



CENTRO UNIVERSITÁRIO FUNVIC



**Mateus Camargo Garcia de Siqueira
Mauricio Camargo Garcia de Siqueira
Rafael Bruno dos Santos Costa**

**DESENVOLVIMENTO DE UM SOFTWARE COMPARADOR DE
OFERTAS DE PRODUTOS DE TECNOLOGIA USANDO O WEB
SCRAPING**

**Pindamonhangaba-SP
2022**



CENTRO UNIVERSITÁRIO FUNVIC



**Mateus Camargo Garcia de Siqueira
Mauricio Camargo Garcia de Siqueira
Rafael Bruno dos Santos Costa**

DESENVOLVIMENTO DE UM SOFTWARE COMPARADOR DE OFERTAS DE PRODUTOS DE TECNOLOGIA USANDO O WEB SCRAPING

Artigo científico apresentado como parte dos requisitos para obtenção do Diploma de Bacharelado em Engenharia de Computação pelo Curso de Engenharia de Computação do Centro Universitário FUNVIC.

Orientador(a): Prof. Esp. Rodrigo Ramos de Oliveira

**Pindamonhangaba-SP
2022**

Siqueira, Mateus Camargo Garcia de; Siqueira, Mauricio Camargo Garcia de; Santos
Consta, Rafael Bruno dos

Desenvolvimento de um software comparador de ofertas de produtos de tecnologia
usando a web *scraping* / Mateus Camargo Garcia de Siqueira; Mauricio Camargo
Garcia de Siqueira; Rafael Bruno dos Santos Costa / Pindamonhangaba-SP :
UniFUNVIC Centro Universitário FUNVIC, 2022.

11f.: il

Artigo Científico (Graduação em Engenharia da Computação) UniFUNVIC-SP.

Orientador: Prof. Esp. Rodrigo Ramos de Oliveira.

1 Web Scraping. 2 BeautifulSoup. 3 Raspagem de Dados.

I Desenvolvimento de um software comparador de ofertas de produtos de tecnologia
usando a web scraping

II Mateus Camargo Garcia de Siqueira; Mauricio Camargo Garcia de Siqueira; Rafael
Bruno dos Santos Costa.

MATEUS CAMARGO GARCIA DE SIQUEIRA
MAURICIO CAMARGO GARCIA DE SIQUEIRA
RAFAEL BRUNO DOS SANTOS COSTA

**DESENVOLVIMENTO DE UM SOFTWARE COMPARADOR DE OFERTAS DE
PRODUTOS DE TECNOLOGIA USANDO O WEB SCRAPING**

Artigo científico apresentado como parte dos requisitos para obtenção do Diploma de Bacharelado em Engenharia de Computação pelo Curso de Engenharia de Computação do Centro Universitário FUNVIC.

Data: 07/12/2022

Resultado: _____

BANCA EXAMINADORA

Prof. Esp. Rodrigo Ramos de Oliveira _____ Centro Universitário FUNVIC - UniFUNVIC

Assinatura: _____

Prof. Dr. Dailton de Freitas _____ Centro Universitário FUNVIC - UniFUNVIC

Assinatura: _____

Prof. Dr. Claudio Augusto Kelly _____ Centro Universitário FUNVIC - UniFUNVIC

Assinatura: _____

Este trabalho foi escrito na forma de artigo científico a ser submetido à revista eletrônica de ciência exatas ISSN 2675-8997, cujas normas estão em anexo (ANEXO A). A parte textual corresponderá ao artigo científico escrito conforme a instrução da revista escolhida, incluindo página de título e autores com suas respectivas vinculações.

**DESENVOLVIMENTO DE UM SOFTWARE COMPARADOR DE OFERTAS DE
PRODUTOS DE TECNOLOGIA USANDO O WEB SCRAPING**
DEVELOPMENT OF A COMPARATOR SOFTWARE OF TECHNOLOGY PRODUCT
OFFERS USING WEB SCRAPING

Mauricio Camargo Garcia de Siqueira¹, Mateus Camargo Garcia de Siqueira¹, Rafael Bruno dos Santos Costa¹, Rodrigo Ramos de Oliveira²

¹Bacharelado em Engenharia da Computação pelo Centro Universitário FUNVIC,
Pindamonhangaba-SP

²Docente do Centro Universitário FUNVIC, Pindamonhangaba-SP.

*mauricio.siqueira2013@hotmail.com

Resumo

A demanda por produtos eletrônicos nunca esteve tão alta como está atualmente. Com isso, a prática de comparação de preços de produtos é algo que faz parte do cotidiano de muitos compradores, auxiliando a terem a melhor economia possível durante uma compra. Em decorrência desses fatores, as ferramentas comparadoras de ofertas, objeto de estudo deste trabalho, vem se tornando a principal escolha dentre os consumidores ao realizarem uma compra. Nesse contexto, este trabalho tem por objetivo o desenvolvimento de uma ferramenta comparadora de ofertas, por meio de um *website*. Para o *website*, utilizou-se a linguagem de marcação HTML e do CSS. O algoritmo *scraper* foi programado em *Python*, fazendo uso da biblioteca *Beautiful Soup* para extração dos dados e da biblioteca *Selenium* para automatização do navegador. E para a integração do *Python* com o HTML da página, foi utilizado a biblioteca *Flask* do *Python*. Os resultados obtidos com o protótipo demonstram a eficiência da ferramenta em automatizar a busca pelos preços, sendo possível comparar com os resultados encontrados, de modo a apoiar o consumidor na escolha de sua compra. Dentre os objetivos propostos pelo trabalho, não foi possível alcançar todos, com tudo, o objetivo geral do trabalho foi alcançado, tendo sido criado o protótipo de uma ferramenta comparadora de preços.

Palavras-chave: *Web Scraping. Beautiful Soup. Automação de Coleta de dados. Comparador de Ofertas.*

Abstract

The demand for electronic products has never been as high as it is today. As a result, the practice of comparing product prices is something that is part of the daily life of many buyers, helping them to have the best possible savings during a purchase. As a result of these factors, the offer comparison tools, object of study of this work, have become the main choice among consumers when making a purchase. In this context, this work presents the development of a prototype, of a tool to compare offers, through a website. For the website, HTML markup language was used for the structure of the site, and CSS for its styling. The scraper algorithm was programmed in Python, using the Beautiful Soup library for data extraction and the Selenium library for browser automation. For the integration of Python with the HTML of the page, the Python Flask library was used. The results obtained with the prototype demonstrate the efficiency of the tool in automating the search for prices, being possible to compare with the results found, in order to support the consumer in choosing his purchase.

