CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

LUCAS WAMSER ROSA DEBETERCO

BUSCAPEÇA - PLATAFORMA WEB DE BUSCA E COMPARAÇÃO DE PREÇOS DE HARDWARE

RIO DO SUL
2023

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

LUCAS WAMSER ROSA DEBETERCO

BUSCAPEÇA - PLATAFORMA WEB DE BUSCA E COMPARAÇÃO DE PREÇOS DE HARDWARE

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Esp. Cleber Nardelli

RIO DO SUL
2023

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

LUCAS WAMSER ROSA DEBETERCO

BUSCAPEÇA - PLATAFORMA WEB DE BUSCA E COMPARAÇÃO DE PREÇOS DE HARDWARE

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: Esp. Cleber Nardelli

Banca Examinadora:

_____

Prof. M.e Fernando Andrade Bastos

_____

Prof. M.e Sandro Alencar Fernandes

Rio do Sul, 27 de novembro de 2023.

Na adversidade uns desistem, enquanto outros batem recordes (Ayrton Senna)

RESUMO

O cenário atual no setor de comércio de produtos de hardware oferece uma ampla gama de lojistas ativos, criando um ambiente desafiador para aqueles que buscam adquirir componentes para montar ou aprimorar um computador. Neste contexto, a proposta deste trabalho é desenvolver uma plataforma web de comparação de preços de hardware. Seu objetivo principal é simplificar e agilizar o processo de busca por este tipo de componente, unificando lojistas do ramo e oferecendo resultados confiáveis e competitivos. Focando em critérios de confiabilidade e preços competitivos, a plataforma oferece resultados de múltiplos lojistas e visa garantir a melhor opção de compra de produtos de hardware. Inicialmente foi realizada uma pesquisa bibliográfica, apresentando um detalhamento sobre os conceitos e tecnologias que foram aplicadas durante a execução do protótipo. O capítulo de desenvolvimento, é composto pelo detalhamento do processo de criação do protótipo, contendo a etapa de engenharia de software, criação de fluxogramas e implementação, onde foi descrito o processo de desenvolvimento técnico do protótipo. Esta solução procura superar os desafios encontrados devido à grande diversidade de lojas no ramo e às limitações dos atuais motores de busca do mercado, os quais muitas vezes não proporcionam os resultados ideais de compra por não priorizarem os lojistas deste ramo. O protótipo, por meio da técnica de pesquisa e extração de dados chamada de *web scraping*, facilita a tomada de decisão no processo de compra de produtos de hardware, oferecendo diversas opções de compra, assegurando uma escolha confiável e um preço competitivo.

Palavras-Chave: Scraping, Hardware, Buscador de Preços.

ABSTRACT

The current scenario in the hardware products trading sector offers a wide range of active retailers, creating a challenging environment for those looking to acquire components to assemble or improve a computer. In this context, the purpose of this work is to develop a web platform for comparing hardware prices. Its main objective is to simplify and speed up the search process for this type of component, unifying retailers in the sector and offering reliable and competitive results. Focusing on reliability criteria and competitive prices, the platform offers results from multiple retailers and aims to guarantee the best option for purchasing hardware products. Initially, a bibliographical research was carried out, presenting details on the concepts and technologies that were applied during the execution of the prototype. The development chapter consists of details of the prototype creation process, containing the software engineering stage, creation of flowcharts and implementation, where the technical development process of the prototype was described. This solution seeks to overcome the challenges encountered due to the great diversity of stores in the sector and the limitations of the market's current search engines, which often do not provide ideal purchasing results as they do not prioritize retailers in this sector. The prototype, through a research and data extraction technique called *web scraping*, facilitates decision-making in the process of purchasing hardware products, offering several purchasing options, ensuring a reliable choice and a competitive price.

Keywords: Scraping, Hardware, Price comparison.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application programming interface
DOM	Document Object Model
HTTP	Hypertext transfer protocol
IBC	Índice de Buscapeça de Confiabilidade
NPM	Node Package Manager
ORM	Object Relational Mapper
PQL	Prisma Query Language
URL	Uniform Resource Locator
VDOM	Virtual Document Object Model

SUMÁRIO

1. INTRODUÇÃO

Montar um computador por um preço atrativo pode ser uma tarefa desafiadora nos dias de hoje. Com a abundância de lojas oferecendo uma ampla variedade de produtos de hardware e os principais motores de busca dificilmente abrangendo todas as opções disponíveis mercado, os consumidores podem se encontrar perdidos em meio a uma infinidade de escolhas. Diante dessa complexidade, surge uma demanda cada vez maior por uma solução que possa unificar os conceituados lojistas do ramo de hardware em uma plataforma web, oferecendo aos consumidores uma experiência simplificada e confiável na busca pelo melhor preço e qualidade.

Ao enfrentar a difícil tarefa de encontrar um novo computador para uso doméstico ou até mesmo para equipar uma empresa, é comum que o comprador leigo se depare com uma variedade de lojistas oferecendo produtos de hardware com características e preços diversos. Nesse contexto, surgem os motores de busca, ferramentas cridas para agrupar todos os lojistas oferecendo um determinado item e listá-los conforme seu preço. Porém, os principais motores de busca muitas vezes mostram limitações em sua capacidade de abranger todas as lojas e apresentar resultados que verdadeiramente atendam às necessidades do consumidor.

É justamente para suprir essa lacuna que surge a proposta de uma plataforma web dedicada à comparação de preços de hardware, o Buscapeça. Essa plataforma terá como objetivo unificar os maiores e melhores lojistas do ramo, proporcionando aos consumidores uma garantia de que encontrarão a melhor opção de compra em se tratando de Hardware. Com o uso da técnica de *web scraping*, de forma direcionada poderá proporcionar uma abrangência mais restrita e resultados mais precisos, proporcionando economia de tempo e esforço, além de oferecer a tranquilidade aos consumidores oferecendo apenas lojas confiáveis do ramo. Com isso, espera-se simplificar e otimizar a experiência de compra de Hardware pelo usuário.

Além do capítulo introdutório, neste trabalho foi realizado um estudo teórico das tecnologias e processos que serão aplicadas durante o desenvolvimento do projeto, apresentando autores que sustentam o explicam o funcionamento de tais ferramentas. No capítulo três, é apresentado o processo pelo qual a aplicação foi planejada e conduzida, fornecendo informações sobre os métodos e procedimentos utilizados no desenvolvimento. O capítulo quatro será composto pelo detalhamento do processo de criação do protótipo, implementação técnica e discussão sobre as tomadas de decisão ao longo do desenvolvimento. Por fim, no último capítulo, são apresentadas as considerações finais.

1.1 PROBLEMA DE PESQUISA

Como possibilitar que o consumidor com foco exclusivo em Hardware de computadores encontre o menor preço disponível em uma loja online confiável?

1.2 OBJETIVOS

1.2.1 Geral

Desenvolver protótipo da plataforma web para comparação de preços com foco exclusivo em Hardware de computador.

1.2.2 Específicos

- Levantar requisitos do projeto;
- Definir tecnologias que serão utilizadas durante o desenvolvimento;
- Desenvolver mecanismo de *web scraping* para a busca dos produtos;
- Desenvolver o protótipo;
- Realizar *deploy* do protótipo;

1.3 JUSTIFICATIVA

No Brasil, é possível observar um uso expressivo de computadores (desktop, notebook e tablet), este tipo de dispositivo tem se mostrado indispensável para a população, seja para uso residencial ou empresarial. Em dezembro de 2022, a quantidade de computadores em utilização ativa no Brasil totalizou 211 milhões. Projeções indicam um aumento para 215 milhões em maio de 2023. Estes números apontam basicamente uma relação de um computador por habitante (MEIRELLES, 2023).

Um desafio comum enfrentado por consumidores que buscam adquirir novos computadores ou componentes de hardware, é a complexidade do processo de compra. Empreendedores que estão iniciando um negócio, usuários domésticos ou até grandes empresas, muitas vezes se deparam com uma vasta gama de lojas especializadas, o que pode causar indecisão e trazer complexidade.

