UNIVERSIDADE FEDERAL DO ABC

Plataforma para comparação de especificações técnicas de produtos com uso Web Scraping na aquisição de informações

Aluno: Franklin Adson Roque
Orientador: Mario Alexandre Gazziro

São José dos Campos
2024

Resumo

O aumento exponencial do comércio eletrônico nas últimas décadas motivado por diversos fatores, como a globalização, acessibilidade à internet e a crise sanitário do COVID-19, propiciou a fabricantes e comerciantes oferecer seus produtos e serviços a usuários geograficamente distantes, aumentando assim a oferta de produtos semelhantes e compatíveis tecnicamente, o que acaba tornando o trabalho de comparação de especificações e compatibilidades entre produtos uma atividade complexa. Este projeto visa a implementação de um aplicativo de comparação de produtos eletrônicos. A solução faz uso de técnicas de web scraping para a aquisição de informações e mineração de dados para a comparação dos mesmos, possui o frontend desenvolvido com React Native, o backend em python, com aplicação das bibliotecas BeautifulSoup e Scrapy, além de infra hospedada em ambiente AWS. A ferramenta mostrou-se tecnicamente viável e com potencial para contribuir com comunidade acadêmica, denotando uma possível adoção prática da técnica do web scraping e também pela adoção de repositório atualizado local, em vista de garantir melhor desempenho.

Abstract

The exponential growth of e-commerce in recent decades, driven by several factors, such as globalization, internet accessibility, and the COVID-19 health crisis, has enabled manufacturers and retailers to offer their products and services to geographically distant users, thus increasing the supply of similar and technically compatible products, which ends up making the work of comparing specifications and compatibilities between products a complex activity. This project aims to implement an electronic product comparison application. The solution uses web scraping techniques to acquire information and data mining for comparison, has a frontend developed with React Native, a backend in Python, with application of the BeautifulSoup and Scrapy libraries, in addition to infrastructure hosted in an AWS environment. The tool proved to be technically viable and with the potential to contribute to the academic community, denoting a possible practical adoption of the web scraping technique and also the adoption of a local updated repository, in order to ensure better performance.

SUMÁRIO

INTRODUÇÃO

O comércio eletrônico, tal como o conhecemos hoje, teve suas origens no início da década de 1990, impulsionado pela crescente popularização da internet no Brasil e impulsionado pela globalização [1, 2, 3, 4 e 5]. Esta tendência que se estendeu pelas últimas décadas, tem gerado uma profunda transformação na dinâmica do consumo e ampliando significativamente o acesso dos consumidores a uma vasta gama de produtos. Este fenômeno foi potencializado ainda mais após a pandemia da COVID-19, onde diversos produtos tiveram sua aquisição quase que majoritariamente por meio do comércio eletrônico [6].

Neste contexto, comerciantes e fabricantes encontraram oportunidades para expandir suas ofertas para além das fronteiras geográficas tradicionais, alcançando um público global. Contribuindo para esse alcance, a evolução da logística moderna desempenhou um papel crucial, simplificando a entrega direta de mercadorias e possibilitando o acesso a mercados anteriormente inacessíveis [7, 8].

Contudo, a ampla variedade de opções disponíveis nesse ambiente digitalizado e globalizado impôs desafios significativos aos consumidores. Estes enfrentam a difícil tarefa de escolher entre produtos com especificações técnicas e preços variados, uma situação complicada pela disparidade entre esses elementos e a presença de padrões incompatíveis [9]. Tal desafio não se limita apenas aos consumidores finais, mas se estende a profissionais técnicos, engenheiros e especialistas em implantação, que encontram dificuldades similares ao avaliar e selecionar produtos específicos [10].

Diante desse cenário desafiador, o presente projeto de pesquisa propõe o desenvolvimento de uma plataforma especializada voltada para a comparação criteriosa das especificações técnicas de produtos similares ou equivalentes. O objetivo principal desta iniciativa é fornecer subsídios para auxiliar na tomada de decisão durante o processo de aquisição de mercadorias, visando simplificar e tornar mais eficiente a escolha do consumidor, bem como otimizar as decisões técnicas para os profissionais envolvidos[11 e 12].

OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver uma plataforma que permita a comparação de especificações técnicas de componentes de hardware, facilitando a tomada de decisão de consumidores e técnicos de manutenção. Esta plataforma busca integrar análises detalhadas e oferecer avaliações transparentes da relação custo-benefício dos produtos, uma necessidade identificada no contexto atual de comércio eletrônico, onde a variedade de opções pode confundir e sobrecarregar os usuários [6].

OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho consistem em:

- Aquisição de especificações técnicas de equipamentos de hardware advindos de sites de comércio eletrônico, sites de fabricantes e sites especializados, por meio da técnica de web scraping (raspagem de dados);

- Técnicas de mineração de dados sobre os dados, no intuito de prover informações suficientes para a composição de comparativos;

- Aplicação WEB (aplicativo) para consultas e consumo das informações advindas da plataforma;

JUSTIFICATIVA

As vendas on-line têm se expandido notavelmente, especialmente após eventos globais como a pandemia da COVID-19, que destacaram a importância de sistemas de comércio eletrônico eficientes [6]. Neste contexto, consumidores e usuários se veem diante de uma ampla variedade de produtos semelhantes, mas frequentemente enfrentam a falta de informações necessárias para fazer escolhas

baseadas em critérios técnicos fundamentados, como a relação custo-benefício. Esta lacuna entre a oferta abundante e a habilidade de tomar decisões informadas reflete um desafio amplamente discutido na literatura sobre o comportamento do consumidor [13].