Keywords: *Web Scraping. Beautiful Soup. Data Collection Automation. Offers Comparator.*

Introdução

Pesquisas apontam as categorias de produtos que os consumidores mais procuram na *internet*, no qual a categoria eletrônica (computadores, *notebooks* e *smartphones*) aparece em primeiro, logo após, equipamentos de *hardwares* e games (placa de vídeo, processador e placa mãe) (FERNANDES, 2019). Em 2021 foi possível ver o reflexo causado por este cenário, no qual, houve uma escassez global de semicondutores, ao lado da alta do dólar, o que resultou em um aumento de 40% nos preços de *notebooks* em 12 meses (DINO, 2021).

As pessoas, ao efetuarem uma compra, naturalmente buscam pelos melhores preços e, por esse e outros motivos, as ferramentas de ofertas têm ganhado mais espaço e importância para os consumidores. Segundo Fernandes (2019), quando se trata de melhor condição de compra, 40% das pessoas entrevistadas utilizam as ferramentas que comparam preços em diversos *sites* e oferecem as condições mais vantajosas, tais como Buscapé e Zoom, sendo esta a alternativa mais comum dentre os entrevistados. Prosseguindo com os resultados, obter referência com especialista aparece com 22%, seguido por 11% das pessoas que procuram em redes sociais e 10% que dizem buscar considerações de amigos e familiares. Analisando esses resultados, é possível verificar que existe grande parcela de pessoas que não buscam comparar por preços em diversos *sites* na hora de comprar um produto, aumentando assim, as chances de não encontrarem a melhor oferta.

Além deste aumento, a pandemia causada pelo isolamento social trouxe uma nova realidade pouco explorada pelas empresas, o trabalho em *home office*, que tornou-se uma tendência mesmo após a pandemia (JANKAVSKI, 2021). Diante dessa nova realidade, as pessoas precisaram adquirir novos equipamentos, aumentando o comércio eletrônico em mais de 67% nos últimos anos. Ainda, com o comércio local fechado devido à pandemia, as pessoas se viram forçadas a comprar pela *internet* (G1, 2022).

Diante do exposto, com aumento dos preços dos produtos eletrônicos (da tecnologia da informação), e o crescimento das compras pela *internet*, notou-se uma oportunidade para projetar e implementar uma aplicação *web*.

A *web scraping* será a técnica central da aplicação. É uma técnica que se baseia em coletar os dados de uma determinada página da *Web*, tratando esses dados da maneira que for necessário, para posteriormente implementar um uso funcional para estes dados. Pode ser feita de maneira manual, mas na maioria das vezes é preparado um método de extração que engloba todo o conjunto de dados pertinentes, tornando tudo mais eficiente e automatizado (BROUCKE; BAESSENS, 2018).

Segundo Broucke e Baesens (2018), o processo se inicia com a identificação do *site* que contém os dados a serem coletados, tendo essa informação a ferramenta consegue ler e capturar todo o código HTML (*HyperText Markup Language*) da página da maneira com que foi desenvolvido. Tendo

o HTML da página, são criados os métodos de extração com as ferramentas específicas, desenvolvidas para localizar e coletar os dados. Por fim esses dados são salvos no formato mais pertinente ao desenvolvedor, como HTML por exemplo.

HTML é uma linguagem de marcação de hipertexto, ou seja, ele tem o objetivo de marcar e indicar onde cada elemento presente em uma página *web* ou em uma aplicação irão aparecer. O HTML é principalmente usado em desenvolvimento *frontend* (parte visual de um site), dando a estrutura base do site e realizando as marcações para os navegadores. (OKUBO, 2022). Junto a ele é utilizado o CSS (*Cascading Style Sheets*) para dar estilo para o código criado em HTML. De forma prática, seria a camada de personalização ao conteúdo visível segundo Okubo (2021).

Python é uma linguagem de programação orientada a objetos, de alto nível, com semântica dinâmica. É uma linguagem que se destaca pela simplicidade, suporta módulos e pacotes que encoraja a programação modularizada e reuso de código. É uma das linguagens que mais cresceu nos últimos anos, devido a sua compatibilidade e capacidade de auxiliar outras linguagens (ALURA, 2022).

O *Beautiful Soup* é a biblioteca *Python* que permite navegar em arquivos HTML e XML, a fim de encontrar e extrair os dados. Funciona com seu interpretador (*parser*) que faz a leitura do arquivo e fornece maneiras intuitivas de buscar com os métodos de *find*, navegar e modificar uma árvore de análise (*parse tree*). Esta biblioteca geralmente economiza horas ou dias de trabalho de programadores ao redor do mundo (RICHARDSON, 2015). E, muito semelhante ao *Beautiful Soup*, existe a biblioteca *Selenium* no *Python*, com o propósito de encontrar elementos na página *Web* e coletá-los, mas existe uma diferença, o *Selenium* permite também interagir com esses elementos, controlar uma página *Web* de forma automatizada. O *Selenium webdriver* é um dos elementos que contem no *selenium*, é por meio de seus métodos que automatizamos o navegador. O *webdriver* seria justamente o aplicativo especial que controla o navegador, mas existe um driver específico para cada navegador, junto as suas características necessárias para utilizar seus métodos (PAULO, 2022). Ambas as bibliotecas são muito utilizadas e incorporadas em métodos de extração de dados na *Web*.

Para integrar essas duas linguagens que se difere, HTML e *Python*, existe um micro *framework* de desenvolvimento *web* chamado *Flask*, uma biblioteca *Python* muito popular no mercado. Marcada pela sua simplicidade e robustez, *Flask* é estendido, precisando instalar apenas o que realmente for utilizado, diferentemente do Django que vem com uma série de componentes pré-instalados (ALURA, 2022).

Em resolução ao que foi dito anteriormente, este trabalho tem por objetivo desenvolver uma aplicação capaz de comparar os preços de produtos eletrônicos, referente a determinada pesquisa feita pelo usuário, informando os preços encontrados nos *sites*, orientando o consumidor à uma compra mais econômica e independente.