A diversidade de lojistas ofertando produtos no mercado de hardware torna a busca pelo melhor preço uma tarefa difícil. Umas das formas de facilitar este processo, são os

comparadores de preço, este tipo de ferramenta utiliza de motores de busca para realizar um levantamento dos produtos disponíveis na internet e apresentá-los de forma resumida e organizada ao usuário. O problema presente nas ferramentas comparadoras consolidadas no mercado, é a adoção de uma abordagem de pesquisa genérica, sem focar nas lojas especializadas no ramo de hardware.

Tendo este problema em vista, foi proposto o desenvolvimento de uma plataforma web responsável por centralizar as diversas opções de compra presentes no mercado de hardware em um só lugar. Por meio da técnica de *web scraping*, o protótipo busca consolidar e levantar informações das lojas, essa abordagem permite oferecer ao consumidor final uma ferramenta web intuitiva e de fácil utilização, proporcionando uma experiência simplificada, apresentando o melhor preço disponível em uma loja online confiável.

2. REFERENCIAL TEÓRICO

Nesta etapa serão apresentados alguns dados sobre os temas referentes ao tema do projeto a principalmente sobre as tecnologias e ferramentas que serão utilizadas durante o desenvolvimento.

2.1 ENGENHARIA DE SOFTWARE

Sommervile (2011) afirma que a Engenharia de Software é uma área da engenharia que se dedica a todos os aspectos da produção de programas de computador, desde a etapa inicial de especificação do sistema até sua manutenção durante o uso. Seu principal objetivo é obter resultados de qualidade dentro do prazo e orçamento estabelecidos. Essa meta muitas vezes requer que os engenheiros façam compromissos, pois não é possível buscar a perfeição em todos os aspectos do projeto. Por outro lado, pessoas que desenvolvem programas para uso próprio podem gastar o tempo que desejarem no processo de desenvolvimento.

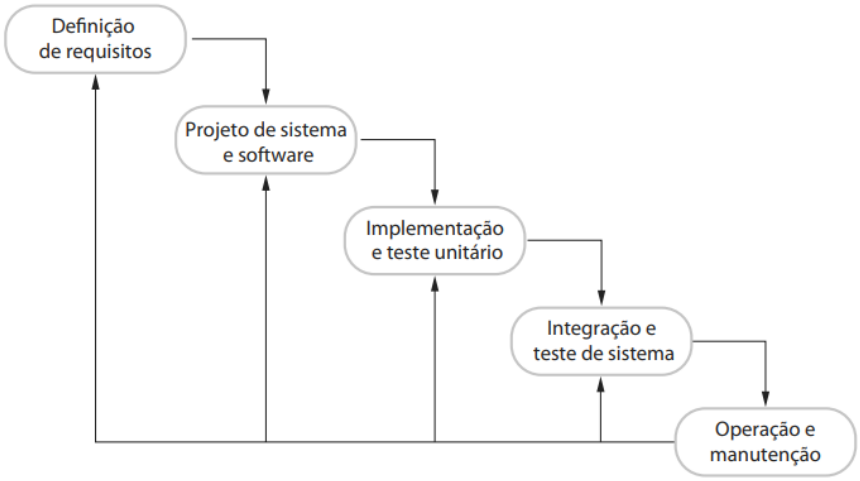
Os engenheiros de software geralmente adotam uma abordagem sistemática e organizada para o trabalho, pois essa é frequentemente a maneira mais eficaz de produzir software de alta qualidade. No entanto, a engenharia também envolve a seleção do método mais adequado para cada conjunto de circunstâncias, e em algumas situações, uma abordagem menos formal e mais criativa pode ser mais eficaz. Por exemplo, o desenvolvimento de sistemas web pode ser mais adequado para uma abordagem menos formal, que envolve uma mistura de habilidades de programação e *design*.

> Engenharia de software é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema e suas funcionalidades, até sua manutenção, quando o sistema já está sendo usado (Sommerville, 2011, p. 5).

2.1.1 Processo de Desenvolvimento

A engenharia de software é o campo que se concentra na aplicação de princípios e métodos e práticas para o desenvolvimento de software. Neste campo, o conceito de implementação de um processo de desenvolvimento bem estruturado é importante para a entrega bem-sucedida do software. Um processo de desenvolvimento eficaz normalmente segue etapas fundamentais, como análise de requisitos, projeto, implementação, testes e manutenção. (SOMMERVILLE, 2011).

Figura 1 - Etapas do processo de desenvolvimento



Fonte: Sommerville (2011, p.7)

Além disso, Sommervile (2011) afirma que diferentes metodologias, como o modelo cascata (Figura 1), o modelo incremental, o desenvolvimento ágil, entre outros, oferecem abordagens variadas para a gestão do desenvolvimento de software, cada uma com seus pontos fortes e limitações. A que análise de requisitos é o ponto inicial, onde as necessidades do usuário e os requisitos do sistema são identificados, documentados e analisados de forma abrangente. Em seguida, o *design* do software é elaborado, com arquiteturas e diagramas estruturam a forma como o sistema será construído. A etapa de implementação traduz esses planos em código, configurando a estrutura funcional do software. Após o lançamento inicial, a manutenção do software é essencial, podendo envolver atualizações, correções de bugs e melhorias contínuas.

O processo de desenvolvimento de software se concentra na aplicação consistente dessas etapas e metodologias para garantir a qualidade, eficiência e entrega oportuna de produtos de software que atendam às necessidades dos usuários e às expectativas do mercado. O objetivo principal é alcançar a satisfação do cliente, produzindo um software funcional, confiável e escalável, atendendo aos padrões de qualidade estabelecidos.

2.1.2 Levantamento de requisitos

Segundo Sommervile (2011), os requisitos de um sistema podem ser definidos como uma lista de funcionalidades ou restrições específicas que devem ser implementadas ou atendidas durante o desenvolvimento do sistema. Esses requisitos descrevem de forma clara e detalhada o que o sistema deve fazer ou como ele deve se comportar em diferentes situações.

> Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado engenharia de requisitos (Sommerville, 2011, p.57).

Os requisitos de software podem ser divididos em duas categorias principais: requisitos funcionais e requisitos não funcionais. Os requisitos funcionais descrevem os serviços que o sistema deve fornecer, como ele deve reagir a entradas específicas e como deve se comportar em determinadas situações. Eles especificam as funcionalidades e recursos que o sistema deve ter para atender às necessidades dos usuários. Em alguns casos, os requisitos funcionais também podem mencionar o que o sistema não deve fazer, definindo limitações ou restrições em determinadas funcionalidades (SOMMERVILLE, 2011).

> A distinção entre diferentes tipos de requisitos não é tão clara como sugerem essas definições simples. Um requisito de usuário relacionado com a proteção, tal como uma declaração de limitação de acesso a usuários autorizados, pode parecer um requisito não funcional. No entanto, quando desenvolvido em mais detalhes, esse requisito pode gerar outros requisitos, claramente funcionais, como a necessidade de incluir recursos de autenticação de usuário no sistema (Sommerville, 2011, p.59).

Já os requisitos não funcionais são restrições ou limitações impostas ao sistema. Eles se referem a aspectos que não estão diretamente relacionados às funcionalidades específicas do sistema, mas sim a restrições de tempo, processos de desenvolvimento, normas ou outros critérios. Os requisitos não funcionais são aplicáveis ao sistema como um todo e podem abranger aspectos como desempenho, segurança, usabilidade, confiabilidade, entre outros (SOMMERVILLE, 2011).

2.2 TECNOLOGIAS MODERNAS DE DESENVOLVIMENTO WEB JAVASCRIPT

De acordo com Duckket (2014), podemos chamar *scripts* de uma série de instruções que um computador pode seguir passo a passo para atingir algum objetivo, o seu navegador pode usar diferentes partes do *script* dependendo de como o usuário interage com a página web. Os *scripts* podem rodar em diferentes seções do código dependendo das ações do usuário.