O desenvolvimento de uma plataforma dedicada à comparação de componentes de hardware surge como uma solução direta para essa necessidade. O foco deste trabalho é criar uma ferramenta que não apenas simplifique o processo de compra online ou em lojas físicas para consumidores, mas também se torne um recurso indispensável para técnicos de manutenção. Esta plataforma propõe integrar informações detalhadas sobre especificações técnicas e fornecer uma avaliação transparente da relação custo-benefício, apoiando-se em teorias comprovadas de tomada de decisão [10].

A iniciativa de compilar e simplificar o acesso a dados técnicos de produtos equivalentes se alinha às tendências contemporâneas de logística e globalização do comércio, que permitem aos fabricantes alcançar mercados anteriormente inexplorados [8 e 14]. Assim, a plataforma visa não apenas facilitar a compra por parte dos consumidores, mas também melhorar a eficácia das decisões técnicas no âmbito do comércio eletrônico moderno, fortalecendo o papel do consumidor como um participante informado e ativo no mercado.

REVISÃO DE LITERATURA

Para tornar este projeto possível, fez-se necessária uma pesquisa aprofundada sobre as técnicas de *Mineração de dados* e *Web Scrapping*. A técnica do Web Scriapping será utilizada na aquisição das informações em sites dos fabricantes de componentes de computador. Já a Mineração de dados será empregada nos resultados, na busca e composição dos comparativos. Estas técnicas serão implementadas em linguagem Python, o que também exigiu aprofundamento na documentação da linguagem. Também foi necessário pesquisa acerca das questões éticas e de direito do uso destas técnicas, pois alguns sites não permitem a reprodução das informações por conta de direitos autorais.

Mineração de Dados: Conceitos e Aplicações

O termo Mineração de dados, segundo a literatura, possui diversas definições [15], mas todas convergem para "*Mineração de dados, de forma simples, é o processo de extração ou mineração de conhecimento em grandes quantidades de dados"* [16]. Esta técnica transforma dados brutos advindos de uma grande massa de dados, em informações e insights (conhecimentos práticos).

As aplicações práticas da mineração de dados podem ser observadas nas mais diversas aplicações. No contexto acadêmico, é possível notar a aplicação destas estratégias na computação (em conjunto com aprendizagem de máquina e Inteligência Artificial, ou de forma isolada), na estatística, nas engenharias e em diversas outras áreas de conhecimento. É possível também observar adoção destas técnicas em soluções comerciais, como em instituições financeiras, Business Intelligence (BI), análise de tendências de mercados, dentre outros.

As técnicas mais comuns de Mineração de dados são:

1) Classificação: Consiste em atribuir rótulos a itens de dados com base em suas características. Algoritmos como Árvores de Decisão, Naive Bayes e Máquinas de Vetores de Suporte (SVM) são comumente usados para classificação.

2) Agrupamento (Clusterização): Agrupa itens de dados similares com base em suas características. Algoritmos populares incluem o K-Means, DBSCAN e o algoritmo de Agrupamento Hierárquico.

3) Regressão: Prevê um valor contínuo com base em variáveis independentes. Regressão Linear, Regressão Logística e Redes Neurais são exemplos de técnicas de regressão.

4) Associação: Identifica padrões de associação entre diferentes variáveis em grandes conjuntos de dados. O algoritmo Apriori é comumente usado para encontrar associações frequentes em conjuntos de transações.

5) Análise de Séries Temporais: Lida com dados que mudam com o tempo para prever tendências futuras ou padrões. Métodos como Suavização Exponencial, Modelos ARIMA e Redes Neurais Recorrentes são usados para análise de séries temporais.

6) Redução de Dimensionalidade: Reduz a quantidade de variáveis em um conjunto de dados, preservando o máximo possível de informações. Técnicas como Análise de Componentes Principais (PCA) e Seleção de Características são usadas para esse fim.

7) Mineração de Texto: Extrai informações úteis e padrões de dados de texto não estruturado. Técnicas incluem Análise de Sentimento, Extração de Tópicos e Classificação de Texto.

8) Mineração de Grafos: Analisa as relações entre entidades em um conjunto de dados representado como um grafo. Algoritmos como PageRank e Centralidade de Grau são usados para identificar nós importantes em redes complexas.

Web Scraping: Fundamentos e Técnicas

O web scraping é uma técnica de mineração de dados que tem como objetivo a busca por expressões, padrões ou informações, que encontram-se pulverizadas em homepages ou sites da WEB. A ferramenta então vasculha em páginas da internet, as informações desejadas, de acordo com parâmetros e estratégias predefinidas, por vezes reproduzindo as atividades normais de um usuário real (clicando em links, preenchendo formulários, etc).

No âmbito acadêmico esta técnica é particularmente importante na aquisição de informações em grandes bases de dados não sistematizados (como data warehouses e data lakes), provendo subsídios para outras ferramentas, como o Machine Learning, análise de tendências e padrões, análise comportamental e Inteligência Artificial. A técnica do web scraping pode ser empregada também para automatização de processos, como ferramenta de atualização contínua, coleta de dados não disponíveis por API's e diversas outras possibilidades.

Comercialmente esta ferramenta pode ser usada estrategicamente, por exemplo: no monitoramento e análise de preços de produtos, tendências do consumidor, plataformas de comparação de preços, provisão de dados estratégicos e geração de leeds, monitoramento de reputação da empresa e feedback dos clientes, ferramentas de Recursos Humanos (que buscam candidatos para vagas

estratégicas), coletas de dados e tendências do mercado imobiliário e diversas outras.

Questões Éticas na Mineração de Dados e Web Scraping

A técnica de web scraping vem sendo empregada com sucesso em diversos contextos e cenários, provendo informações de forma importantes e de forma eficiente mesmo quando estas não se encontram sistematizadas.