Método

Primeiramente foi realizada uma pesquisa em *sites* especializados, em artigos, monografias, livros e periódicos, a fim de ampliar o conhecimento dos conceitos e ferramentas a serem utilizados, principalmente das bibliotecas disponibilizadas pela linguagem de programação *Python*, dentre elas *beautiful soup*, empregada no desenvolvimento do algoritmo de extração para a comparação de dados.

Para extrair os preços dos *sites* foi usada a técnica de *web scraping*, que consiste em minerar os dados em um *site* específico, de forma a utilizar toda a informação coletada para comparar com os preços dos outros *sites*, obtendo a melhor oferta na compra.

Os *sites* que foram selecionados para integrarem ao projeto e que tiveram os preços dos produtos coletados para fins de comparação e escolha do usuário, foram *sites* considerados seguros e íntegros com os dados dos visitantes. Apenas os *sites* que atendem a Lei Geral de Proteção de Dados (LGPD, Lei nº 13.709/2018) foram levados em conta (PLANALTO, 2018). Além disso, foi analisado também o histórico de vazamento do *site*, bem como o nível de satisfação dos clientes ao realizarem a compra. Para isto, foi disposto de um tempo navegando pelo *site* a fim de procurar os comentários dos clientes que tiveram alguma experiência de compra com o *site*, verificando se eram positivos ou negativos, a quantidade de cada um deles, e qual o nível de insatisfação com a experiência. Feito essa pesquisa, foi possível selecionar os sites com as melhores avaliações e estes foram utilizados para a comparação dos preços.

Foi feito também um estudo sobre as ferramentas que fizeram parte da construção da aplicação. Partindo da interface que foi desenvolvida, levou-se em consideração a 8ª heurística de Nielsen no qual diz respeito a estética e design da aplicação (PINHATA, 2022). As linguagens de programação utilizadas para a *front-end* foi o HTML na versão 5, CSS e Javascript, e para a *back-end Python*, sendo este o responsável de ter desenvolvido o *Script* de coleta de dados.

Para o desenvolvimento do *Script* da coleta de dados funcionar foi dividido em duas partes, sendo a primeira desenvolvida com a biblioteca *Selenium*, com o objetivo de automatizar o navegador para inserir as credenciais no site da Kabum. O código começa recebendo o método *Options*, que no *Selenium* seria para configurar o navegador virtual, seguido de seu acionamento junto a opção *headless*, onde *false* torna o navegador visível no momento da execução, e *true* oculta todo o processo do navegador virtual. Foi uma opção muito útil no momento do desenvolvimento, pois permitiu ter o acompanhamento do funcionamento do navegador na parte de criação do código. Em seguida foi instanciado um objeto do *webdriver*, utilizando como navegador o *Firefox*. Tendo acessado ao link de login, foi feito a busca pelos campos de usuário e senha por meio do método *find_element*, sendo possível inserir os acessos com o método *send_keys*. Efetuado o login na página já seria possível enviar o HTML para o Beautiful Soup, mas antes, foi usado mais uma vez os métodos para encontrar um elemento, mas agora no campo de busca do site enviando o nome do produto digitado pelo usuário

como parâmetro de busca, só então que todo o HTML da página navegada pelo webdriver será enviada para o BeautifulSoup por meio do método `page_source`, é então que se inicia a segunda parte do algoritmo de coleta. O código começa com a criação do objeto *soup*, fornecendo o HTML da página para o interpretador *parser*. Logo em seguida foi usado o método *findAll* para encontrar todos os produtos da página, caso dê algum erro ao processar o código o tratador de exceções vai capturá-lo, exibindo uma mensagem de erro. Em seguida, foi feito um *for* para coletar os dados dos produtos a serem exibidos na página, seria basicamente um filtro. Essas informações são armazenadas em uma lista, e enviadas para a página HTML por meio do *micro-framework Flask*. Por meio do *Flask* foi possível integrar o algoritmo de extração de dados feita no *Python* junto a página em HTML, a versão utilizada foi a 2.2.2.

Para a execução do algoritmo *scraper* foi utilizado a versão 3.9.0 do *Python*, por meio do IDE PyCharm, que oferece recursos como detecção de erros de código e autopreenchimento de diversos comandos, tornando toda escrita do algoritmo mais eficiente. *Python* é uma linguagem de programação de alto nível, orientada a objetos e de tipagem dinâmica, além de contar com recursos poderosos de suas bibliotecas e módulos (CUNHA, 2018).

Junto ao desenvolvimento do projeto foi dedicado um tempo todos os dias para a realização da documentação. Por fim, tendo tudo pronto, realizou-se uma inspeção em toda a documentação a fim de encontrar erros. Com a aplicação teve-se o mesmo cuidado, realizando o mesmo procedimento por todo o código, mas por questões de prazo foi deixado de lado a busca por melhorias ou grandes modificações, deixando este para projetos futuros. Foi feito também baterias de testes com o objetivo de encontrar algum erro que pudesse comprometer a aplicação. Os testes realizados foram:

- Teste de funcionalidade: Foi testado todos os botões da página, a fim de verificar se estavam fazendo tudo da maneira que foi planejado.
- Teste de busca: Foi dividido em duas partes. Na primeira foi feita a busca por diversos produtos relacionados a tecnologia da informação. Já a segunda parte, foi feito buscas com produtos que não tinham nenhuma relação com a área da tecnologia da informação, a fim de observar qual seria o resultado exibido.
- Teste de redirecionamento: Por último realizou-se o teste de redirecionamento, a fim de verificar se ao clicar sobre o produto iria encaminhar corretamente para a página em que o produto estaria disponível para compra.

Resultados

Na Figura 1 é mostrado um trecho da programação, o código utilizado para a automatização do navegador *Firefox*, que teve como objetivo efetuar *login* no *site* da Kabum.

Figura 1 – Efetuando Login com a Ferramenta *Selenium*

```

try:
    options = Options()
    options.headless = False # executar de forma visível (False) ou oculta (True)

    # Entrando no site
    browser = webdriver.Firefox(options=options)
    url = 'https://www.kabum.com.br/login'
    browser.get(url)

    # Colocando NOME e SENHA
    username = browser.find_element(by=By.XPATH, value='//*[@id="__next"]/main/div/div/div/div/form/div/div[1]/div/div/input')
    username.send_keys('mateus.cg.siqueira@hotmail.com')
    sleep(1)
    password = browser.find_element(by=By.XPATH, value='//*[@id="__next"]/main/div/div/div/div/form/div/div[2]/div/div/input')
    password.send_keys("*****")
    sleep(1)
    login_attempt = browser.find_element(by=By.XPATH, value='//*[@id="__next"]/main/div/div/div/div/form/div/button')
    login_attempt.submit()
    sleep(5)

    campo_busca = browser.find_element(by=By.ID, value='input-busca')
    campo_busca.send_keys(procura)
    campo_busca.submit()
    sleep(5)

    htmlcontent = browser.page_source
    sleep(2)

```

Na Figura 2 é possível ver um outro trecho do código utilizado para a extração de dados, com o uso da biblioteca *Beautiful Soup*, logo após receber todo o HTML do *webdriver*.