Para isso, Duckket (2014) afirma que todos os navegadores utilizam a linguagem Javascript para interpretar as interações com uma página web. Para começar a desenvolver um *script* para uma aplicação web, primeiramente precisamos definir um objetivo, e sem seguida uma lista de tarefas que precisamos que sejam completadas para atingirmos este objetivo. Cada

um destes passos precisa ser descrito em uma linguagem de programação, no caso de uma ferramenta web, usaremos o Javascript.

O Javascript permite a criação de página web interativas através do acesso e modificação do conteúdo usado na página web sendo visualizada no navegador. Ele permite que a página web aparenta responder as iterações de usuário executa na página. A linguagem Javascript ainda pode ser utilizada tanto no servidor *back-end* quanto no cliente *front-end* através de bibliotecas como o React, conforme afirma Flanagan:

> Javascript é a linguagem de programação da Web e a maior parte do código Javascript é escrita para navegadores Web. Mas Javascript é uma linguagem de uso geral rápida e competente, não havendo motivos para que não possa ser usada em outras tarefas de programação (FLANAGAN, 2014, p.281).

A linguagem Javascript é principalmente usada no *front-end*, conhecido como lado do cliente, e permite criar uma ampla gama de funcionalidades e melhorias na experiência do usuário. Uma das principais áreas em que o Javascript é usado no lado do cliente e a manipulação do DOM, com essa interação com o Document Object Model (DOM), que representa a estrutura HTML da página, é possível selecionar elementos da página, alterar seu conteúdo, estilo e propriedades, adicionar ou remover elementos dinamicamente, criar animações e muito mais. Essa capacidade de manipulação do DOM é fundamental para criar interatividade e dinamismo nas aplicações web (DUCKKET, 2014).

Além disso, o Javascript permite responder a eventos no *front-end* da aplicação, como cliques, digitação, movimento do mouse e outros eventos do navegador. Com isso, é possível criar interações personalizadas, como validação de formulários, menus expansíveis, carregamento dinâmico de conteúdo, entre outros. Os eventos são utilizados para acionar ações e executar código em resposta às ações do usuário (DUCKKET, 2014).

Diferente do Javascript no lado do cliente, que é executado no navegador do usuário, Flanagan (2014) afirma que o Javascript no lado do servidor permite que o código seja executado no próprio servidor, antes de enviar a resposta para o cliente. O Node.js é uma plataforma de tempo de execução Javascript de código aberto que permite o desenvolvimento de aplicações server-side escaláveis e de alto desempenho. Com o Javascript no lado do servidor, é possível construir uma variedade de aplicações, desde servidores web até aplicativos de linha de comando.

2.3 WEB DESIGN COM A FERRAMENTA FIGMA

Segundo Villain (2022), o Figma é uma plataforma colaborativa para a construção de *design* de interfaces e protótipos, que pertence à empresa Figma, Inc. Lançada em 2016 por Dylan Field e Evan Wallace, o objetivo era criar uma ferramenta gratuita que trouxesse colaboração entre pessoas e times, permitindo criar um produto para as mais diversas plataformas, mantendo a acessibilidade do sistema. A plataforma é usada por *designers* e outros profissionais em todo o mundo para construir o *design* de produtos digitais inteiros, desde sites até aplicativos para dispositivos móveis, como tablets, smartphones ou até mesmo smartwatches.

Villain (2022), complementa dizendo que a plataforma do Figma oferece recursos poderosos para a criação de *design* de interface, como criação de componentes e estilos globais, permitindo que os usuários criem *designs* consistentes em toda a interface. Além disso, a plataforma é altamente personalizável e extensível, permitindo que os usuários adicionem novos recursos e integrações conforme necessários. Em resumo, o Figma é uma ferramenta essencial para *designers* e outros profissionais que desejam criar *designs* de interface e protótipos para uma ampla variedade de plataformas.

> Como toda plataforma de *design* de interface, mas também *design* num geral, o Figma conta com as ferramentas comuns que geralmente *designers* utilizam, como formas geométricas, imagens, vetores, caixas de texto, dentre outras. Você sabe ou imagina o que cada um desses recursos executa, mas o ponto relevante é que no Figma, cada coisa exerce mais funções do que aparenta (VILLAIN, 2019).

Villain (2022) ainda afirma que, com o Figma, é possível criar protótipos interativos e simular fluxos completos, permitindo que os *designers* e outros profissionais possam explorar o máximo possível do *design* de interface. A plataforma é conhecida por sua facilidade de uso e sua capacidade de colaboração em tempo real, permitindo que as equipes trabalhem juntas em projetos em tempo real, mesmo que estejam em locais diferentes.

> É super comum que um único produto tenha diversos fluxos diferentes, afinal, quanto mais funções e níveis de responsabilidade, mais processos vão acabar nascendo internamente. Com o Figma, é possível definir diversos fluxos diferentes usando o recurso Flow, e coletar o link de início para cada fluxo para separar informações diferentes (VILLAIN, 2019).

Algo proporcionado pela ferramenta que pode ajudar durante o desenvolvimento da aplicação, são os recursos para desenvolvedores. Na verdade, *designers* de interface estão

constantemente colaborando com programadores, que são responsáveis por transformar as telas projetadas em realidade. O Figma oferece recursos que facilitam essa colaboração. Uma dessas funcionalidades é a área de inspeção, que permite visualizar todos os detalhes de um elemento, como suas dimensões, espaçamentos, cores e tipografia. Além disso, o Figma gera uma pré-visualização de código para CSS, Android e iOS, o que melhora a comunicação entre *designers* e programadores durante a fase de implementação do projeto (VILLAIN, 2019).

Essa área de inspeção e visualização de código facilita o processo de transferência do projeto do *designer* para o programador, conhecido como "handoff". Ao fornecer informações detalhadas sobre cada elemento da interface, o Figma permite que o *designer* compartilhe com precisão as especificações necessárias para a implementação (VILLAIN, 2019).

2.4 PRISMA ORM (OBJECT RELATIONAL MAPPER)

Abba (2022) afirma que um Object Relational Mapper (ORM) é uma ferramenta que permite aos desenvolvedores interagirem com bancos de dados relacionais usando linguagens de programação orientadas a objetos. Em outras palavras, um ORM é um software que mapeia objetos de uma aplicação em tabelas de um banco de dados relacional. Tradicionalmente, as aplicações web são desenvolvidas utilizando linguagens de programação orientadas a objetos, como Java, Python, Ruby e c#. No entanto, os bancos de dados relacionais, como o MySQL, o PostgreSQL e o Oracle, armazenam dados em tabelas com colunas e linhas. Esse modelo de armazenamento de dados pode ser bastante diferente da forma como os desenvolvedores de software pensam sobre dados em suas aplicações.

Segundo Fonseca (2019), um ORM é um intermediário entre a aplicação e o banco de dados. Ele permite que os desenvolvedores trabalhem com objetos em sua aplicação, sem ter que lidar diretamente com o banco de dados relacional subjacente. O ORM é responsável por mapear os objetos em tabelas de banco de dados, criar consultas SQL, executá-las e retornar os resultados como objetos para a aplicação. O objetivo do ORM é simplificar o processo de desenvolvimento de aplicativos e reduzir a quantidade de código que precisa ser escrito.

> As bibliotecas ou *frameworks* ORM definem o modo como os dados serão mapeados entre os ambientes, como serão acessados e gravados. Isso diminui o tempo de desenvolvimento, uma vez que não é necessário desenvolver toda essa parte. Outra vantagem está na adaptação de novos membros na equipe, como muitos projetos comerciais utilizam a mesma ferramenta, é possível encontrar membros que já estão acostumados com o padrão de trabalho (FONSECA, 2019)

Segundo Fonseca (2019), os ORM têm muitas vantagens. Eles permitem que os desenvolvedores escrevam menos código, reduzam erros e aumentem a produtividade. Os ORMs também tornam as aplicações mais fáceis de manter, pois os desenvolvedores podem fazer alterações no esquema do banco de dados sem ter que alterar o código da aplicação. Além disso, os ORMs fornecem um alto nível de abstração, o que torna o código da aplicação mais fácil de entender e modificar. No entanto, os ORMs também têm algumas desvantagens. Eles podem ser menos eficientes em termos de desempenho do que escrever SQL manualmente, especialmente para consultas complexas. Além disso, os ORMs podem ter uma curva de aprendizado íngreme para os desenvolvedores que não estão familiarizados com eles. Finalmente, os ORMs podem limitar a flexibilidade do desenvolvedor em relação ao banco de dados, já que algumas funcionalidades do banco de dados podem não estar disponíveis através do ORM.