Porém a técnica vem rechaçada de críticas acerca da adoção da mesma, por questões acerca da propriedade intelectual das informações, da privacidade dos dados pessoais de usuários e/ou dados sensíveis de pessoas ou instituições.

É necessário frisar que o uso da técnica de forma ética e legal, do ponto de vista da legislação em vigor (Lei dos direitos autorais e a Lei Geral de Proteção aos Dados - no âmbito nacional - e suas respectivas equivalentes ao redor do mundo) exigem uma série de ações antes mesmo da aplicação da técnica. No caso de coleta de dados pessoais, faz-se necessário verificar como estes dados serão usados, se as fontes que os fornecem tem permissão para isso, se essas fontes permitem o uso destes dados por terceiros e também se estes dados podem ser replicados e/ou expostos (caso seja necessário). Essa verificação deve ser realizada junto às fontes de dados, onde constam os termos de uso. Alguns websites provêm estas informações (permissão ou não do uso das informações do site) por meio de arquivos robots.txt, que são carregados juntamente com o conteúdo HTML da página. Nestes arquivos podem ser definidas as áreas públicas e não públicas do site.

METODOLOGIA

Para atingir os objetivos estabelecidos e garantir a solidez do escopo, foram adotadas ferramentas da engenharia de software, como o Diagrama de Contexto e o Diagrama de Fluxo de Dados, visando expor toda a sequência de trabalho do sistema. Inicialmente, foram identificadas quatro entidades principais.

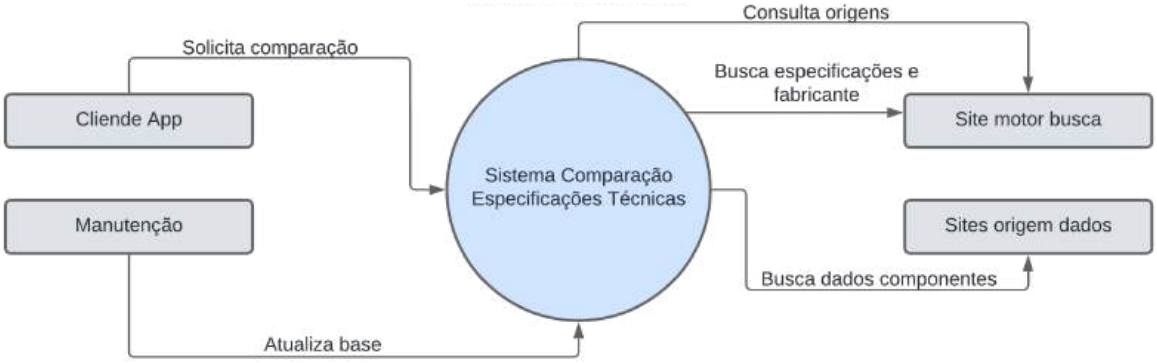

Figura 1: Diagrama de contexto

A entidade “Cliente App” representa o agente externo que solicita a comparação entre os componentes, ou seja, o usuário final do aplicativo. A entidade “Manutenção” é um recurso automatizado destinado a verificar e atualizar as URLs (Uniform Resource Locator - Localizador Uniforme de Recursos) dos sites de fabricantes e provedores de informações técnicas, garantindo a atualização constante do acervo de URLs. Já a entidade “Site motor busca” é responsável tanto pela aquisição das URLs dos fabricantes quanto pela busca e normalização dos atributos utilizados na pesquisa nos sites dos fabricantes. Por fim, a entidade “Sites origem dados” refere-se aos provedores de informações das especificações técnicas dos componentes a serem comparados.

O fluxo dos dados, bem como os processos principais podem ser verificados no Diagrama de Fluxo de Dados a seguir.

