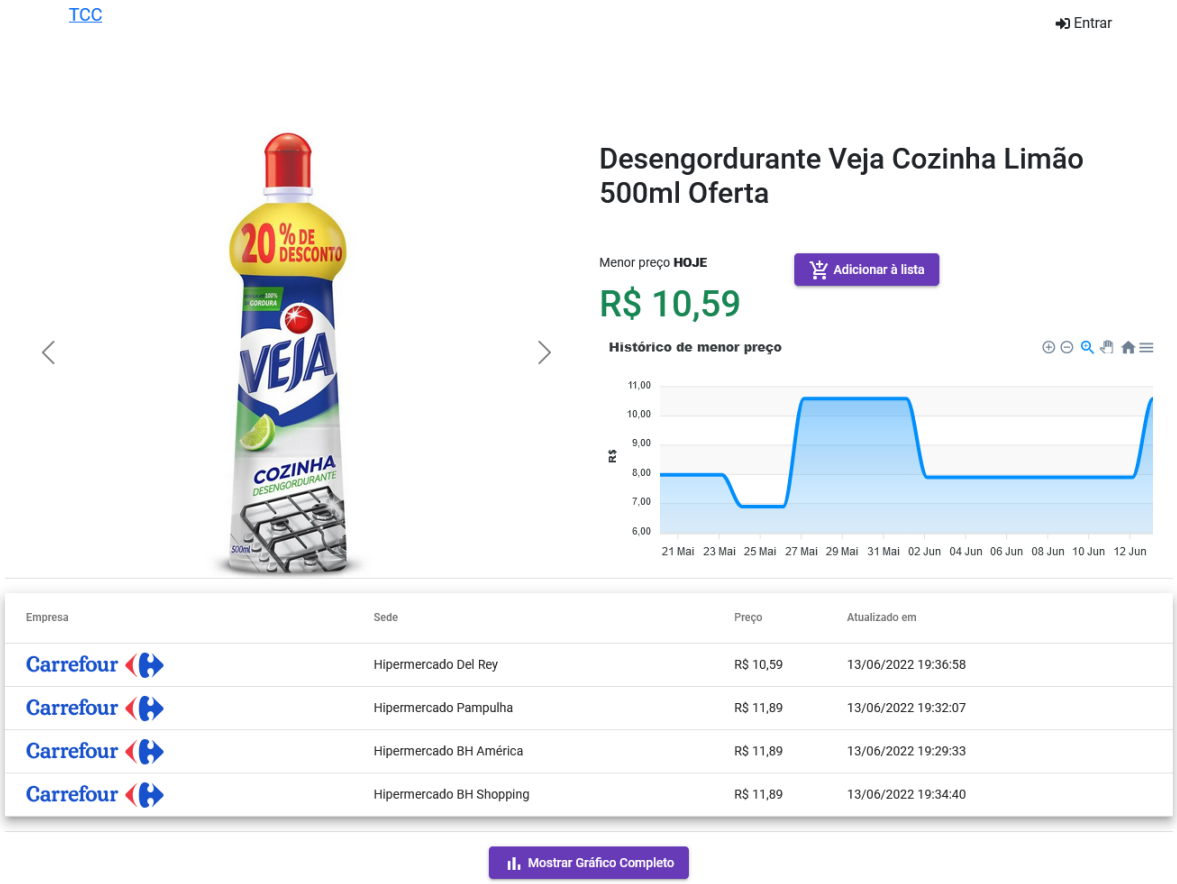


Fonte: Elaborado pelo autor

Na página de preços por supermercado são montadas várias listagens contendo os produtos da lista do usuário, com seus preços para cada uma das filiais cadastradas no sistema. Esta página é mostrada na Figura 15. A lista de filiais na esquerda é ordenada de acordo com os melhores locais de compra, com base na disponibilidade e menor preço dos produtos. Para cada lista, é mostrado o preço da filial e o preço de referência, isto é, o menor preço praticado para este produto.

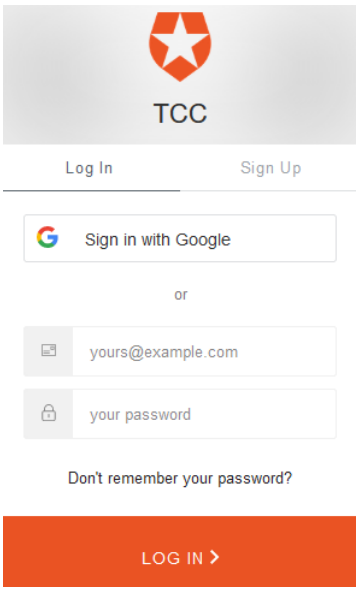
Quando um supermercado não possui todos os itens do usuário, isto é sinalizado na barra lateral. Dentro da lista, também é mostrado qual ou quais produtos estão faltando, assim como mostrado na Figura 16.