Figura 2 – Função Scraping com o *Beautiful Soup*

```

soup = BeautifulSoup(htmlcontent, "html.parser")

produtos = soup.findAll('div', attrs={'class': 'sc-ff8a9791-7 dZlrn productCard'})

content = produtos

except:
    flash('Failed to retrieve URL "%s" % url, 'danger')
    content = ''

lista_produtos = []
for produto in produtos:
    # nome do produto
    nome_prod = produto.find('span', attrs={'class': 'sc-d99ca57-0 iRparH sc-ff8a9791-16 kRYNji nameCard'})
    # preço produto
    real = produto.find('span', attrs={'class': 'sc-3b515ca1-2 jIvomc priceCard'})
    # link produto
    link = produto.find('a', attrs={'class': 'sc-ff8a9791-10 cUkkYL'})
    # link imagem produto
    link_imagem = produto.find('img', attrs={'class': 'imageCard'})

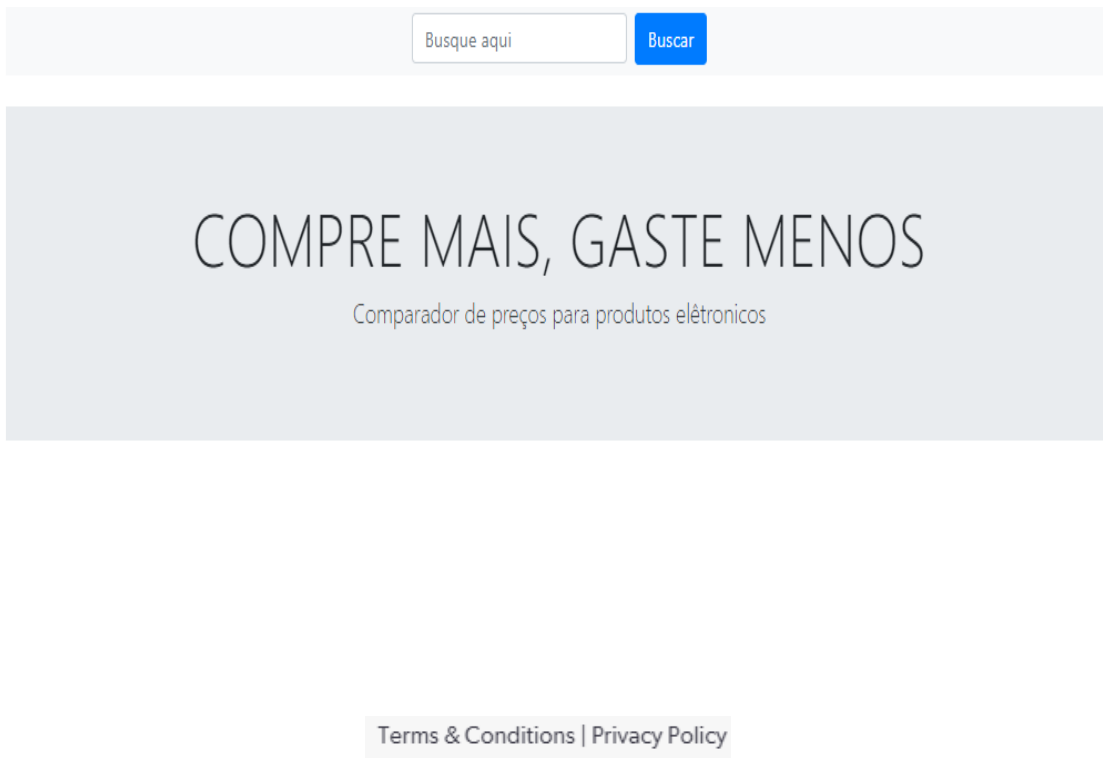
    lista_produtos.append(
        [nome_prod.text, real.text, 'https://www.kabum.com.br' + link['href'], link_imagem['src']]
    )

browser.quit()

```

Na Figura 3 o usuário pode inserir sua busca no campo de pesquisa, “Busque aqui”. Esta é a página inicial do *site*, e pode-se observar o campo de pesquisa, onde o usuário vai digitar a informação desejada. Será exibido os produtos em uma janela como é possível observar na Figura 4 logo em seguida.




Figura 3 – Página Inicial



Como demonstrado na Figura 4, uma janela exibe os produtos buscados, de acordo com o que foi pesquisado pelo usuário, sendo distribuído em três colunas, “Imagem do produto”, “Preço do produto” e “Nome do produto”, organizando a exibição em ordem crescente, do produto mais barato ao produto mais caro, exibindo uma quantidade de 20 itens na página. Neste seguimento, para o funcionamento do *site*, já é necessário que a função do *web scraping* tenha realizado a coleta de dados.

Além das informações exibidas, para cada produto existe um botão chamado de copiar *link*, e cada botão corresponde ao *link* para o *site* em que o produto foi ofertado, para quando o usuário se interessar em comprar o produto, acessar diretamente na loja *online*.

Figura 4 – Exibição dos Produtos em Oferta

	Preço do Produto	Nome do Produto
	À vista no PIX R\$ 299.99	Monitor Prizi 17 HD LED, 60Hz, HDMI, VESA, Ajuste de Ângulo, Preto - PZ0017HDMI Copiar Link
	À vista no PIX R\$ 399.99	Monitor Concordia PCFort 19.5 LED, HD, 60Hz, 5ms, HDMI e VGA, VESA, Ajuste de Ângulo - Z195 Copiar Link
	À vista no PIX R\$ 599.99	Monitor LG 19.5' LED, HD, HDMI, VESA, Ajuste de Ângulo - 20MK400H-B Copiar Link

Na Figura 5 podemos verificar no código fonte, como é realizado a forma de exibição dos resultados. No código é usado as *tags table* para montar a tabela, junto as *tags* “tr” e “th” para estruturar as linhas e colunas da tabela, exibindo as imagens dos produtos com seus respectivos preços e nomes.