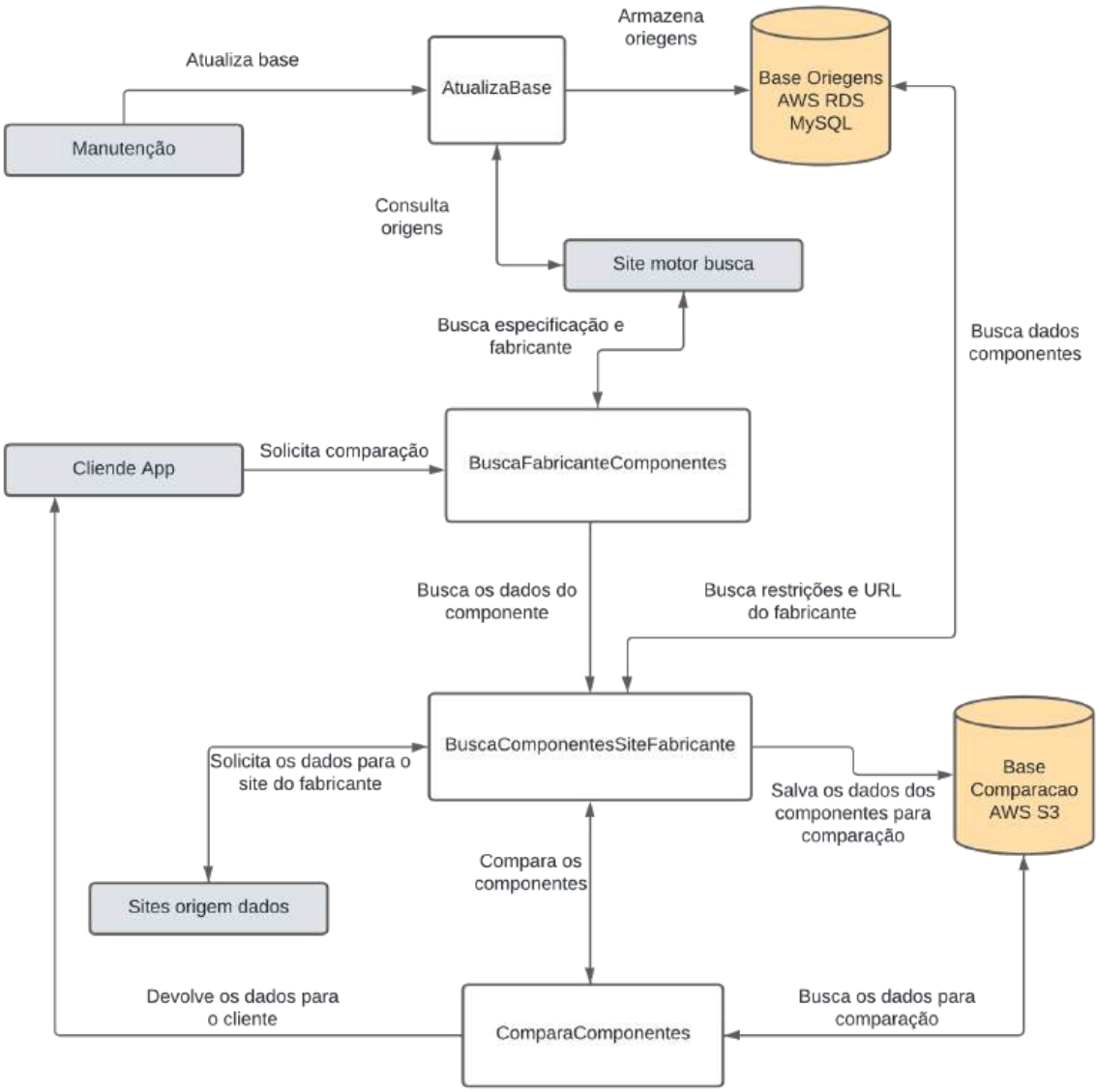

Figura 2: Diagrama de Fluxo de dados Nível 0

As entidades constantes no Diagrama de contexto (figura 1) interagem entre si, conforme exemplificado na figura 2. Segue abaixo a descrição das funcionalidades previstas para cada um dos processos:

- “AtualizaBase”: processo que é acionado pelo agente automatizador “Manutenção” e tem como objetivo atualizar e renovar o acervo de URLs dos fabricantes armazenadas no sistema. Este processo verifica a atividade das páginas (se estão operando) e mudanças nos termos de uso. Caso a página esteja operando e permita a técnica de web scraping, a URL é armazenada no acervo como origem para busca de componentes da marca especificada. Se o retorno for negativo, o processo busca alternativas com as mesmas regras. Os resultados, positivos ou negativos, são armazenados em AWS RDS MySQL, um banco de dados relacional hospedado na Amazon Web Services.

- “BuscaFabricanteComponentes”: recebe do “Cliente App” a informação de dois componentes a serem comparados. A primeira atividade deste processo é utilizar a entidade “Site motor busca” para verificar se a entrada fornecida pelo usuário corresponde a um componente de computador, identificando o tipo de componente e seu fabricante. Este processo é executado de forma assíncrona para as duas entradas fornecidas pelo usuário. Se os dois componentes forem compatíveis, o processo aciona o “BuscaComponentesSiteFabricante”.

- “BuscaComponentesSiteFabricante”: é responsável por coletar informações dos “Sites origem dados”, previamente armazenadas e atualizadas na base de dados AWS RDS. Este processo vasculha os sites origem em busca das informações do componente desejado e armazena toda a especificação “sem tratamento” no AWS S3, um repositório multiuso da Amazon Web Services. Posteriormente, o processo aciona o “ComparaComponentes”.

- “ComparaComponentes”: vasculha todas as especificações armazenadas no AWS S3 e prepara as informações para o “Cliente App” em formato de tabela comparativa. Se a incompatibilidade entre os elementos comparados ultrapassar 40%, este processo retorna uma mensagem de incompatibilidade ao “Cliente App”.

Como o processo mais crítico da plataforma é a aquisição das informações por meio da técnica de Web Scraping, optou-se pela linguagem Python por ser a mais empregada para Mineração de Dados, incluindo Web Scraping. Python é uma linguagem de alto nível, interpretada, multiparadigma (suporta orientação a objetos, imperativo, funcional e procedural), com uma ampla comunidade e possui diversos Frameworks e bibliotecas.

O processo de Web Scraping será implementado fazendo uso das bibliotecas BeutifulSoup e scrapy, cada uma em funções distintas. Uma terceira biblioteca a ser empregada para apoiar a funcionalidade de buscas em acervos é o Selenium.

BeautifulSoup é uma biblioteca amplamente usada para Web Scraping por tem como características o uso facilitado e intuitivo, uma curva de aprendizagem simples, diversos tutoriais online, uma extensa comunidade e a facilidade em fazer *parse* de documentos HTML. A mesma será responsável pela extração dos dados armazenados em forma de páginas HTML no AWS S3.

O scrapy é um framework completo para Web Scraping, que inclui funcionalidades avançadas como rastreamento automático de sites, lida nativamente com requisições HTTP, possui a capacidade de seguir links por várias páginas de forma automatizada e permite definir regras de navegação que evitam loops. Possui uma maior complexidade e a curva de aprendizagem é mais íngreme, no entanto tem alto desempenho, trabalha com pipelines e pode armazenar os dados diretamente no AWS S3 através de middlewares ou extensões. O scrapy será empregado no processo de aquisição das páginas HTML com as especificações dos componentes.