Figura 11 – Página do produto



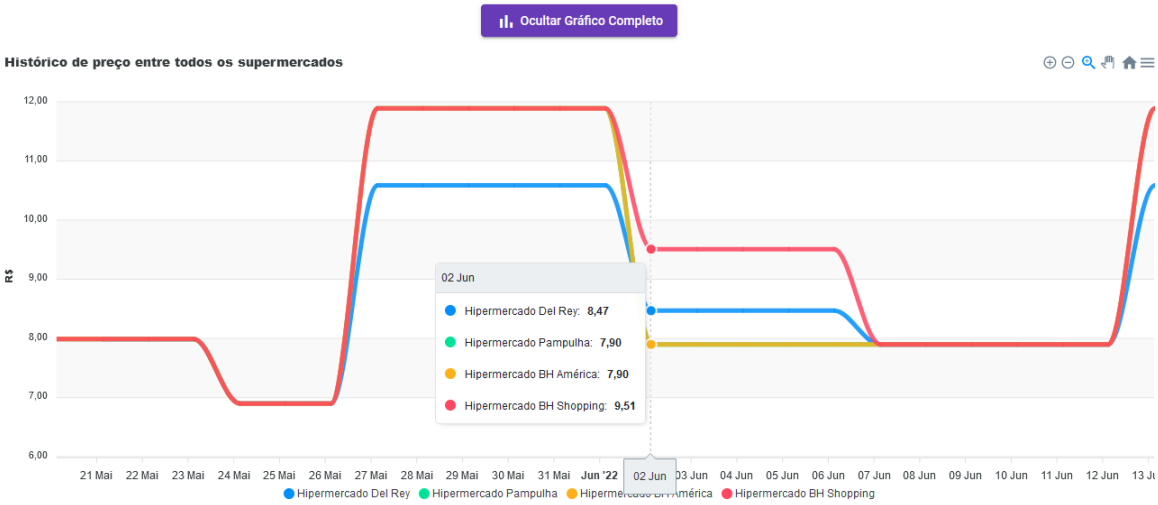
Fonte: Elaborado pelo autor

Figura 12 – Página de *login*



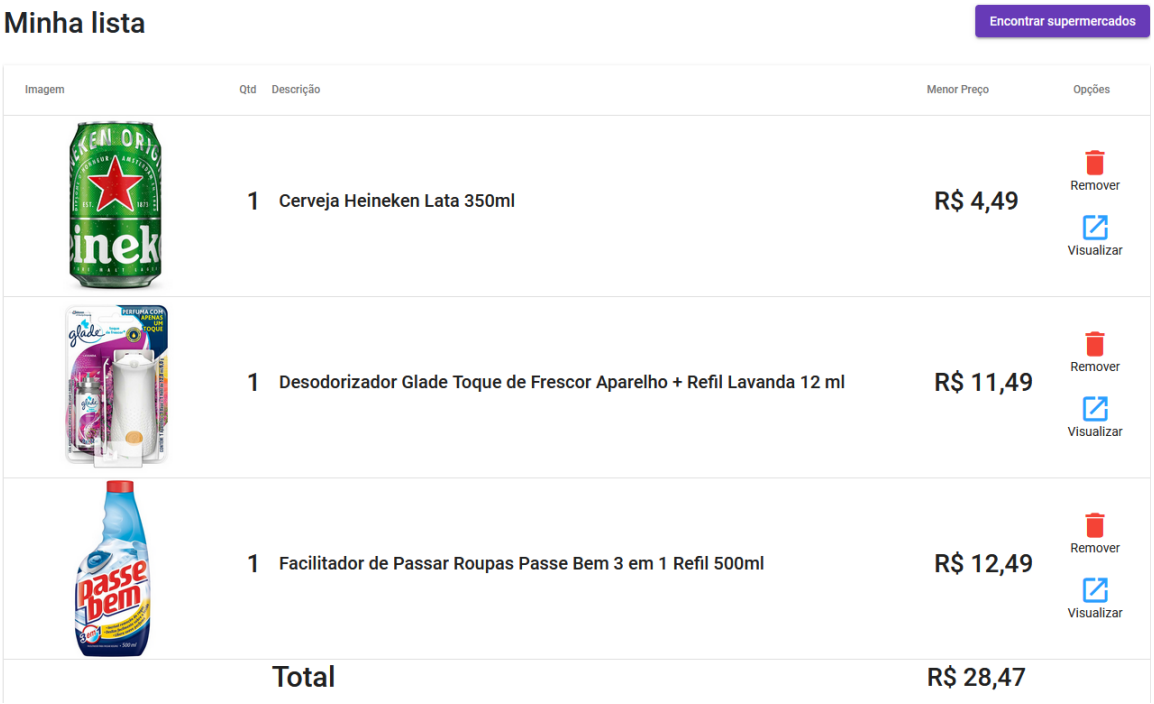
Fonte: Elaborado pelo autor

Figura 13 – Gráfico de preço em todos os supermercados



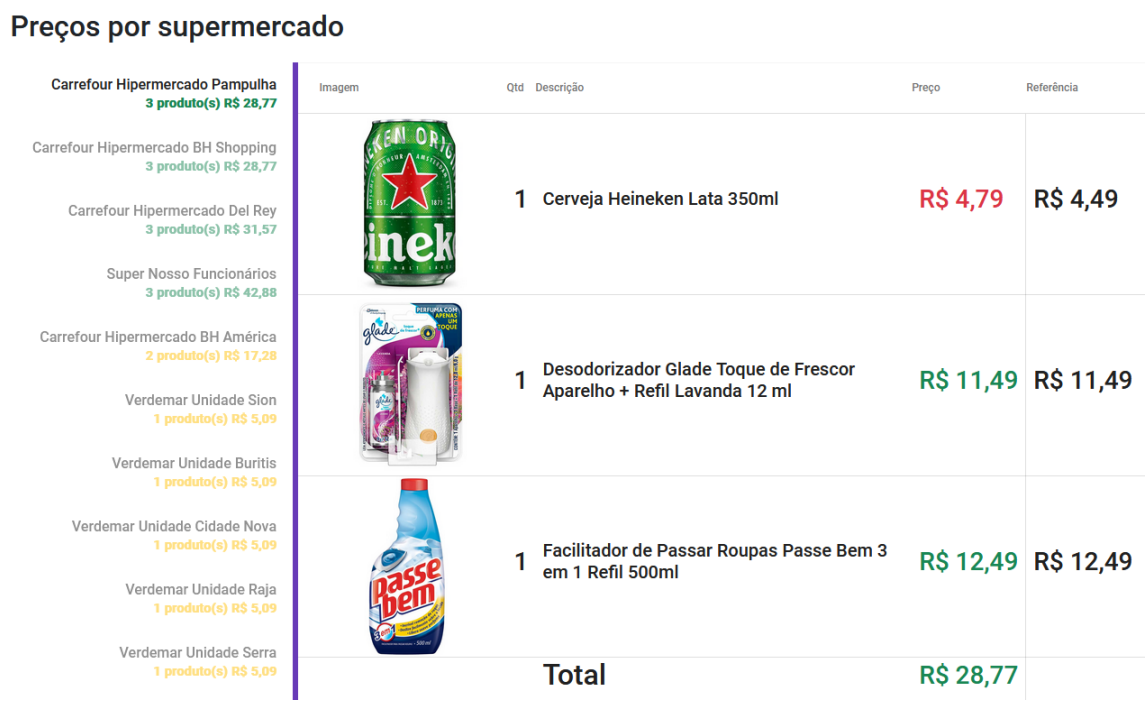
Fonte: Elaborado pelo autor

Figura 14 – Lista de produtos do usuário



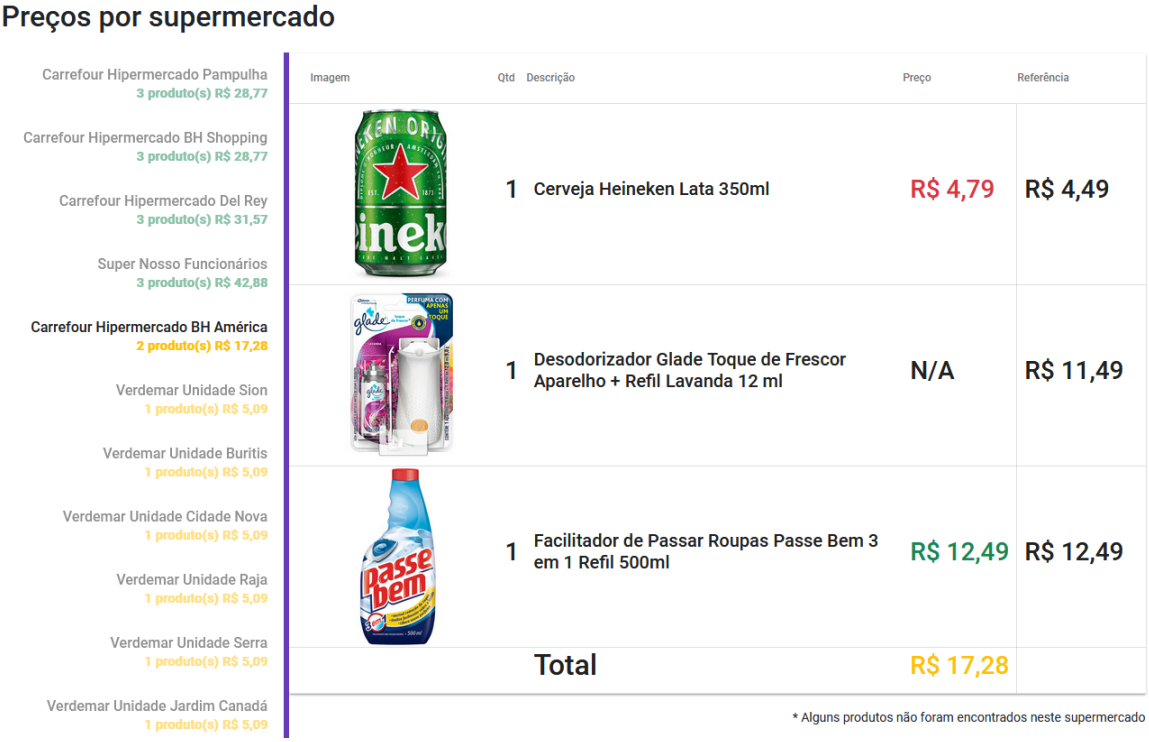
Fonte: Elaborado pelo autor

Figura 15 – Comparação de preço por supermercado



Fonte: Elaborado pelo autor

Figura 16 – Comparação de preço por supermercado faltando produtos



Fonte: Elaborado pelo autor

5 Conclusões

Este trabalho apresentou o desenvolvimento de um sistema *web* para comparação de preço dos supermercados *online* na cidade de Belo Horizonte. Para obtenção dos dados dos produtos, foi construído um robô que percorre os sistemas dos supermercados de forma automatizada. A análise de ferramentas similares foi uma parte fundamental para guiar o desenvolvimento do projeto, assim como as tecnologias escolhidas tiveram ligação com o objetivo desejado.

A aplicação resultante permite que o usuário monte sua própria lista de produtos e, em seguida, encontre os estabelecimentos que oferecem os melhores preços. O robô desenvolvido para realizar a raspagem de dados obtêm uma quantidade de dados satisfatória de acordo com o tempo de processamento. Foi implementada uma categorização e agrupamento de produtos utilizando a lógica de *matching* proposta. O objetivo principal de gerar listas de produtos para diferentes supermercados, que dependia da construção e integração das partes do sistema propostas, conseguiu ser alcançado.