Figura 5 - Código HTML: Exibindo os Resultados tabulados da extração feita.

```

<div class="container">
  <div class="row">
    <!-- Download form -->
    <div class="col">
      <!-- Download button-->
      <button class="btn btn-success col-sm mt-2" type='button' onclick="exportTableToCSV('results.csv')">0fertas</button>
    </div>

    <!-- Results table-->
    <table class="table mt-3" id="tags">
      <thead>
        <tr>
          <th scope="col"></th>
          <th scope="col"></th>
          <th scope="col">Preço do Produto</th>
          <th scope="col">Nome do Produto</th>
        </tr>
      </thead>
      <tbody>
        {% for nome, preco, link_prod, link_img in results %}
        <tr>
          <td></td>
          <td></td>
          <td>À vista no PIX {{ preco }}</td>
          <td>{{ nome }}</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
</div>

```

Discussão

A princípio, não seria utilizado a biblioteca *Selenium* para a implementação da aplicação, pois quando efetuado requisições, em alguns sites, ocorria um bloqueio, e isso por conta de não estar com o acesso de usuário naquele site, uma maneira de proteger os dados da extração, permitindo apenas, extrair dados com o perfil conectado. O *Selenium* acabou sendo uma alternativa muito válida para o processo de *login*, atendeu a necessidade e ainda ajudou a efetuar a busca pelo produto requisitado pelo usuário. O único problema que apresentou, está relacionado ao seu desempenho, pois o navegador virtual demandou mais tempo do que o *Beautiful Soup* para alcançar o HTML da página. Em testes feitos durante o desenvolvimento, pode-se concluir que este problema se deve a quantidade de atividades e requisições a serem realizadas, uma vez que, para ter acesso a página o *selenium* precisa esperar o carregamento de todas as ferramentas do *site* em questão, como o *JavaScript* por exemplo, em seguida, identificar os elementos, enviar os acessos e entrar no perfil de usuário, para por fim, ter acesso a página, enquanto que para o *Beautiful Soup* basta fazer uma requisição da página por meio do método *request*, copiando toda a fonte da página (PORNARAS, 2021). Mas no geral, ambos trabalharam muito bem juntos, sendo possível alcançar o objetivo, extraíndo os dados da página necessários para este trabalho.

Por conta de mais problemas enfrentados, agora no redirecionamento do produto, teve de ser implementado um novo recurso, o botão de copiar link, ele foi criado para que o consumidor consiga ter acesso à página que ofertou o produto. Pelo mesmo motivo citado anteriormente, alguns *sites* bloquearam o redirecionamento, não sendo possível acessar a página, isso foi verificado durante os testes de redirecionamento, estes sites identificaram o acesso como uma ameaça, no qual permitia o acesso apenas com as credenciais inseridas, porém iria ficar inviável para a aplicação manter as credenciais gravadas durante o redirecionamento. Na tela aparecia o erro 403, e esse código se enquadra na faixa de acessos, no qual este em específico, informa que não foi possível acessar o endereço, por conta de não ter acesso (HOSTINGER, 2022). O acionamento deste erro se da pela tecnologia implementada no *site*, os *anti-bots*, presentes em sites robustos a fim de proteger contra trapagens comportamentais, hackers, roubo de dados e etc. (ANTIBOT, 2022).

Como já citado, os testes de redirecionamento não ocorreram como o esperado, por outro lado, os testes de funcionalidades obtiveram o resultado esperado, todos os botões funcionaram corretamente como deveriam. Já em relação ao teste de busca, foi obtido um resultado parcialmente satisfatório. Quando pesquisado por produtos relacionados a tecnologia da informação ocorreu tudo como deveria, buscando os produtos corretamente, porém quando essa busca era feita com produtos que não tinham em nenhuma das lojas de oferta, ele retornava produtos aleatórios.

Dentre os pontos positivos, a linguagem de programação *Python* mostrou-se ser muito eficiente, na qual prioriza a velocidade, pois requer uma quantidade muito menor de código necessária para executar as funções do que quando programadas em outra linguagem. Um dos princípios no qual o *Python* foi desenvolvido é “Simples é melhor do que complexo”, pois se trata de uma linguagem com comandos intuitivos, como o “*find*” para encontrar a posição de uma palavra ou o “*print*” para imprimir um texto na tela. É também uma linguagem multiplataforma, no qual permite diversos tipos de aplicações, como *web*, *mobile*, *desktop* e indústria de jogos (KRIGER, 2022). A biblioteca *Flask* foi outro recurso do *Python* muito útil, sendo uma ferramenta robusta e ágil para criar aplicações *web* com *Python* (UDEMY, 2022). Foi possível verificar que é uma ferramenta bem simples de ser implementada.

Será apresentado a seguir trabalhos cujo o objetivo geral é similar à proposta deste trabalho. Tais projetos foram analisados a fim de verificar a maneira com que a ferramenta foi implementada e utilizada no mercado de ofertas. Ambos os trabalhos envolvem comparação de preços por meio do *Web Scraping*, um envolvendo preços de automóveis, e o outro, preços de *skins* do jogo *CSGO* (*Counter-Strike: Global Offensive*).

Com o objetivo de auxiliar o consumidor na hora de adquirir um automóvel, proporcionando maior custo benefício, Borges e Gamini (2018), implementaram um algoritmo *scraper*, utilizando a biblioteca *Selenium*, para automatizar a coleta de dados do *site* da FIPE e de procedimentos *web scraping* para a análise e a consulta desses dados. Os dados coletados foram armazenados em um banco de dados *Oracle* para posteriormente serem exibidos em tabelas e gráficos.

Já Santos e Carrijo (2021), exploraram o mercado de *skins* do jogo *CSGO*. *Skins* é um termo que vem do inglês que significa pele, e no contexto deste jogo seria um visual para as armas dos personagens, onde inúmeros jogadores gastam dinheiro adquirindo essas *skins* para equipá-las durante seus jogos. O objetivo deste trabalho é auxiliar o mercado de *skins* da comunidade *CSGO*, desenvolvendo uma aplicação capaz de comparar os preços. Para a coleta dos dados foi usado a biblioteca *Beautiful Soup* do *Python*, os resultados obtidos foram salvos no banco de dados e depois exibidos em uma página da *Web*, uma ao lado da outra, sendo possível visualizar o nome do produto, o preço, e o *link* de acesso ao *site* em que o produto está disponível para compra.