Em resumo ORM é uma ferramenta que permite aos desenvolvedores trabalharem com bancos de dados relacionais usando objetos em sua aplicação. Ele simplifica o processo de desenvolvimento de aplicativos, reduz erros e aumenta a produtividade. No caso do trabalho em questão, será utilizado o ORM Prisma, desenvolvido em Javascript. Na ferramenta Prisma iremos utilizar Node.js e Typescript para a criação e modelagem do banco de dados.

> Dividido em três camadas como núcleo de sua arquitetura, o Prisma nasceu no ecossistema Javascript com a promessa de ser uma ferramenta facilitadora e produtiva para devs que trabalham diretamente com databases. Por uma série de razões, a tecnologia chegou a ser reconhecida como “uma das melhores coisas que já aconteceu” na programação *back-end* entre usuários de Node.js (BUZZI, 2022).

Buzzi (2022), ainda afirma que, dentro da ferramenta Prisma, existe uma biblioteca chamada Prisma Client que é gerada automaticamente a partir do schema do banco de dados definido pelo desenvolvedor. O Prisma permite que os desenvolvedores escrevam consultas e manipulem o banco de dados usando uma sintaxe amigável, conhecida como Prisma Query Language (PQL). Com o Prisma Client, também é possível executar consultas complexas, realizar operações de criação, leitura, atualização e exclusão (CRUD) e utilizar recursos avançados, como paginação, ordenação e filtragem de dados.

2.5 JAVASCRIPT NO BACK-END (NODEJS)

Rungta (2016) afirma que o Node.js é um ambiente de execução de código aberto multiplataforma que possibilita o desenvolvimento de aplicações web do lado do servidor. Os

aplicativos Node.js são escritos em Javascript e podem ser executados em diferentes sistemas operacionais. Ele é baseado em uma arquitetura orientada a eventos e uma API de entrada/saída sem bloqueio, que visa maximizar a eficiência e escalabilidade de aplicativos para web em tempo real. O Node.js é construído em cima do mecanismo de Javascript do Google, o V8, que é conhecido por sua velocidade e eficiência. Com o Node.js, os desenvolvedores podem criar aplicativos de servidor, APIs e serviços da web usando Javascript em vez de linguagens de servidor tradicionais, como Java, Python ou Ruby.

Rungta (2016) ainda afirma que por muito tempo, os *frameworks* disponíveis para o desenvolvimento web se basearam em um modelo sem estado, onde os dados gerados em uma sessão não eram mantidos para uso futuro. Com o Node.js, porém, finalmente existe uma solução para as aplicações web terem uma conexão bidirecional em tempo real, permitindo que tanto o cliente quanto o servidor iniciem a comunicação e troquem dados livremente, eliminando a necessidade de se realizar muito trabalho para manter as informações da sessão entre as solicitações do usuário.

O Node.js permite que os desenvolvedores usem um único idioma em todo o stack, o que pode tornar o desenvolvimento mais rápido e fácil de entender. O uso do Javascript no *back-end* tem muitas vantagens. Uma delas é que os desenvolvedores não precisam aprender uma nova linguagem de programação para desenvolver o *back-end* da aplicação. Eles podem usar as mesmas ferramentas e técnicas que usam para desenvolver o *front-end*, o que pode acelerar o processo de desenvolvimento. Além disso, o Node.js é altamente escalável e pode lidar com muitas solicitações de clientes simultâneas sem afetar o desempenho da aplicação. (FLANAGAN, 2014).

> O Node é um interpretador Javascript rápido, baseado em C++, com vínculos para as APIs Unix de baixo nível para trabalhar com processos, arquivos, soquetes de rede, etc., e também para cliente HTTP e APIs de servidor. A não ser por alguns métodos síncronos com nomes especiais, os vínculos do Node são todos assíncronos e, por padrão, os programas Node nunca são bloqueados, isso quer dizer que normalmente mudam bem de escala e lidam com cargas grandes de forma eficiente (FLANAGAN, 2014, p.288).

Segundo Mbathi (2013), as principais vantagens de se trabalhar com o Node é a possibilidade de utilização de várias bibliotecas de terceiros para executarem tarefas que, muitas vezes não precisam ser codificadas novamente. Existem muitos *frameworks* Javascript diferentes disponíveis que podem tornar o desenvolvimento de *back-end* mais simples e eficiente.

Flanagan (2014), afirma que os *frameworks* podem ser usados por vários motivos, mas os dois mais comuns são para economizar tempo e impor padrões. Quando você usa uma estrutura, está basicamente reutilizando código escrito e testado anteriormente. Você economiza tempo porque não precisa começar do zero. Enquanto isso, impor padrões é vantajoso para os desenvolvedores porque elimina a necessidade de se preocupar com pequenos detalhes. Também é benéfico para os usuários porque significa que eles podem esperar uma experiência consistente em vários aplicativos.

O Node.js também vem com muitas bibliotecas e módulos que os desenvolvedores podem usar para criar aplicativos. Por exemplo, durante o desenvolvimento deste projeto, será utilizado o Express é um popular *framework* de *back-end* para Node.js que simplifica o processo de criação de aplicativos da web. Ele fornece recursos como roteamento, middleware e manipulação de solicitações e respostas HTTP (FLANAGAN, 2014).

2.5.1 Express

Todas as rotas presentes no *back-end* da aplicação, serão criadas e estruturadas pela ferramenta Express.js, um *framework* popular e amplamente utilizado em combinação com o Node.js, fornecendo uma maneira fácil de desenvolver aplicativos *back-end* e, em conjunto com sistemas de templates, até mesmo aplicativos full-stack. Desenvolvido em Javascript, o Express.js é adotado por grandes empresas mundialmente. (ANDRADE, 2021)

Andrade (2021) afirma que, tanto nas grandes empresas quanto na comunidade em geral, o Express é reconhecido como um *framework* poderoso que simplifica o processo de criação de aplicativos usando o Node.js e Javascript, oferecendo uma ampla gama de recursos que tornam o desenvolvimento de APIs mais facilitado, principalmente através de um sistema de roteamento completo que permite gerenciar as rotas da aplicação de forma organizada, da capacidade de lidar com exceções dentro da aplicação, facilitando a manipulação de erros e a implementação de tratamentos adequados, integração com sistemas de templates, que tornam mais simples a criação de páginas web para suas aplicações, Suporte para gerenciar diferentes tipos de requisições HTTP e conjunto reduzido de arquivos e pastas.

Essas características do Express.js tornam-no um *framework* efetivo, proporcionando facilidades significativas no desenvolvimento de aplicações. Ele simplifica a construção de rotas, o tratamento de erros e a gestão de requisições HTTP, em um ambiente de rápida criação de aplicativos usando um conjunto enxuto de recursos (ANDRADE, 2021).

2.5.2 Web Scraping

A coleta automatizada de informações da internet tem uma longa história que remonta aos primeiros dias da própria internet. Embora o conceito de coletar dados da web não seja novo, nos últimos anos, essa prática foi denominada de várias maneiras, como *screen scraping*, *data mining*, *web harvesting* e outras variações similares. Atualmente, o termo mais amplamente aceito é *web scraping* (MITCHELL, 2015).