Uma limitação observadas no projeto é referente ao *matching* de produtos. Devido a identificação global do GTIN não estar presente nos *sites* dos supermercados, se criou um impedimento durante o desenvolvimento, mas ao utilizar a lógica de *matching* proposta, foi possível contornar este problema. Uma consequência gerada por esta abordagem foi não ser possível tratar todos os casos de agrupamento corretamente, devido às variações no nome dos mesmos produtos de diferentes supermercados. Um impacto disso no projeto, são mais produtos isolados que não conseguem ser identificados adequadamente, o que implicou em listas de preço por supermercado menos completas. Além disso, existiu um fator limitante no número de estabelecimentos contemplados pela raspagem de dados. Se encontrou uma escassez de supermercados que disponibilizavam seus catálogos de produtos *online*. Em muitos casos existem *sites*, mas eles não possuem uma listagem de produtos completa.

A relevância deste trabalho está relacionada com a possibilidade de auxiliar os consumidores, na hora de escolher o supermercado mais adequado. Existem múltiplas opções de *sites* para realizar uma compra, mas para saber se uma lista de produtos tem bom preço, é necessário investir tempo pesquisando diferentes fontes. Foi entendido que o projeto proposto pode ser uma solução viável para esta questão.

Por fim, se acredita que a abordagem realizada durante o trabalho possibilitou o estudo e construção de uma ferramenta para comparação de preços em *sites* de supermercados, que é pouco explorado na região escolhida. Os resultados obtidos se encontram de acordo com a proposta do trabalho, atendendo os requisitos identificados.

5.1 Trabalhos futuros

O desenvolvimento de um *backend* desacoplado da interface do usuário, possibilita a expansão do sistema para outras áreas como a da aplicação móvel, sem que seja preciso reescrever código. A construção de um aplicativo móvel para o sistema, pode se tornar uma expansão de alcance considerável, visto o grande número de usuários de dispositivos móveis.

Devido à complexidade do projeto, neste momento não foi integrado a API fornecida pelo Openrouteservice para localização do usuário dentro da aplicação. Atualmente o sistema não leva em conta a localização do usuário ao sugerir supermercados. Além disso, o *framework* do Angular possui ferramentas nativas para testes, que não foram abordadas neste trabalho. Dentre essas e outras possíveis melhorias e funcionalidades, pode-se listar resumidamente:

- Expansão do sistema para o ambiente *mobile* com a criação de um novo *frontend*.
- Implementação da localização de usuário.
- Melhorias gráficas na interface *web* existente.
- Adicionar funcionalidade de alterar quantidade de produtos na lista de compra.
- Adicionar mais filtros de pesquisa, como empresa e limite de preço.
- Adicionar um sistema de inscrição para receber alertas de mudança de preço.
- Realizar testes para validar mais a fundo os componentes da interface *web*.
- Melhoria na lógica de *matching* dos produtos.
- Disponibilizar a aplicação em um ambiente na nuvem.