A biblioteca Selenium é uma biblioteca amplamente utilizada para automação e testes de interface, pois é capaz de seguir links em páginas construídas dinamicamente por javascript, como as SPA's (Single Page Applications - Aplicações de Páginas Simples). Por se tratar de um simulador de uso, esta biblioteca exige uso dos recursos de hardware e possui um desempenho inferior ao scrapy. Por este

motivo, ela será usada neste projeto nos casos onde o scrapy não consiga atingir o objetivo.

A exibição para o usuário final (frontend) será desenvolvida adotando a biblioteca React Native, uma biblioteca javascript para a construção de componentes que possui como principal característica a retrocompatibilidade entre as principais plataformas mobile (Android e IPhone), além de permitir também reuso de componentes para o desenvolvimento para desktop, com a mesma experiência de uso do aplicativo no smartphone (responsividade).

RESULTADOS

A ferramenta foi arquitetada usando o paradigma cliente-servidor (consagrada em aplicações web) com responsabilidades bem definidas em suas extremidades. Esta divisão de responsabilidades não só permite uma melhor organização do projeto como também possibilita a entrega de artefatos por módulos (versões intermediárias). Segue abaixo a descrição dos resultados obtidos acerca da adoção das tecnologias elencadas e os desafios encontrados durante sua implementação, divididos em frontend (visão do cliente, no paradigma cliente-servidor) e backend (responsabilidade do servidor).

Frontend

O desenvolvimento do frontend da aplicação iniciou-se com a ideação do design da ferramenta através da plataforma Figma (ferramenta WEB que permite diagramar diversos tipos de templates, tanto de aplicações desktop como para os mais variados dispositivos mobile). Foi pensada uma tela inicial, onde o usuário selecionará os componentes a serem comparados e um botão único para o processamento das informações. Na tela seguinte os resultados serão exibidos em formato de tabela, organizados de forma alfabética, com cada coluna correspondendo a um componente selecionado. A figura abaixo mostra o resultado do esboço inicial do aplicativo através da plataforma Figma.

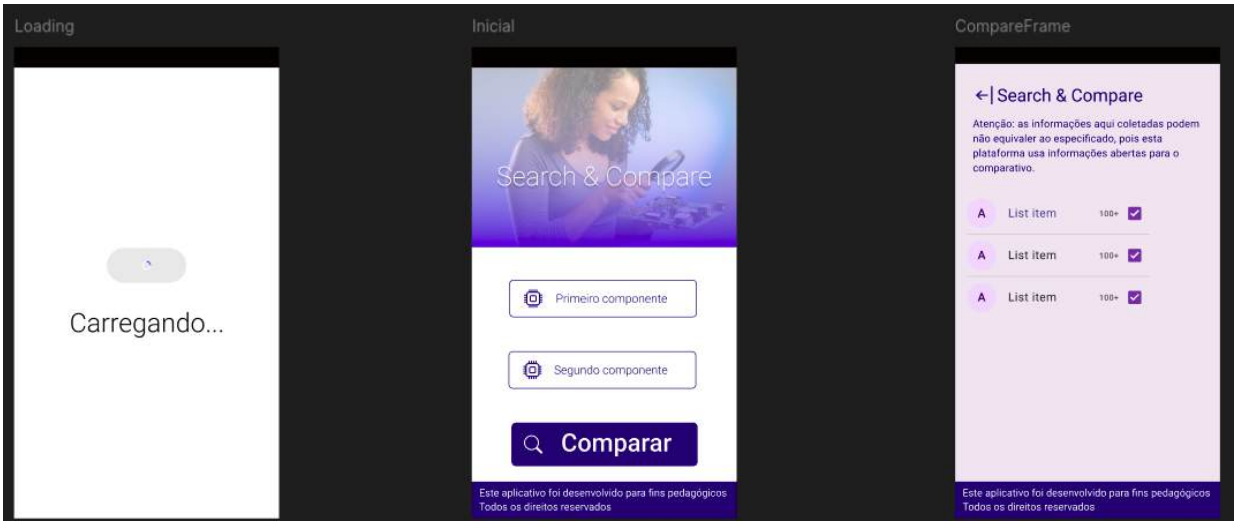

Figura 3: Esboços do template no Figma

O próximo passo no desenvolvimento do frontend é a conversão do esboço inicial do Figma para o modelo funcional, fazendo uso do framework/biblioteca escolhido para o desenvolvimento do frontend, neste caso o React-Native.

A adoção da biblioteca React-Native (versão do React para desenvolvimento mobile) através do framework Expo permitiu a construção e organização dos componentes de forma rápida e simplificada. Graças a sua curva de aprendizagem tênue e uma ampla gama de bibliotecas e componentes disponíveis para integração, foi possível implementar os componentes discriminados no esboço de forma simples e relativamente rápida. Segue abaixo a figura da tela de carregamento do aplicativo (comumente chamada de tela splash) e a tela principal.

Figura 4: tela de carregamento

Figura 5: tela principal

Inicialmente foi pensado em apenas dois botões de seleção dos componentes a serem comparados, e a equivalência da categoria dos mesmos (se os dois componentes pertencem a uma mesma categoria, ex: os dois são processadores ou os dois são memórias RAM) seria conferida pela aplicação após a seleção dos dois componentes, mas essa opção incidiria em mais complexidade na implementação. Optou-se então pela seleção inicial da categoria de componentes, reduzindo assim a complexidade da implementação.

Ao clicar no botão de seleção de categorias, uma caixa de diálogo é aberta com as categorias cadastradas.