Na essência, o *web scraping* refere-se à ação de extrair informações de uma fonte online sem a necessidade de um programa interagir diretamente com uma API (Interface de Programação de Aplicativos), ou até, sem a intervenção de um usuário que utiliza um navegador da web para coletar os dados. Isso implica em desenvolver métodos e técnicas automatizadas para acessar e extrair dados de páginas da web de forma sistemática, o que pode ser útil para uma ampla gama de aplicações, desde análise de mercado até pesquisa acadêmica. É importante notar que a raspagem da web pode ser uma área legalmente delicada, uma vez que envolve a coleta de dados de recursos online, e pode haver regulamentos e políticas específicos a serem considerados ao realizar essa prática (MITCHELL, 2015).

A atividade de *web scraping* é tipicamente realizada através da criação de um programa automatizado que se conecta a um servidor web, faz solicitações de dados (geralmente obtendo respostas na forma de HTML e outros elementos que constituem páginas da web) e, em seguida, processa esses dados para extrair as informações desejadas. Este processo envolve a elaboração de um código de computador que simula a ação humana de visitar um site, navegando por suas páginas e coletando os dados necessários para análise. Esses dados podem ser utilizados em diversas áreas, desde a extração de informações de preços de produtos em lojas online até a coleta de dados para pesquisa acadêmica (MITCHELL, 2015).

> Os web scrapers são excelentes para reunir e processar grandes quantidades de dados (entre outras coisas). Em vez de visualizar uma página de cada vez através da janela estreita de um monitor, você pode visualizar bancos de dados abrangendo milhares ou até milhões de páginas de uma vez (Mitchell, 2015, p.10).

A prática de *web scraping* abrange uma variedade de técnicas e ferramentas de programação que incluem habilidades em análise de dados e preocupações com a segurança da informação. Isso significa que, ao realizar *web scraping*, é necessário dominar uma série de métodos e tecnologias que vão desde a capacidade de analisar os dados coletados até garantir que a operação seja segura. A análise de dados é crucial para extrair informações úteis dos dados brutos obtidos da web (MITCHELL, 2015).

2.5.3 Scraping e APIs

Em geral, é mais aconselhável utilizar uma API, caso disponível, em vez de desenvolver um processo de *web scraping* para acessar os mesmos dados. No entanto, há várias situações em que uma API pode não estar disponível ou não ser a escolha ideal. Alguns exemplos pode ser quando você precisa coletar dados de uma variedade de sites que não possuem uma API unificada para acesso eficiente, quando os dados desejados são relativamente pequenos em escala e complexidade, tornando a criação de uma API desnecessária do ponto de vista do administrador do site, quando a fonte não possui a infraestrutura técnica ou conhecimento necessário para desenvolver uma API (MITCHELL, 2015).

Mesmo quando uma API está disponível, ela pode ter limitações, como restrições de volume e taxa de solicitações, formatos de dados inadequados ou tipos de informações limitados, que podem não atender às suas necessidades específicas. Portanto, a escolha entre o uso de uma API ou *web scraping* depende das circunstâncias individuais e dos requisitos do projeto, considerando fatores como a disponibilidade de APIs, a complexidade dos dados desejados e as limitações impostas pelas fontes de dados (MITCHELL, 2015).

2.5.4 Web Crawler

Um *web crawler*, também conhecido como bot de mecanismo de busca, tem a função de baixar e indexar informações de diversas partes da internet. Sua finalidade principal é analisar o conteúdo de páginas na web, a fim de identificar seu conteúdo, de modo que essas informações possam ser facilmente recuperadas quando necessário. Os mecanismos de busca normalmente controlam esses bots, usando algoritmos para analisar os dados obtidos pelos *web crawlers*, o que permite apresentar links, imagens ou qualquer tipo de dados pertinentes quando o usuário realizar a pesquisa (CLOUDFLARE, 2023).

Este tipo de ferramenta permite que os mecanismos de busca forneçam resultados relevantes aos usuários quando fazem algum tipo de pesquisa na internet, pois tem a capacidade de fornecer um índice atualizado das informações disponíveis online. Os *crawlers* comumente precisam revisitar regularmente as páginas para garantir que a versão mais recente do conteúdo seja indexada, uma vez que o conteúdo na web está em constante mudança (CLOUDFLARE, 2023).

2.6 NEXT.JS

O Next.js é um *framework* para desenvolvimento de aplicativos da web baseados em React. Ele se destaca por sua flexibilidade e facilidade de uso, e atualmente, se encontra como uma das principais escolhas entre desenvolvedores que desejam criar aplicativos da web modernos, rápidos e eficientes. O que diferencia o Next.js é a sua capacidade de lidar com muitas das complexidades que podem surgir ao criar aplicativos web, tornando o processo de desenvolvimento da aplicação mais simples e produtivo (NEXTJS, 2023).

A estrutura do *framework* Next.js oferece uma série de funcionalidades essenciais para o desenvolvimento de aplicativos da web. Essas ferramentas incluem recursos como roteamento, renderização do lado do servidor (SSR), renderização estática (SSG), otimizações de desempenho e ferramentas de desenvolvimento. Essas funcionalidades simplificam a criação de aplicativos web complexos e permitem que os desenvolvedores foquem mais na lógica de negócios do que na configuração técnica (NEXTJS, 2023).

> Ele tem suporte para divisão de código, ou code splitting enquanto com o create react app combina todos os arquivos Javascript em um único pacote, com o next quando ao lidar com roteamento, estamos apenas carregando o Javascript usado naquela página e isso traz um ganho de performance (Nascimento, 2022).

Uma das vantagens significativas do Next.js é a melhoria da experiência do desenvolvedor. Ele facilita muitas das tarefas de configuração complexas e repetitivas que podem ser encontradas no desenvolvimento React tradicional. Isso significa que o processo de desenvolvimento pode ser mais produtivo. Além disso, o Next.js oferece um ambiente de desenvolvimento facilitado, incluindo recursos como atualização automática da página, permitindo que o desenvolvedor da aplicação veja as alterações em tempo real, além de oferecer suporte a *hot-reloading*, que acelera o ciclo de desenvolvimento (NEXTJS, 2023).

O Next.js permite a adoção gradual dos recursos em um aplicativo React já existente. Isso é uma grande vantagem para desenvolvedores que desejam migrar seus aplicativos para o Next.js sem precisar reescrever tudo desde o início. É possível começar com uma parte do aplicativo e, à medida que se familiariza com os recursos e benefícios do Next.js, expandir o uso da estrutura para outros aspectos do projeto. O Next.js lida com muitos requisitos comuns encontrados em aplicativos web, como roteamento de páginas, busca e integração de dados. Com o roteamento integrado, é possível criar URLs e gerenciar a navegação de forma intuitiva. Além disso, ele facilita a busca e busca de dados, permitindo integrar dados de APIs e outras fontes facilmente (NEXTJS, 2023).

2.6.1 SSR (Server Side Rendering)

O Next.js é projetado para oferecer alto desempenho, principalmente graças um recurso chamado SSR, a renderização do lado do servidor é uma das características distintivas do Next.js e desempenha um papel crucial no desempenho e na experiência do usuário em aplicativos da web. A renderização do lado do servidor é uma técnica de renderização de páginas da web em que a renderização ocorre no servidor antes de a página ser enviada para o navegador do usuário. Isso é diferente da abordagem tradicional de renderização no lado do cliente, em que o navegador do usuário é responsável por renderizar a página (NEXTJS, 2023).

No contexto do Next.js, o SSR significa que, quando um usuário acessa uma página da web, a página é pré-renderizada no servidor interno do Next.js com os dados necessários e, em seguida, o HTML da página é enviado para o navegador e apresentado ao usuário (NEXTJS, 2023).

> No que diz respeito à arquitetura cliente-side, ou lado do cliente em tradução literal, uma vez que carregamos o conteúdo temos que esperar o pacote que contém todo o Javascript ou bundle como é conhecido carregar antes de determinar o que mostrar na página. E isso torna o tempo de carregamento mais alto o que acaba virando problema para pessoas com dispositivos móveis antigos e conexões fracas (Nascimento, 2022).