Referências

ALAM, A. et al. Upoma: A dynamic online price comparison tool for bangladeshi e-commerce websites. In: *2020 IEEE Region 10 Symposium (TENSYMP)*. [S.l.: s.n.], 2020. p. 194–197. Citado 2 vezes nas páginas 20 e 21.

ARAÚJO, M. C. et al. Análise de usabilidade do aplicativo de compras para dispositivo móvel "zoom". *Blucher Engineering Proceedings*, v. 3, n. 3, p. 1114–1124, 2016. Citado na página 23.

BAKOS, Y. The emerging landscape for retail e-commerce. *Journal of economic perspectives*, v. 15, n. 1, p. 69–80, 2001. Citado na página 17.

BRYNJOLFSSON, E.; SMITH, M. D. Frictionless commerce? a comparison of internet and conventional retailers. *Management science*, INFORMS, v. 46, n. 4, p. 563–585, 2000. Citado na página 20.

COSTA, P. T. G. C. da et al. E-commerce no brasil: revisão sistemática de literatura de 2011 a 2021. *Brazilian Journal of Business*, v. 3, n. 4, p. 2969–2982, 2021. Citado na página 16.

DEGERATU, A. M.; RANGASWAMY, A.; WU, J. Consumer choice behavior in online and traditional supermarkets: The effects of brand name, price, and other search attributes. *International Journal of research in Marketing*, Elsevier, v. 17, n. 1, p. 55–78, 2000. Citado 2 vezes nas páginas 17 e 20.

FERNANDES, B. S. L. Shopbots: um estudo exploratório dos comparadores de preços. Mestrado – Universidade do Porto, 2022. Citado na página 19.

G1. *Zoom compra concorrente Buscapé*. [S.l.], 2022. Disponível em: <https://g1.globo.com/economia/noticia/2019/05/14/zoom-compra-concorrente-buscape.ghtml>. Acesso em: 30 mai. 2022. Citado na página 24.

GAO, D. et al. The bullwhip effect in an online retail supply chain: A perspective of price-sensitive demand based on the price discount in e-commerce. *IEEE Transactions on Engineering Management*, v. 64, n. 2, p. 134–148, 2017. Citado na página 17.

GOEL, R. *E-commerce*. [S.l.]: New Age International, 2007. Citado na página 16.

GUISSONI, L. A.; FARINHA, R. L. E-commerce com resultado. *GV-executivo*, v. 18, n. 1, p. 40–42, 2019. Citado na página 16.

LAGHMARI, G. et al. Comparison shopping engines state of the art, exposure of shortcomings and discussion of new innovations. 01 2019. Citado 2 vezes nas páginas 19 e 20.

MEHAK, S. et al. Exploiting filtering approach with web scrapping for smart online shopping: Penny wise: A wise tool for online shopping. In: IEEE. *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. [S.l.], 2019. p. 1–5. Citado na página 20.

OBE, R. O.; HSU, L. S. *PostgreSQL: up and running: a practical guide to the advanced open source database*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado na página 52.

PAULA, A. F. d. Socialmarket: aplicativo colaborativo para auxiliar nas compras em supermercados. Trabalho de Conclusão de Curso – Universidade Federal de Ouro Preto, 2018. Citado na página 19.

RATHOD, U.; PAVATE, A.; PATIL, V. Product rank based search engine for e-commerce. In: IEEE. *2018 3rd International Conference for Convergence in Technology (I2CT)*. [S.l.], 2018. p. 1–5. Citado 2 vezes nas páginas 20 e 21.

SALES, C. A. C. Impacto da crise econômica no comportamento do consumidor de supermercados. *CONNEXIO-ISSN 2236-8760*, v. 6, n. 2, p. 11–20, 2017. Citado na página 16.

SILVA, W. M. da et al. Marketing digital, e-commerce e pandemia: uma revisão bibliográfica sobre o panorama brasileiro. *Research, Society and Development*, v. 10, n. 5, p. e45210515054–e45210515054, 2021. Citado na página 16.

SMITH, M. D.; BRYNJOLFSSON, E. Consumer decision-making at an internet shopbot: Brand still matters. *The Journal of Industrial Economics*, Wiley Online Library, v. 49, n. 4, p. 541–558, 2001. Citado na página 20.

SOMMERVILLE, I. *Engenharia de software*. 10. ed. Pearson Education, 2019. ISBN 9788543024974. Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Loader/168127/pdf>. Citado 4 vezes nas páginas 28, 29, 36 e 51.

WAN, Y. *Comparison-shopping services and agent designs*. [S.l.]: IGI Global, 2009. Citado 2 vezes nas páginas 19 e 20.