Figura 6: modal de seleção das categorias

Figura 7: opções de categorias

Backend

Para permitir maior celeridade na implementação das funcionalidades do backend, optou-se pelo framework FastAPI do Python, por sua curva tênue de aprendizagem e aderência às tecnologias empregadas, como o BeautifulSoup, Selenium e Scrapy. O FastAPI permite a implementação de funções/métodos responderem chamadas HTTP, bastando apenas anotações (annotations) antes do mesmo. Os arquivos do projeto no backend foram organizados de acordo com seus contextos e funcionalidades, como demonstrado na imagem abaixo.

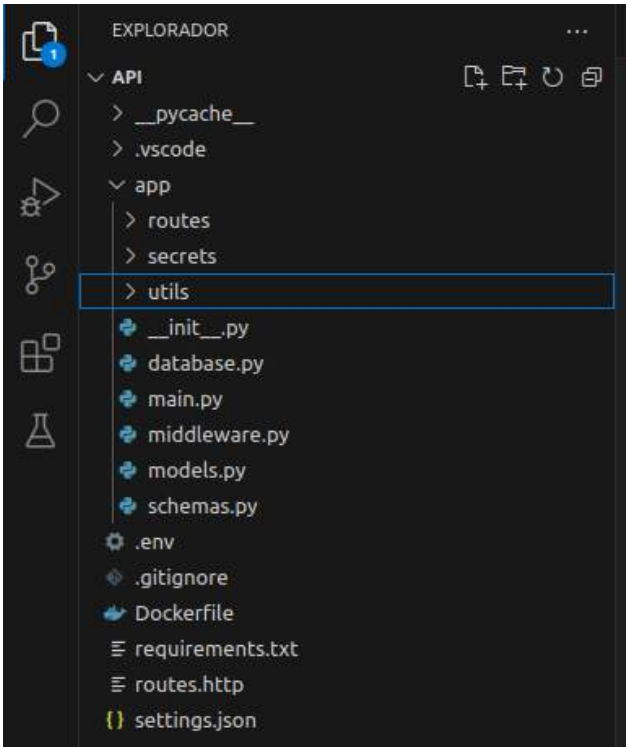

Figura 8: descrição da organização do projeto

O banco de dados da aplicação se resume a três tabelas envolvidas com a regra de negócio (produtos, categorias e urls) e uma tabela de logs, como descrito na figura abaixo.

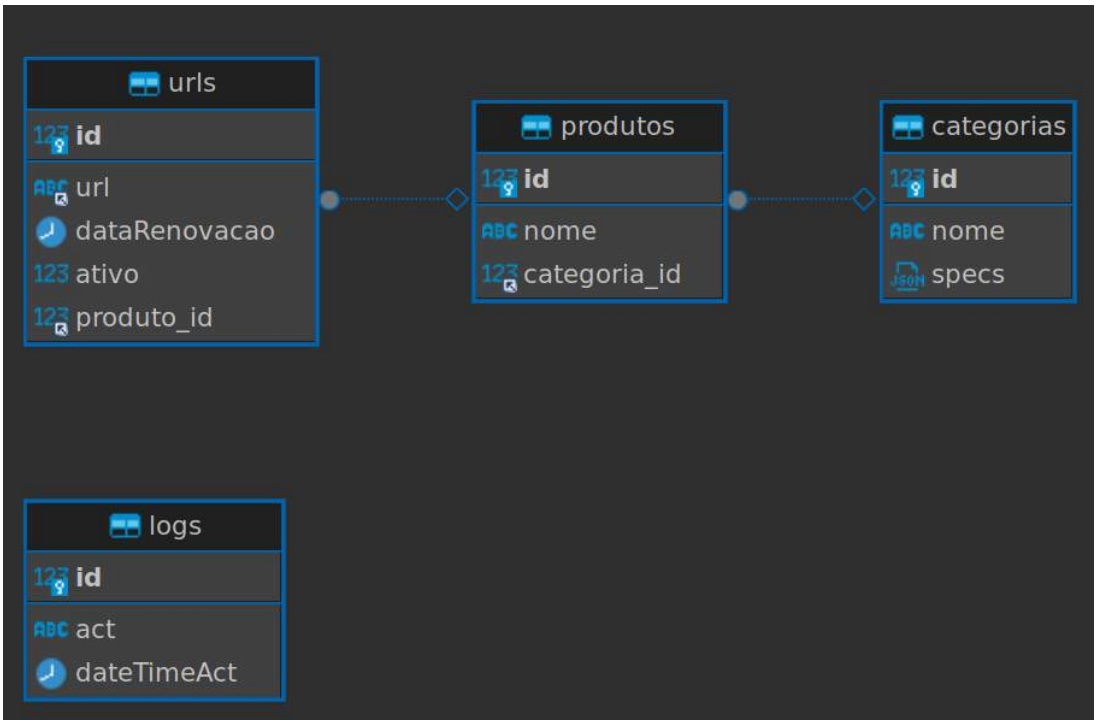

Figura 9: Modelagem dos dados

Os relacionamentos entre as tabelas da solução são:

Tabela origem	Relacionamento	Tabela destino	
categorias	1 <> N	produtos	
produtos	1 <> N	urls	* O armazenamento de mais de uma url de especificações para um mesmo produto se justifica para garantir a disponibilidade dos dados.

Tabela 1: Relacionamentos da regra de negócio

Para integração com o banco de dados MySQL, adotou-se o ORM SQLAlchemy, o qual também possui uma curva de aprendizagem extremamente tênue e aderência ao FastAPI.