Ambos os trabalhos fizeram uso da mesma biblioteca de extração utilizada nesta aplicação, o *Beautiful Soup*, alcançando seus objetivos com um resultado satisfatório. E diferente deste trabalho, não precisaram utilizar a biblioteca *Selenium* para complementar o processo de extração, pois as fontes coletadas não exigiram identificação de usuário para a coleta, o que resultou em um desempenho um pouco melhor.

Conclusões

Este trabalho propôs um site para apoiar a decisão de consumidores na compra de produtos eletrônicos da área de TI. Com esse objetivo, o projeto conta com dados extraídos de sites *e-commerce*, como os preços e os nomes de cada produto e seu *link* de redirecionamento. Com os produtos extraídos, é possível, por meio do algoritmo feito, indicar as melhores ofertas, organizando os resultados obtidos em uma lista que será exibida na página *web* permitindo ao usuário a identificação dos melhores preços.

A biblioteca *Beautiful Soup* facilitou a extração e o tratamento dos dados pela sua simplicidade e facilidade de aprendizado, na qual basta apenas ter conhecimentos de HTML e da linguagem utilizada para a construção do algoritmo *scraper*. Outra biblioteca de grande importância para o projeto foi a *Selenium*, que permitiu a automatização do *browser*, realizando as requisições em cada *site*.

Dentre os objetivos propostos pelo trabalho, quase todos foram alcançados, tendo sido criado o protótipo de uma ferramenta comparadora de preços, na qual entregou os resultados da forma esperada. A aplicação desenvolvida traz uma viabilidade para o problema da variedade de preços no mercado de *e-commerce*, pois tendo um *site* mais específico, disponibilizando os preços melhores ofertados de maneira organizada facilita a comparação entre esses *e-commerces*.

Para trabalhos futuros, pode ser implementado um cadastro de usuário no site, fazendo com que a aplicação tenha mais funções a partir do momento em que efetuar o *login*, como histórico de compras, carrinho para adicionar cada produto, favoritos etc. Além disso, poderia estender a aplicação, tornando-a inteligente e adaptável, iria agregar em dois fatores, o primeiro seria na adaptação do *site* ao usuário, no qual com base em cada perfil de usuário, recomendaria um determinado tipo de produto, que possa ser de seu interesse, e o segundo fator seria em relação a uma busca mais inteligente e refinada, retornando mensagens esclarecedoras para quando o produto não for encontrado. Outra implementação que pode ser feita, é a melhoria no desempenho da busca, utilizando ferramentas mais completas e específicas para esta função, na qual não foi possível explorar na elaboração deste trabalho. E para solucionar os problemas de acesso aos *sites*, poderia ser feito uma parceria, permitindo assim coletar os dados e redirecionar ao *site*, para o consumidor efetuar a compra.

Referências

ALURA. Formação começando com *Flask*. Alura. 2022. Disponível em: <<https://www.alura.com.br/formacao-flask>>. Acesso em: 02 set. 2022.

ANTIBOT. O que é *antibot*?. **Antibot Cloud**. 2022. Disponível em: <<https://antibot.cloud/pt.html>>. Acesso em: 01 set. 2022.

ALURA. O que é *Python*. **Alura**. 2022. Disponível em: <https://www.alura.com.br/apostila-python-orientacao-a-objetos/o-que-e-python?gclid=CjwKCAiA7IGcBhA8EiwAFfUDsWzFijXhatehZcsNRxn9h3WBfnb0vd7kEowI1jDzgFOEKaa7UtL7DhoCSI0QAvD_BwE>. Acesso em: 01 out. 2022.

BROUCKE, S. V.; BAESENS, B. *Practical Web Scraping for Data Science: Best Practices and Examples with Python*. 1 ed. USA: Editora Apress, 2018.

BORGES, Thiago da Cunha; GAMINI, Zeus Olenchuk. **Extração de dados com *web scraping* para análise da variação de preço de veículos automotores**. Niterói. 2018. 53 p. Trabalho de Conclusão de Curso (Tecnólogo em Sistemas de Computação) - Universidade Federal Fluminense. Disponível em: <<https://app.uff.br/riuff/handle/1/8930>>. Acesso em: 30 out 2022.

DINO. Preços de notebooks disparam com alta do dólar. **Terra**. 2021. Disponível em: <<https://www.terra.com.br/noticias/precos-de-notebooks-disparam-com-alta-do-dolar,3dd5b62a3dd54e1b366bb04623017cae9d2e751o.html#:~:text=A%20alta%20do%20d%C3%B3lar%2C%20ao,de%20an%C3%A1lises%20de%20mercado%20GfK.>>. Acesso em: 10 out. 2022.

FERNANDES, Dinalva. Compra online é preferência de 74% dos consumidores brasileiros. **e-commercebrasil**. 2019. Disponível em: <<https://www.ecommercebrasil.com.br/noticias/compra-online-preferencia-de-consumidores-brasileiros>>. Acesso em: 24 set. 2022.

G1. Varejo eletrônico cresce mais de 67% no Brasil; saiba como comprar de forma segura na internet. **G1**. 2022. Disponível em: <<https://g1.globo.com/pa/santarem-regiao/noticia/2022/08/28/varejo-eletronico-cresce-mais-de-67percent-no-brasil-saiba-como-comprar-de-forma-segura-na-internet.ghtml>>. Acesso em: 15 set 2022.

HOSTINGER. Erro 403 *forbidden*: O que significa e como corrigir. **HOSTINGER**. 2022. Disponível em: <<https://www.hostinger.com.br/tutoriais/o-que-significa-erro-403>>. Acesso em: 01 set. 2022.

JANKAVSKI, André. Brasileiros querem manter home office, mas temem excesso de trabalho, diz estudo. **CNN Brasil**. 2021. Disponível em: <<https://www.cnnbrasil.com.br/business/maioria-aprova-o-home-office-mas-ha-preocupacao-com-excesso-de-trabalho/>>. Acesso em: 10 set 2022.

KRIGER, Daniel. O que é *python*, para que serve e por que aprender?. **KENZIE**. 2022. Disponível em: <<https://kenzie.com.br/blog/o-que-e-python/>>. Acesso em: 19 nov 2022.