Algumas das vantagens deste tipo de implementação são o tempo de apresentação do conteúdo da página, o que significa que o usuário final vê o conteúdo mais rapidamente. Isso é particularmente benéfico para páginas com muito conteúdo ou complexidade, pois evita um tempo de carregamento inicial mais longo. Os mecanismos de busca podem facilmente rastrear e indexar o conteúdo das páginas SSR, melhorando a visibilidade da aplicação desenvolvida em Next.js em resultados de pesquisa. Outra vantagem que a renderização no servidor proporciona, é o carregamento dos *scripts* da página de forma mais rápida, pois os usuários não precisam esperar que o Javascript seja baixado e executado para ver o conteúdo e as funções da página (NEXTJS, 2023).

2.6.2 ReactJS

O React.js é uma biblioteca de *front-end* de código aberto, baseada em componentes, responsável apenas pela camada de visualização do aplicativo. A biblioteca React é um conjunto de ferramentas Javascript que tem como foco principal a construção de interfaces de usuário (UI). Com o React.js, é possível criar interfaces interativas de forma mais simples e intuitiva. Ele se destaca por sua eficiência na renderização de páginas, pois atualiza apenas os

componentes necessários à medida que eles sofrem alterações. Com o React, os desenvolvedores podem criar componentes reutilizáveis e modularizados, o que facilita a construção e manutenção de interfaces complexas. Os componentes são blocos independentes que possuem sua própria lógica e estrutura, podendo ser combinados para formar uma interface completa. (REACT, 2023).

Uma das principais vantagens do React.js é sua capacidade de atualizar apenas as partes da interface que foram modificadas, ao invés de renderizar toda a página novamente. Isso torna o React.js extremamente eficiente, pois evita a necessidade de atualizar e recarregar todo o conteúdo, resultando em uma experiência mais rápida e responsiva para o usuário. Além disso, o React.js adota uma abordagem baseada em fluxo de dados unidirecional, o que significa que os dados fluem de cima para baixo na hierarquia dos componentes. Isso simplifica o gerenciamento de estado e torna mais fácil entender como os dados são manipulados e atualizados na interface. O React.js é uma biblioteca Javascript poderosa e eficiente, projetada para simplificar a criação de interfaces de usuário interativas. Com sua abordagem baseada em componentes reutilizáveis e renderização eficiente, o React.js é uma escolha popular para o desenvolvimento de UIs modernas e responsivas. (REACT, 2023).

O React.js permite escrever componentes usando uma linguagem específica de domínio chamada JSX. Por sua vez, o JSX permite escrever componentes usando HTML em conjunto com eventos Javascript. O React.js converterá isso internamente em um DOM virtual e, por fim, produzirá o HTML da página. O React.js “reage” às alterações de estado em seus componentes de forma rápida e automática para renderizar novamente os componentes no HTML DOM utilizando o DOM virtual. O DOM virtual é uma representação na memória de um DOM real. Fazendo a maior parte do processamento dentro do DOM virtual em vez de diretamente no DOM do navegador, o React.js pode agir rapidamente e apenas adicionar, atualizar e remover componentes que foram alterados desde o último ciclo de renderização. (Kowalczyk, 2019).

Flanagan (2014) ainda afirma que a linguagem Javascript é empregada para manipular documentos web produzidos com o HTML. Entretanto, é crucial que se utilize esta linguagem de modo moderado e com controle. O objetivo do Javascript é otimizar a experiência de navegação do usuário, tornando-a mais ágil e propiciando uma transmissão de informações mais fluída. Dentre as maneiras de se alcançar tal meta, estão a criação de animações e efeitos visuais, a organização das colunas de tabelas e a ocultação de conteúdos desnecessários. Se o Javascript for empregado de forma apropriada, pode-se obter melhor qualidade e eficiência na navegação em páginas web.

> Assim como o React é simples de ser escrito e compreendido, ele também é muito fácil de ser utilizado. Um programador jovem com uma boa base de conhecimento em Javascript já é capaz de manipular React com desempenho satisfatório em pouquíssimo tempo. (Roveda, 2023).

Segundo Kowalczyk (2019), o React.js usa o mecanismo baseado em DOM virtual para preencher os dados (visualizações) no HTML DOM. O DOM virtual funciona rápido devido ao fato de que ele apenas altera elementos individuais do DOM em vez de recarregar o DOM completo toda vez. Um aplicativo React.js é composto de vários componentes, cada um responsável pela saída de um pequeno pedaço reutilizável de HTML. Os componentes podem ser aninhados dentro de outros componentes para permitir que aplicativos complexos sejam construídos a partir de blocos de construção simples.

Sobre o DOM vitual, Roveda (2023) complementa, dizendo que o React.js utiliza o Virtual DOM (VDOM) como uma abordagem eficiente para lidar com os desafios relacionados à atualização de telas na web. Essa tecnologia foi desenvolvida com o objetivo de superar os problemas comuns encontrados na atualização de telas reais. O VDOM é uma representação virtual da estrutura da página, que o React.js manipula e compara para determinar as alterações necessárias. Em vez de atualizar diretamente a tela real a cada mudança, o React.js atualiza o VDOM de forma rápida, identificando as diferenças entre a versão anterior e a nova.

Uma vez que as alterações são detectadas no VDOM, o React.js aplica apenas as mudanças necessárias na tela real, minimizando a quantidade de operações de atualização e melhorando o desempenho geral da aplicação. Isso resulta em atualizações mais rápidas, proporcionando uma experiência de usuário mais fluida (ROVEDA, 2023).

2.6.3 Tailwind

O Tailwind é uma API projetada para melhorar o processo de *design* e desenvolvimento de interfaces. Ele oferece classes utilitárias que permitem trabalhar dentro das diretrizes de um sistema de *design*, em vez de sobrecarregar suas folhas de estilo com valores arbitrários. Essas classes utilitárias tornam mais fácil manter a consistência em relação a escolhas de cores, espaçamento, tipografia, sombras e outros elementos que constituem um sistema de *design* bem estruturado (TAILWIND, 2023).

Uma característica notável do Tailwind é seu nível de abstração bastante baixo, o que significa que não força o usuário a repetir o processo de *design* em cada novo projeto. Mesmo quando utilizada a mesma paleta de cores e escalas de dimensionamento, o Tailwind permite construir componentes com aparências completamente diferentes em projetos subsequentes.

Além disso, o Tailwind possui uma otimização automática que remove qualquer CSS não utilizado durante o processo de construção para produção. Isso garante que o pacote final de CSS enviado para o cliente seja o menor possível, muitas vezes com menos de 10kB (TAILWIND, 2023).

Uma funcionalidade prática é a capacidade de criar *designs* responsivos diretamente em seu HTML, adicionando tamanhos de tela às classes de utilitários. Isso elimina a necessidade de lidar com consultas de mídia complexas em seu CSS. Se houver a necessidade de estar repetindo os mesmos utilitários várias vezes, é possível extrair essas partes em um componente ou modelo parcial, criando uma única fonte de verdade onde as alterações podem ser feitas de forma centralizada. Isso ajuda a manter o código organizado e fácil de manter (TAILWIND, 2023).

2.7 TYPESCRIPT

Sobre o Typescript, Adriano (2021) afirma que, se trata de uma extensão do Javascript desenvolvida pela Microsoft, e atua como um *superset* que oferece recursos adicionais e vantagens ao desenvolvimento de software. Esta ferramenta tem foco na simplificação e escalabilidade de códigos, e se baseia em princípios de Orientação a Objetos, introduzindo conceitos como classes, tipagem estática, interfaces e Generics. Esses elementos enriquecem o Javascript, proporcionando maior robustez e facilitando a construção de aplicações mais complexas, mantendo a flexibilidade e versatilidade do Javascript. A inclusão de tipagem estática é um dos diferenciais mais significativos do Typescript. Isso permite a definição de tipos para variáveis, parâmetros de função e estruturas de dados, tornando a detecção de erros mais precoce e eficiente, o que reduz potenciais falhas durante o desenvolvimento e agiliza o processo de *debug*. A tipagem estática torna o código mais compreensível e auto documentado, melhorando a clareza e a manutenibilidade.