ZHAO, B. Web scraping. *Encyclopedia of big data*, Springer Living ed. Cham, p. 1–3, 2017. Citado na página 21.

Apêndices

APÊNDICE A – Tecnologias

Este apêndice contempla as principais tecnologias e ferramentas utilizadas para o desenvolvimento do trabalho. O versionamento e listagem desses recursos é dado a seguir:

- Angular 12.2.13
- Express 4.17.1
- Node.js 14.18.1
- PostgreSQL 12.11
- pgModeler 0.9.4
- Auth0 API v2

A.1 *Frameworks*

Os *frameworks* se popularizaram no desenvolvimento de aplicações, por conseguirem facilitar a construção do *software*, ao mesmo tempo que permitem um trabalho mais organizado e com qualidade. Para Sommerville (2019), "um *framework* é uma estrutura genérica estendida para criar um subsistema ou aplicação mais específicos", por meio deles é possível criar uma estrutura para o projeto que serve de guia para o desenvolvimento.

A.1.1 Angular

Angular[1] é uma plataforma moderna baseada na linguagem *TypeScript*, com suporte à requisições *Hypertext Transfer Protocol* (HTTP) e *Hyper Text Transfer Protocol Secure* (HTTPS), com roteamento integradas diretamente no *framework*, uma arquitetura baseada em componentes ao invés da abordagem *Model View Controller* (MVC), seguindo o formato de *Single Page Application* (SPA). De maneira geral, é um *framework* popular e consolidado para desenvolvimento de aplicações *web*.

Na construção do *frontend web*, foram utilizados os recursos gráficos do Angular Material[2], uma biblioteca oficial da plataforma que disponibiliza componentes prontos e estilizados. Notavelmente, também se fez o uso da biblioteca Bootstrap[3], para organização dos elementos em *HyperText Markup Language* (HTML) e estilização com *Cascading Style Sheets* (CSS).

[1] Disponível em: <https://angular.io/>
[2] Disponível em: <https://material.angular.io/>
[3] Disponível em: <https://getbootstrap.com/>

A.1.2 Express

Express[4] é um *framework web* rápido e flexível construído em cima da plataforma do Node.js[5], que permite a execução de códigos JavaScript fora do ambiente dos navegadores. Possui uma variedade de métodos e *middlewares*, que facilitam a criação de APIs robustas.

No trabalho, este *framework* foi utilizado para construção do *backend* da aplicação, responsável pelos *endpoints* de comunicação e conexão com o banco de dados. Esse tipo de organização permite que, caso existam outros *frontends* além do citado na subseção A.1.1, eles consumam os mesmos *endpoints* disponibilizados no *backend*.

A.2 Banco de Dados

O banco de dados é a parte da aplicação responsável pelo armazenamento de todos os registros, utilizados durante as operações do sistema. As ferramentas, funções e complementos disponíveis dependem exclusivamente da plataforma escolhida. Neste cenário, se optou pelo uso do PostgreSQL[6], um Sistema de Gerenciamento de Banco de Dados (SGBD) relacional de código aberto, que emprega diversos recursos avançados e é facilmente expansível (OBE; HSU, 2017).

A escolha deste SGBD para o projeto, está ligado a popularidade e a ampla gama de recursos oferecidos, principalmente em questão de integração com o sistema, além da afinidade do autor com esta tecnologia. Em um momento futuro, pode ser interessante buscar uma abordagem não-relacional, para suportar a entrada de dados contínua do sistema.

Juntamente do banco, também foi utilizada a ferramenta pgModeler[7], um *software* que permite a modelagem dos objetos e relações do banco de dados, como tabelas, cardinalidade, funções, procedimentos, tipos, dentre muitos outros. Por se tratar de um recurso de código aberto, é possível compilar o executável para uso a partir do seu repositório.

A.3 Autenticação

A autenticação de usuários de qualquer sistema, é um ponto crítico que precisa de atenção, pois se entende que um erro de implementação pode por em riscos informações sensíveis. Buscando uma abordagem mais segura e simplificada, foi optado pelo uso da

[4] Disponível em: <https://expressjs.com/>
[5] Disponível em: <https://nodejs.org/en/>
[6] Disponível em: <https://www.postgresql.org/>
[7] Disponível em: <https://pgmodeler.io/>

plataforma de autenticação Auth0[8], que implementa o protocolo OAuth 2.0 descrito pela RFC-6749[9].

Uma das vantagens em utilizar uma ferramenta externa, para lidar com a autenticação, é não se preocupar em guardar senhas dos usuários, visto que tudo é feito por uma plataforma confiável. Essa aplicação também permite o *login* com contas de outros domínios, como da Google.

A.4 Ambiente de desenvolvimento

Foi escolhido o uso de um sistema operacional baseado em Linux, para compilação e execução do código, permitindo que a maior parte dos recursos sejam instalados via linha de comando. O desenvolvimento foi mantido dentro de um ambiente Windows por meio do recurso de Subsistema Windows para Linux (WSL)[10].