OKUBO, Beatriz. O Que é HTML e para que serve?. **GoDaddy**. 2022. Disponível em: <<https://br.godaddy.com/blog/o-que-e-html-e-para-que-serve/>>. Acesso em: 18 nov. 2022.

OKUBO, Beatriz. Você sabe o que é CSS? Entenda como funciona e para que serve. **GoDaddy**. 2021. Disponível em: <<https://br.godaddy.com/blog/voce-sabe-o-que-e-css-entenda-como-funciona-e-para-que-serve/>>. Acesso em: 19 nov. 2022.

PAULO, João. Guia para automação *web* no *python* utilizando o *selenium*. **Hashtag**. 2022. Disponível em: <<https://www.hashtagtreinamentos.com/automacao-web-no-python#:~:text=O%20Selenium%2C%20nada%20mais%20%20C3%A9,basta%20escrever%20pip%20install%20selenium>>. Acesso em: 31 out. 2022.

PINHATA, Gabrielle. 10 Heurísticas de Nielsen: melhorando a usabilidade de suas interfaces. **Programmers**. 2022. Disponível em: <<https://www.programmers.com.br/blog/10-heuristicas-de-nielsen-melhorando-a-usabilidade/>>. Acesso em: 06 ago. 2022.

PLANALTO. Lei N° 13.709, de 14 de agosto. **Planalto**. 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/113709.htm>. Acesso em: 31 fev. 2022.

PORNARAS, Jorge. *Selenium vs. Beautiful Soup: A full comparison*. **BlazeMeter**. 2021. Disponível em: <<https://www.blazemeter.com/blog/selenium-vs-beautiful-soup-python>>. Acesso em: 02 set. 2022.

RICHARDSON, Leonardo. Documentação *beautiful soup*. **Crummy**. 2015. Disponível em: <<https://www.crummy.com/software/Beautiful Soup/bs4/doc.ptbr/>>. Acesso em: 20 out. 2022.

SANTOS, Eliseu Xavier; CARRIJO, Pedro Felipe. **Comparador de preços de skins do CSGO utilizando web scraping**. Anápolis. 2021. 37 p. Trabalho de Conclusão de Curso (Bacharel em Engenharia de Software) - Universidade Evangélica de Goiás. Disponível: <<http://repositorio.aee.edu.br/handle/aee/19626>>. Acesso em: 31 out 2022.

UDEMY. Flask masterclass - desenvolvendo aplicações web com *python*. **Udemy**. 2022. Disponível em: <<https://www.udemy.com/course/criando-aplicacoes-com-flask-docker/>>. Acesso em: 03 set. 2022.

ANEXO A

DIRETRIZES PARA A SUBMISSÃO DE ARTIGOS DA REVISTA CIENTÍFICA FUNVIC

Os trabalhos devem ser redigidos em português, com uso obrigatório da norma culta. Durante o preenchimento cadastral, o nome completo de cada autor e respectiva afiliação institucional devem ser inseridos nos campos adequados e devem aparecer no arquivo. A Revista Eletrônica de Ciências Exatas sugere que o número máximo de autores por artigo seja 6 (seis). Artigos com número superior a 6 (seis) serão considerados exceções e avaliados pelo Conselho Editorial que poderá solicitar a adequação. **Pesquisas feitas com seres humanos e animais devem, obrigatoriamente, citar a aprovação da pesquisa pelo respectivo Comitê de Ética, citando o protocolo de aprovação.** O não atendimento implica em recusa da submissão. Da mesma forma, o plágio implicará na recusa do trabalho.

O uso da norma culta da Língua Portuguesa, a obediência às normas da Revista e a originalidade do artigo são de total responsabilidade dos autores. O não atendimento a esses critérios implicará na recusa imediata do trabalho.

APRESENTAÇÃO DO MATERIAL

O número máximo é de 20 páginas, incluindo referências, figuras, tabelas e quadros. Os textos devem ser redigidos em **Fonte Times New Roman, tamanho 12, espaçamento 1,5 cm, justificado.** Devem ser utilizadas margens de 2 cm em cada lado.

As Figuras (gráficos, imagens, desenhos, fluxogramas e esquemas) deverão apresentar boa nitidez, estar em formato JPEG, com resolução de 800dpi e com tamanho de 15cm x 10cm. As figuras deverão ser enumeradas consecutivamente em algarismos arábicos, encabeçadas pelas respectivas legendas; as fontes, logo abaixo das mesmas, com fonte de tamanho 11 e espaçamento simples.

Tabelas e Quadros deverão ser enumeradas consecutivamente com algarismos arábicos e encabeçados pelos respectivos títulos, com fonte de tamanho 11 e espaçamento simples.

As citações deverão aparecer ao longo do texto, conforme Normas da ABNT (NBR 10520, 2002), seguidas pelo ano de publicação, cujas chamadas podem ser pelo sobrenome do autor, pela instituição responsável ou pelo título. As citações podem ser incluídas na sentença: sobrenome (ano). Ex.: Gomes, Faria e Esper (2006) ou entre parênteses: (SOBRENOME, ano). Ex.: (GOMES; FARIA; ESPER, 2006). Quando se tratar de citação direta (transcrição literal), indicar, após o ano, a página de onde o texto foi extraído. O trecho transcrito deverá estar entre aspas quando ocupar até três linhas. As citações diretas com mais de três linhas devem ser destacadas com recuo de 4 cm da margem esquerda, ser escritas com tamanho 11, com espaçamento entre linhas simples e sem aspas. Citações indiretas de vários documentos simultâneos devem constar em ordem alfabética (como nas referências). Citação de citação: deve-se fazer a referência do autor lido. Ex.: Pádua (1996 apud FERNANDES, 2012, p. 5) salienta que “[...] pesquisa é toda atividade voltada para a solução de problemas [...]”.

Teses e dissertações, quando não houver o respectivo artigo científico publicado em periódico, devem ser dos **últimos três anos**; obrigatoriamente indicando o **link** que remeta ao cadastro nacional de teses da CAPES ou das universidades onde esses documentos foram publicados. Não serão aceitas Monografias de Especialização como referência.

Grafia de termos científicos, comerciais, unidades de medida e palavras estrangeiras devem ser grafados por extenso, em vez de seus correspondentes simbólicos abreviados, quando citados pela primeira vez. Deve-se utilizar o Sistema Internacional de Unidades.