No script principal (main.py) implementou-se middleware que automatiza o registro de todos os scripts que poderão receber requisições HTTP Rest (que estão alojados na pasta routes) e uma rotina de logs da aplicação, capturando todos os erros de execução e armazenando em um banco de dados.

app > main.py > ...

```python
import logging
from fastapi import FastAPI, Request, Depends, HTTPException
from app.middleware import LoggingMiddleware
import importlib
import os
import app.models as models
from app.database import SessionLocal, engine

models.Base.metadata.create_all(bind=engine)

app = FastAPI()

# Dependência
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# Função para registrar todos os routers
def register_routers(app):
    routes_dir = os.path.join(os.path.dirname(__file__), 'routes')
    for filename in os.listdir(routes_dir):
        if filename.endswith('.py') and filename != '__init__.py':
            module_name = f"app.routes.{filename[:-3]}"
            try:
                module = importlib.import_module(module_name)
                if hasattr(module, 'router'):
                    app.include_router(module.router)
            except Exception as e:
                print(f"Erro ao importar o módulo {module_name}: {e}")
register_routers(app)

app.add_middleware(LoggingMiddleware)

if __name__ == '__main__':
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Figura 10: código do programa principal, que recebe as requisições HTTP Rest e trata exceções de processamento

A pasta utils possui scripts python que não são acessados diretamente por métodos HTTP Rest, mas que oferecem à aplicação funcionalidades das tecnologias

empregadas, como conexão com a AWS, open_ai, GCP entre outros.

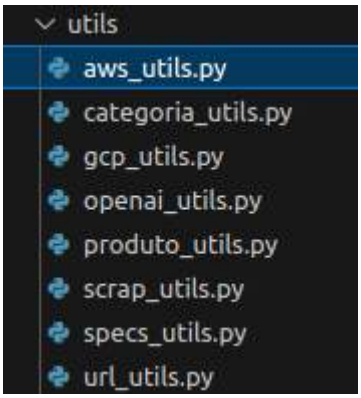

Figura 11: pasta de programas utilitários a serem usados no software.

Os arquivos python da pasta routes possuem scripts que recebem acesso diretamente via HTTP Rest.

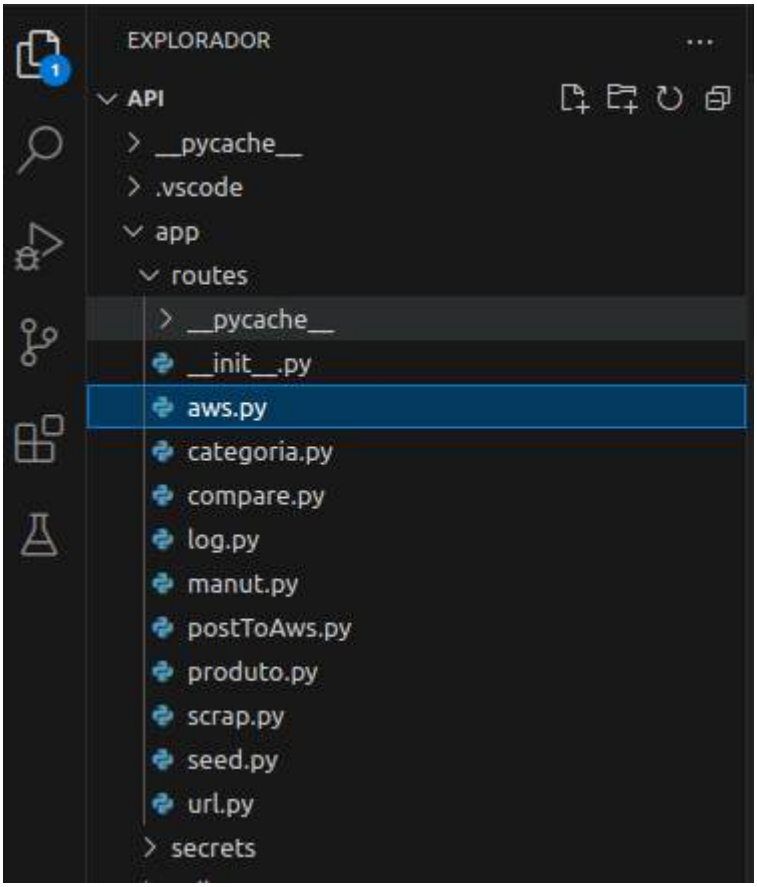

Figura 12: Métodos a serem acessados via HTTP Rest

DISCUSSÃO

O desenvolvimento da ferramenta de comparação de componentes de informática, utilizando técnicas de web scraping para aquisição de informações, mostrou-se uma alternativa interessante em relação aos sistemas tradicionais, que geralmente dependem de repositórios próprios e exigem atualizações frequentes. A escolha da linguagem Python para o backend, encarregada de rastrear as páginas em busca de informações relevantes, foi uma decisão acertada. Python oferece ferramentas robustas, como a biblioteca BeautifulSoup, que possui uma curva de aprendizado suave e ampla documentação disponível, e o framework Scrapy, conhecido por sua eficiência e performance, embora com uma curva de aprendizado mais acentuada. O Selenium foi considerado, mas não implementado, uma vez que as páginas utilizadas forneciam as informações diretamente no HTML, sem a necessidade de técnicas mais avançadas de extração

No frontend, a adoção do React Native foi igualmente eficaz, devido à sua facilidade de configuração e compatibilidade com as plataformas Android e iOS, o que amplia o potencial de adoção. No entanto, surgiram desafios na implementação de componentes específicos, como campos de seleção editáveis e a listagem dos resultados de comparação. Apesar de contar com uma ampla comunidade e diversas bibliotecas auxiliares, houve uma escassez de exemplos práticos para esses casos, o que exigiu o desenvolvimento de soluções personalizadas.

A utilização dos *buckets* S3 para armazenamento de dados estáticos mostrou-se uma opção interessante, com a vantagem de descentralizar o gerenciamento de arquivos e reduzir a carga sobre a aplicação. No entanto, essa abordagem também apresenta desvantagens, como o tráfego entre os componentes da arquitetura AWS, o que pode gerar custos adicionais em um cenário comercial.

Para que a ferramenta seja robusta e viável para uso público, são necessários ajustes importantes. Em primeiro lugar, a conformidade com a legislação sobre *web scraping* é uma preocupação, pois algumas páginas utilizadas como fonte de dados não deixam claro se permitem ou proíbem essa técnica. Além disso, a escalabilidade da ferramenta ainda precisa ser testada de maneira mais rigorosa. Até o momento,

os testes foram realizados em ambiente controlado, sem avaliar sobrecarga ou paralelismo, o que seria essencial para uma adoção em larga escala.