Os navegadores escolhidos como foco do sistema *web*, foram o Google Chrome e Mozilla Firefox, respectivamente devido a popularidade e pelas ferramentas de depuração avançadas. O controle do versionamento e repositórios de código para cada parte do sistema foram feitos pelo GitHub[11].

[8] Disponível em: <https://auth0.com/>
[9] Disponível em: <https://datatracker.ietf.org/doc/html/rfc6749>
[10] Disponível em: <https://docs.microsoft.com/pt-br/windows/wsl/about>
[11] Disponível em: <https://github.com/>

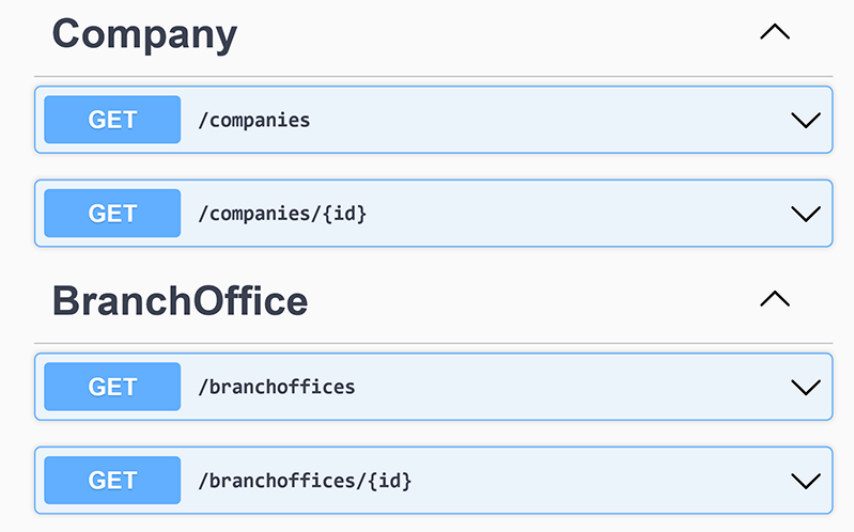
APÊNDICE B – *Endpoints* da API Rest

Durante o desenvolvimento, foi construída uma documentação dos *endpoints* da API REST utilizando o Swagger[1]. Esta ferramenta consegue identificar as rotas criadas no projeto, gerando a documentação com base nas configurações definidas. Os *endpoints* listados a seguir são encontrados na aplicação de *backend*.

B.1 Empresa e filial

A Figura 17, mostra os *endpoints* de empresa e de filial. Para os dois casos, existe uma rota para retornar todos as entradas do banco, e uma para retornar dados pertencentes a um registro específico. Não existem *endpoints* para os verbos POST, PUT ou DELETE, pois não é esperado que essas informações sejam inseridas ou alteradas com frequência, visto que as tabelas envolvidas também afetam as operações de raspagem de dados.

Figura 17 – *Endpoints* de empresa e filial



Fonte: Elaborado pelo autor

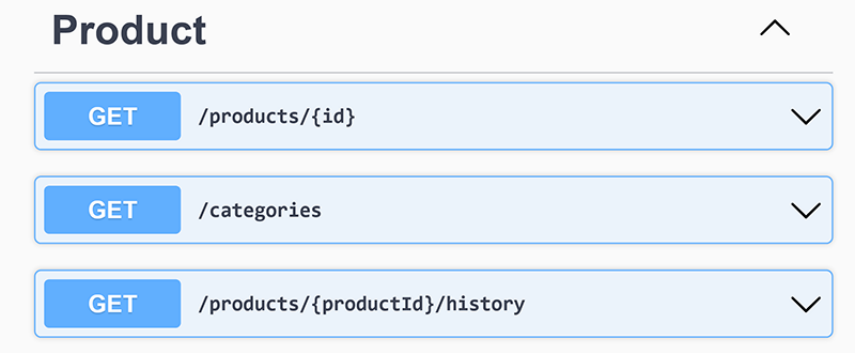
B.2 Produto

Os *endpoints* referentes aos produtos individuais de cada filial, são listados na Figura 18. Respectivamente, existe uma rota para trazer os dados de um produto específico,

[1] Disponível em: <https://swagger.io/>

outra para recuperar a lista de categoria dos produtos do sistema, e por fim, uma que lista todo o histórico de preço de um determinado produto. Não foram feitas rotas para alterar ou inserir produtos, pois a entrada destes dados são de responsabilidade do robô.