Palavras estrangeiras devem ser evitadas, utilizar preferentemente a sua tradução. Na impossibilidade, os termos estrangeiros devem ser grafados em itálico.

ESTRUTURA DO ARTIGO

PESQUISAS ORIGINAIS devem ter no máximo 20 páginas com até 40 citações; organizar da seguinte forma:

Título em português: caixa alta, centralizado, negrito, conciso, com um máximo de 25 palavras.

Título em inglês (obrigatório): caixa alta, centralizado. Versão do título em português.

Autor(es): o(s) nome(s) completo(s) do(s) autor(es) e seus títulos e afiliações à Sociedade ou Instituições. Indicar com asterisco o autor de correspondência. Ao final das afiliações fornecer o e-mail do autor de correspondência.

Resumo: parágrafo único sem deslocamento, fonte tamanho 11, espaço simples, justificado, contendo entre 150 e 250 palavras. Deve conter a apresentação concisa de cada parte do trabalho, abordando objetivos, métodos, resultados, discussão e conclusões. Deve ser escrito sequencialmente, sem subdivisões. Não deve conter símbolos, equações, diagramas, fórmulas e contrações que não sejam de uso corrente.

Palavras-chave: de 3 a 5 palavras-chave, iniciadas por letra maiúscula, separadas e finalizadas por ponto.

Abstract: tradução literal do resumo, com formatação idêntica à do resumo.

Keywords: tradução literal das Palavras-chave em Português.

Introdução: deve apresentar o assunto a ser tratado, fornecer ao leitor os antecedentes que justificam o trabalho, incluir informações sobre a natureza e importância do problema, sua relação com outros estudos correlatos e suas limitações. Essa seção deve representar a essência do pensamento do pesquisador em relação ao assunto estudado e apresentar o que existe de mais significativa na literatura científica. Os objetivos da pesquisa devem figurar como o último parágrafo desse item.

Método: destina-se a expor os meios dos quais o autor se valeu para a execução do trabalho. Pode ser redigido em corpo único ou dividido em subseções. Especificar tipo e origem de produtos e equipamentos utilizados. Citar as fontes que serviram como referência para o método escolhido.

Pesquisas feitas com seres humanos e animais devem, obrigatoriamente, citar a aprovação da pesquisa pelo respectivo Comitê de Ética, citando o protocolo de aprovação.

Resultados: Nesta seção o autor irá expor o obtido em suas observações. Os resultados poderão ser apresentados em quadros, tabelas ou figuras, não podendo ser repetidos em mais de um tipo de ilustração.

Discussão: O autor, ao tempo que justifica os meios que usou para a obtenção dos resultados, deve confrontá-los com a literatura pertinente; estabelecer relações entre causas e efeitos; apontar as

generalizações e os princípios básicos que tenham comprovações nas observações experimentais; esclarecer as exceções, modificações e contradições das hipóteses, teorias e princípios diretamente relacionados com o trabalho realizado; indicar as aplicações teóricas ou práticas dos resultados obtidos, bem como, suas limitações; indicar, quando necessário, uma teoria para explicar certas observações ou resultados obtidos; sugerir, quando for o caso, novas pesquisas a partir da experiência adquirida no desenvolvimento do trabalho e visando a sua complementação.

Conclusões: Deve expressar de forma lógica e objetiva o que foi demonstrado com a pesquisa.

Agradecimentos (opcionais): O autor pode agradecer às fontes de fomentos e àqueles que contribuíram efetivamente para a realização do trabalho. Agradecimento a suporte técnico deve ser feito em parágrafo separado.

Referências (e não bibliografia): Espaço simples entre linhas e duplo entre referencias. A lista completa de referências, no final do artigo, deve ser apresentada em ordem alfabética e de acordo com as normas da ABNT (NBR 6023, 2002). Quando a obra tiver até três autores, todos devem ser citados. Mais de três autores, indicar o primeiro, seguido de et al. Alguns exemplos:

Artigo publicado em periódico:

LUDKE, M.; CRUZ, G. B. dos. Aproximando universidade e escola de educação básica pela pesquisa. **Caderno de pesquisa**, São Paulo, v. 35, n. 125, p. 81-109, maio/ago. 2005.

Artigo publicado em periódico em formato eletrônico:

SILVA JUNIOR, N. A. da. Satisfação no trabalho: um estudo entre os funcionários dos hotéis de João Pessoa. **Psico-USF**, Itatiba, v. 6, n. 1, p. 47-57, jun. 2001. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-82712001000100007&lng=pt&nrm=iso>. Acesso em: 13 jul. 2015.

Livro (como um todo)

MENDONÇA, L. G. et al. **Matemática financeira**. 10. ed. Rio de Janeiro: Editora FGV, 2010.

Capítulo de livro

MARTÍN, E.; SOLÉ, I. A aprendizagem significativa e a teoria da assimilação. In: COLL, C.; MARCHESI, A.; PALACIOS, J. (Org.). **Desenvolvimento psicológico e educação: psicologia da educação escolar**. 2. ed. Porto Alegre: Artmed, 2008. cap. 3, p. 60-80.

ARTIGOS DE REVISÃO

Poderão ser aceitos para submissão, desde que abordem temas atuais e de interesse. Devem ter até 20 páginas, incluindo resumos, tabelas, quadros, figuras e referências. As figuras não devem repetir dados já descritos em tabelas.

Devem conter: título em português e inglês, autores e afiliações, resumo e abstract (de 150 a 250 palavras), palavras-chave/keywords, introdução, método (como nos artigos de pesquisas originais) considerações finais (neste item serão retomadas as diferentes discussões dos autores estudados de maneira a conduzir a um fechamento, porém, não havendo conclusões definitivas), agradecimentos (opcional) e referências.

Artigos de revisão de literatura contendo metanálise, depois do item método deverá ser apresentado o item resultados (contendo a metanálise) e as conclusões.

Autorizamos a cópia total ou parcial desta obra, apenas para fins de estudo e pesquisa, sendo expressamente vedado qualquer tipo de reprodução para fins comerciais sem prévia autorização específica do autor. Autorizo também a divulgação do arquivo no formato PDF no banco de monografias da Biblioteca institucional.

Mateus Camargo Garcia de Siqueira

Mauricio Camargo Garcia de Siqueira

Rafael Bruno dos Santos Costa

Pindamonhangaba, 30 de novembro 2022