Outro ponto crítico diz respeito à inclusão manual dos parâmetros de comparação para cada categoria de dispositivo. Atualmente, esses parâmetros são adicionados via consultas SQL, um processo exaustivo que, embora viável para um número limitado de categorias, torna-se inviável diante da ampla variedade de componentes de informática disponíveis no mercado.

CONCLUSÃO

A ferramenta desenvolvida para comparação de componentes de informática, baseada em técnicas de *web scraping*, provou ser tecnicamente viável. As escolhas tecnológicas, como o uso de Python para o backend e React Native no frontend, mostraram-se eficientes e adequadas às necessidades do projeto.

Este trabalho tem potencial para contribuir significativamente no contexto acadêmico, ao aplicar técnicas de mineração de dados em uma solução prática e acessível ao público leigo. A adoção de um repositório local para a busca de dados pode ainda reduzir o tempo de resposta, oferecendo um diferencial importante.

Entre os principais desafios identificados, destacam-se as questões éticas e legais relacionadas ao uso de *web scraping* em sites de fabricantes e fornecedores, a escassez de componentes nativos do React Native para funcionalidades específicas, e a falta de testes extensivos para assegurar o desempenho ideal da ferramenta em ambientes mais exigentes.

Para aumentar a autonomia da ferramenta e expandir seu potencial de uso, uma possível melhoria seria a incorporação de inteligência artificial generativa em tarefas específicas, como a definição de parâmetros de avaliação para cada categoria de dispositivos. Atualmente, esses parâmetros são inseridos manualmente no banco de dados, mas com a automação desse processo por meio de IA, seria possível incluir novas categorias de forma automática, aumentando exponencialmente a capacidade de comparação da ferramenta.

REFERÊNCIAS BIBLIOGRÁFICAS

1. ALBERTIN AL. Comércio eletrônico: benefícios e aspectos de sua aplicação. Rev adm empres [Internet]. 1998 Jan; 38(1):52–63. Disponível em: https://doi.org/10.1590/S0034-75901998000100006

2. LAUDON, Kenneth C.; TRAVER, Carol Guercio. E-commerce 2020-2021: business, technology, society. Pearson, 2021.

3. KUNIYOSHI, M. S. Comércio Eletrônico: A revolução em tempos digitais. Revista Administração em Diálogo - RAD, [S. l.], v. 2, n. 1, 2009. DOI: 10.20946/rad.v2i1.1689. Disponível em: https://revistas.pucsp.br/index.php/rad/article/view/1689. Acesso em: 20 abr. 2024.

4. CARVALHO, M. S. R. M. A trajetória da Internet no Brasil: do surgimento das redes de computadores à instituição dos mecanismos de governança. Unpublished Estudos de Ciência e Tecnologia no Brasil, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

5. GHEMAWAT, Pankaj. The laws of globalization and business applications. Cambridge University Press, 2017.

6. SHARMA, A., ADHIKARY, A., & BORAH, S.B. (2021). "Covid-19 and e-commerce: a global perspective." Transnational Corporations Review, 13(2), 224–239.

7. DE MOURA, Delmo Alves. Logística e gerenciamento da cadeia de suprimentos. RAE-Revista de Administração de Empresas, v. 39, n. 1, p. 98-101, 1999.

8. CHRISTOPHER, Martin. Logistics and Supply Chain Management: Logistics & Supply Chain Management. Pearson UK, 2016.

9. FAVERI, Dinorá Baldo de; VALENTIM, Ilda; KROETZ, Marilei. Teoria do Prospecto: uma investigação dos efeitos certeza, reflexão e isolamento na tomada de decisão envolvendo risco. Simpósio de Excelência em Gestão e Tecnologia, v. 10, 2013.

10. PAYNE, John W.; BETTMAN, James R.; JOHNSON, Eric J. The adaptive decision maker. Cambridge university press, 1993.

11. PORTER, Michael E. et al. How smart, connected products are transforming competition. Harvard business review, v. 92, n. 11, p. 64-88, 2014.

12. PRAHALAD, Coimbatore Krishna; RAMASWAMY, Venkat. The future of competition: Co-creating unique value with customers. Harvard Business Press, 2004.

13. SOLOMON, Michael R. Consumer behavior: Buying, having, and being. Pearson, 2020

14. BOWERSOX, Donald J. et al. Supply chain logistics management. Mcgraw-hill, 2020.

15. DA COSTA CÔRTES, Sérgio; PORCARO, Rosa Maria; LIFSCHITZ, Sérgio. Mineração de dados-funcionalidades, técnicas e abordagens. PUC, 2002.

16. JIAWEI, Han; MICHELINE, Kamber. Data mining: concepts and techniques. Morgan kaufmann, 2006.

Assinatura do Orientador