Figura 18 – *Endpoints* de produto



Fonte: Elaborado pelo autor

B.3 Grupo de produtos

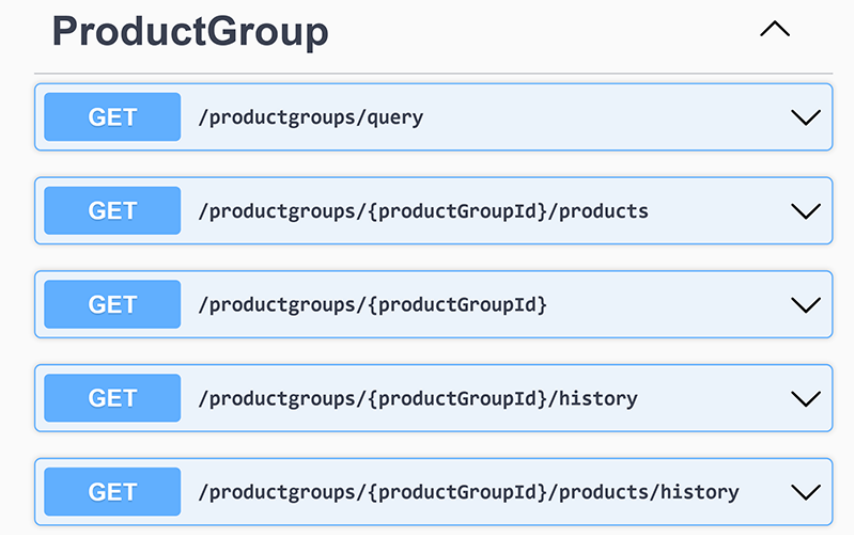
Os grupos de produtos são os principais objetos da regra de negócio do sistema, seus *endpoints* são mostrados na Figura 19. A primeira rota é responsável pela pesquisa de produtos na página inicial, a segunda retorna todos os produtos que pertencem a um determinado grupo, enquanto a terceira traz informações sobre um grupo de produtos específico. O quarto *endpoint* lista o histórico de menores preços de um grupo, e por fim, a última rota traz o histórico de preço de todos os produtos do grupo em questão.

Assim como nas documentações anteriores, não existem maneiras de alterar os dados referentes aos grupos de produtos. Os grupos são gerados de maneira automática, utilizando a lógica discutida na subseção 3.4.2. Dessa maneira, as alterações ocorrem de acordo com a entrada de dados.

B.4 Usuário

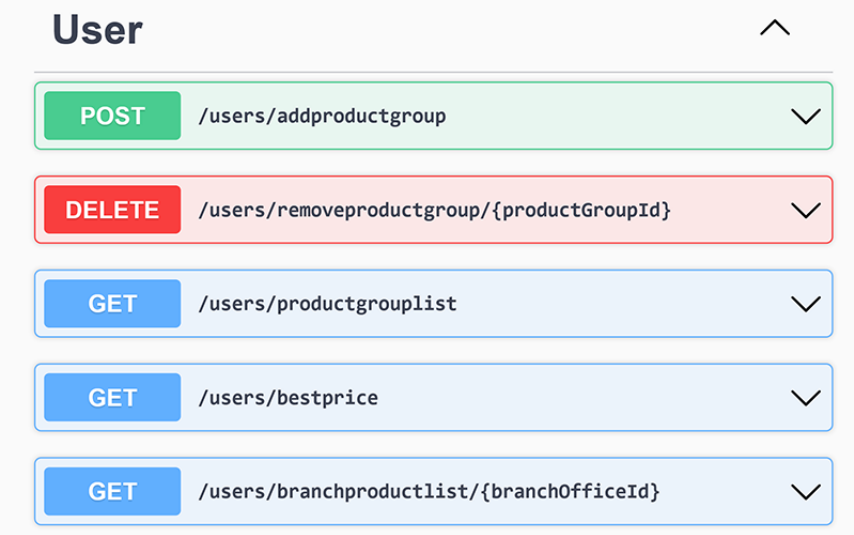
Os *endpoints* de usuário são protegidos pelo *middleware* de autenticação, isto é, todas necessitam de um *token* para validar o processo. A lista das operações disponíveis é encontrada na Figura 20. As duas primeiras rotas são responsáveis por adicionar e remover produtos da lista pessoal do usuário, enquanto a terceira retorna a lista propriamente dita. O quarto *endpoint* devolve uma lista ordenada de supermercados, com base na disponibilidade e menores preços da lista de itens do utilizador. Por fim, a última rota retorna a lista de preços dos produtos do usuário, para uma filial específica.

Figura 19 – *Endpoints* de grupo de produtos



Fonte: Elaborado pelo autor

Figura 20 – *Endpoints* de usuário



Fonte: Elaborado pelo autor

Em um ambiente em nuvem, a inserção de usuários no banco de dados seria uma ação de responsabilidade da plataforma Auth0, no momento da primeira autenticação. Não foram feitas rotas para realizar inserções ou alterações, com exceção da lista de compras. Por meio do *token* de autenticação recebido e validado pelo *backend*, é possível distinguir qual usuário fez a requisição.