Além disso, Adriano (2021) complementa dizendo que o Typescript oferece suporte a ferramentas de desenvolvimento robustas, como o *autocomplete* e sugestões de código, facilitando o trabalho dos desenvolvedores. Com a validação de tipos durante a escrita do código, é possível identificar possíveis inconsistências e erros antes mesmo da execução.

> O compilador do Typescript é altamente configurável. Ele nos permite definir o local onde estão os arquivos .ts dentro do nosso projeto, o diretório de destino dos arquivos transpilados, a versão ECMAScript que será utilizada, o nível de restrição do verificador de tipos e até se o compilador deve permitir arquivos Javascript. (ADRIANO, 2021, p.10).

Sobre o processo de transpilação do código Typescript, Adriano (2021) finaliza dizendo que os navegadores não têm a capacidade de interpretar diretamente o código escrito em Typescript, pois essa linguagem, embora seja um *superset* do Javascript, apresenta recursos e sintaxes específicas que os navegadores não reconhecem. Por esse motivo, é essencial transpilar o código Typescript para uma versão válida do ECMAScript. Esse processo de transpilação é fundamental para converter o código Typescript, com suas estruturas de tipagem estática, interfaces e outras funcionalidades, em um código Javascript puro, compreensível e executável pelos navegadores. Isso é realizado por meio de compiladores Typescript, que transformam o código Typescript em Javascript correspondente, mantendo a lógica do programa e garantindo a compatibilidade com os navegadores e ambientes onde a aplicação será executada (ADRIANO, 2021).

3. METODOLOGIA DA PESQUISA

O presente trabalho de conclusão de curso caracteriza-se como pesquisa aplicada, com os objetivos de desenvolver um sistema capaz de realizar uma comparação de preços de produtos de hardware nos principais lojistas do mercado com este foco. O trabalho busca responder a seguinte pergunta: Como possibilitar que o consumidor de Hardware encontre o menor preço disponível em uma loja on-line confiável?

Inicialmente será realizado o planejamento do projeto, utilizando técnicas de levantamento de requisitos, para organizar todos os processos presentes na aplicação e os requisitos essenciais para o sistema. Este levantamento também irá considerar a diversidade de lojas e os pontos fracos dos motores de busca existentes, buscando soluções que atendam de forma eficiente às demandas dos consumidores.

Após o levantamento de requisitos, inicia-se o processo de criação de um protótipo da aplicação. A ferramenta Figma é utilizada nessa etapa para visualizar e organizar o *design* da plataforma, dando uma ideia clara do funcionamento e da experiência do usuário. Com o protótipo visual finalizado, será dado início ao desenvolvimento da plataforma. Será utilizado um conjunto de ferramentas de desenvolvimento web Javascript, o Next.js e Node.js.

Estas modernas tecnologias permitem a criação de uma aplicação desacoplada e escalável. O Next.js será utilizado para a construção da interface do usuário, oferecendo uma experiência interativa e responsiva. Já o Node.js é utilizado para o desenvolvimento do *back-end*, lidando com a lógica de negócios e integração com os diferentes lojistas.

Após a conclusão do desenvolvimento e dos testes, a plataforma será disponibilizada em um ambiente de produção. Nesse contexto, a plataforma Vercel será utilizada em conjunto com o serviço EC2 da Amazon para hospedagem, garantindo uma infraestrutura confiável e escalável para a aplicação. A Vercel oferece recursos que permitem a disponibilização rápida e segura da plataforma web, assegurando que os usuários possam acessá-la de maneira ágil e confiável, além de permitir *deploy* automático, facilitando qualquer alteração pós liberação ou melhoria no código.

No contexto metodológico deste trabalho, após uma introdução inicial, foi conduzido um estudo teórico abrangente das tecnologias e processos fundamentais para o desenvolvimento do projeto. Este estudo inclui a revisão da literatura, destacando as contribuições de diversos autores que fundamentam e explicam o funcionamento das ferramentas selecionadas. No terceiro capítulo, detalha-se minuciosamente o processo de planejamento e execução da aplicação, oferecendo uma visão completa dos métodos e procedimentos adotados durante o

desenvolvimento. O quarto capítulo concentra-se na exposição detalhada do processo de criação do protótipo, abordando a implementação técnica e promovendo uma discussão aprofundada sobre as decisões estratégicas tomadas ao longo do desenvolvimento. Encerrando o trabalho, o último capítulo contempla as considerações finais, consolidando as principais descobertas e reflexões geradas ao longo da pesquisa e implementação do projeto.

3.1 ESTADO DA ARTE

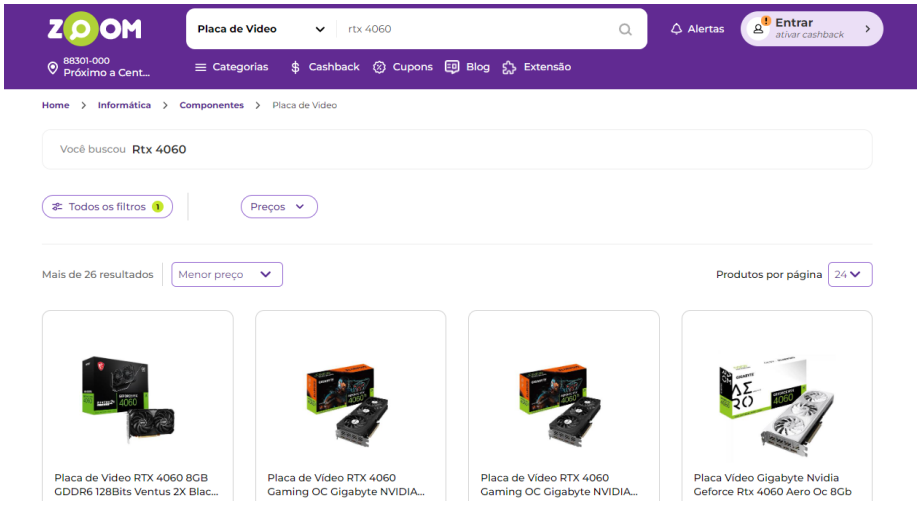
Nesta etapa, será apresentado um resumo do panorama atual do mercado de comparadores de preço na web. Serão apresentados os principais serviços que possuem um objetivo semelhante ao tema do presente trabalho. Ambas as ferramentas de comparação apresentadas nesta etapa do estudo, diferentemente da aplicação proposta aqui, não possuem como finalidade única e específica produtos de hardware de computador. Não foram encontradas ferramentas no mercado que possuam o mesmo escopo de pesquisa da aplicação proposta por este trabalho.

3.1.1 Zoom

A ferramenta de comparação de preços Zoom é um serviço online que permite aos consumidores pesquisarem e compararem preços de produtos em várias lojas. Com base nas preferências do usuário e nas informações fornecidas pela pesquisa, o Zoom oferece uma visão abrangente das opções de compra disponíveis, destacando os preços, disponibilidade e características do produto.

A ferramenta realiza uma análise rigorosa, estudando o histórico de cada loja individualmente antes de recomendar opções de compra. Com mais de um ano de experiência no mercado, o Zoom coloca a segurança em primeiro lugar, aderindo a padrões rigorosos e regulamentações, incluindo conformidade com a Receita Federal, respeito às diretrizes do PROCON (órgão de defesa do consumidor) e cuidados com a análise de crédito através da SERASA (ZOOM, 2023). O Zoom também oferece recursos adicionais, como alertas de preços e notificações de descontos, para manter os usuários atualizados sobre as melhores oportunidades de compra. Na figura 2, é apresentada a página de pesquisa da plataforma Zoom, com os resultados da busca por uma placa de vídeo.

Figura 2 - Zoom (Página - Pesquisa)



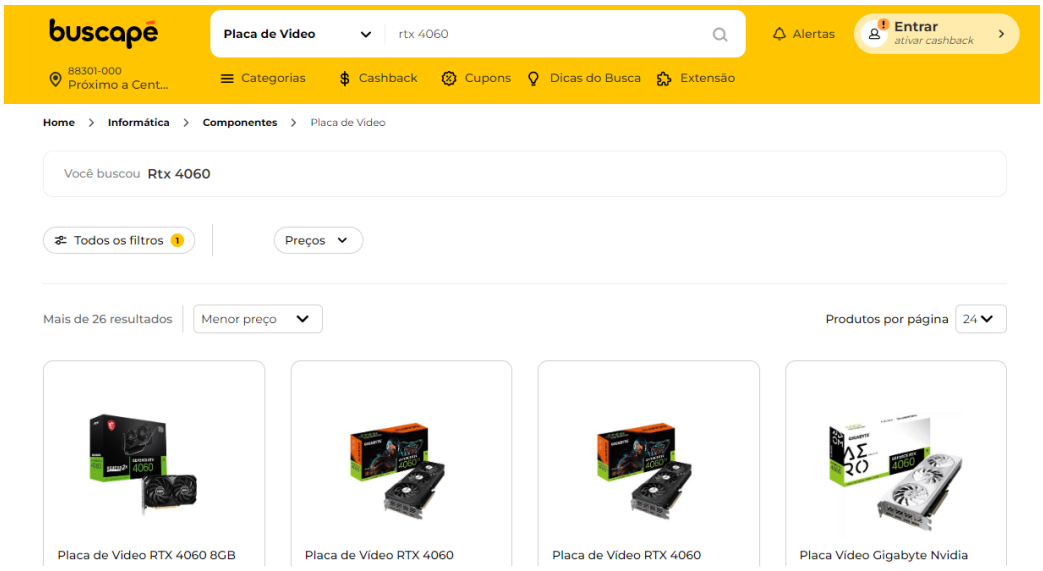
Fonte: Acervo do Autor (2023).

3.1.2 Buscapé

O Buscapé é uma ferramenta de comparação de preços que permite consumidores pesquisarem, compararem e encontrarem os melhores preços para produtos em uma variedade de categorias, como eletrônicos, moda, eletrodomésticos, entre outros. (BUSCAPÉ, 2023).

A plataforma oferece uma extensa base de dados de produtos, listando ofertas de diferentes lojas e varejistas online. Na figura 3, é apresentada a página de pesquisa da plataforma Buscapé, com os resultados da busca por uma placa de vídeo (Figura 1).

Figura 3 - Buscapé (Página - Pesquisa)



Fonte: Acervo do Autor (2023).

4. BUSCAPEÇA - PLATAFORMA WEB DE BUSCA E COMPARAÇÃO DE PREÇOS DE HARDWARE

Este capítulo visa abordar detalhes do desenvolvimento do protótipo, demostrando os passos para a criação, análise e implementação da ferramenta.

4.1 ANÁLISE

A engenharia de software desempenha um papel central durante o desenvolvimento de uma aplicação, facilitando o planejamento e implementação, além de identificar possíveis pontos de melhora em um ciclo de desenvolvimento contínuo. Nos tópicos abaixo são apresentados os fluxogramas referentes ao desenvolvimento e funcionamento do protótipo.

Nesta etapa inicial do desenvolvimento, também foi realizado o levantamento de requisitos, processo responsável pela coleta e documentação das necessidades, expectativas e funcionalidades que a aplicação deve ter para atender os objetivos específicos propostos. Esta etapa envolve a identificação e a descrição detalhada de tudo o que o software deve fazer, desde recursos principais até características específicas, interfaces de usuário e requisitos não funcionais, como desempenho e segurança.

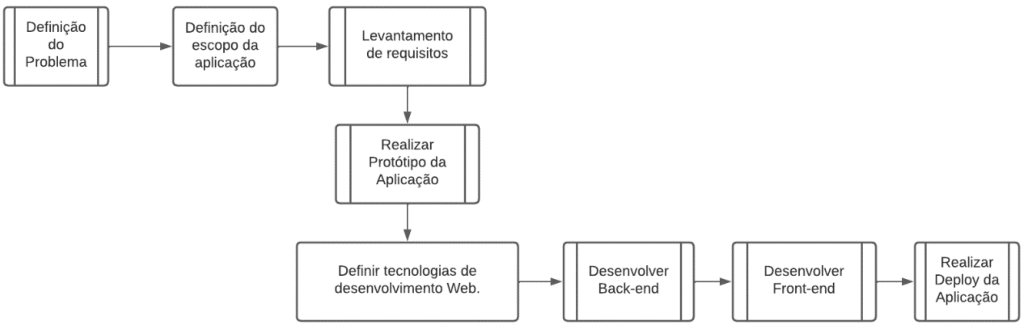
As regras de negócio foram igualmente levantadas e documentadas, especificando de forma clara quais diretrizes o protótipo deve seguir para garantir que o software atenda aos requisitos do negócio de acordo com os objetivos.

4.1.1 Fluxograma de Desenvolvimento

Conforme representado na Figura 4, para o presente trabalho de conclusão de curso em primeiro lugar fora realizada a definição do problema e do escopo da aplicação, em seguida realizado levantamento de requisitos e regras de negócio. Após a etapa de engenharia de software realizada, foi criado um protótipo visual de alta fidelidade na ferramenta Figma.

Na etapa de desenvolvimento, as tecnologias previstas foram definidas e o desenvolvimento da API foi finalizado e testado. Com o retorno dos produtos, o *front-end* da aplicação foi desenvolvido e a ferramenta foi liberada em ambiente de produção. Como forma de ilustrar os passos, na Figura 4 é apresentado um fluxograma de desenvolvimento do projeto.

Figura 4 - Fluxograma de Desenvolvimento do Projeto



Fonte: Acervo do Autor (2023).

4.1.2 Comparação do Protótipo com Estado da Arte

No Quadro 1, é demonstrado o resultado de uma análise comparativa entre as funcionalidades propostas como parte integrante deste projeto e as ferramentas destacadas no estado da arte. Essa comparação visa não apenas identificar os pontos de convergência e divergência entre as funcionalidades planejadas para este trabalho e as ferramentas já existentes, mas também avaliar a eficácia e a abrangência das soluções propostas em relação ao cenário atual de tecnologias similares no campo de busca e análise de produtos de hardware.

Quadro 1 - Comparação do Protótipo com o Estado da Arte

Funcionalidade	Buscapeça	Buscapé	Zoom
Comparação de preço em múltiplos lojistas	X	X	X
Nota de confiabilidade da plataforma	X		
Histórico e Indicação de queda de preço		X	X
Múltiplos Departamentos		X	X
Departamento de Hardware	X	X	X
Parcerias com Cashback		X	X
Foco exclusivo em indexação e comparação de preços de hardware de computador	X		

Fonte: Acervo do Autor.

4.1.3 Regras de Negócio

As regras de negócio representam um conjunto de diretrizes, restrições e critérios que delimitam o funcionamento da aplicação. Estas normas desempenham um papel importante, assegurando que o protótipo funcione de acordo com os objetivos estabelecidos para o seu

desenvolvimento. No Quadro 2, serão apresentadas as regras de negócio fundamentais para esta aplicação. Estas regras incluem orientações sobre como os dados do *scraping* são gerenciados, e quais serão as regras para apresentação dos produtos na aplicação.

Quadro 2 - Regras de Negócio

Número	Nome	Descrição
RN01	Termos de Busca	Deverá ser possível a pesquisa baseada em termos de busca, estes termos devem ser pesquisados em cada loja indexada. Objetivo final é que o termo de busca do usuário seja convertido para o formato de uma URL, passando por um processo de formatação para a remoção de caracteres especiais, espaçamentos e letras maiúsculas. Esta URL final será utilizada como termo de busca nas lojas indexadas.
RN09	Dados de *Web Scraping*	De cada loja indexada pesquisado os seguintes dados devem ser carregados e persistidos: 1 - Nome do produto 2 - Marca 3 - Preço a Vista 4 - Imagem 5 - Link de atalho (para acesso pelo usuário)
RN02	Produtos indisponíveis	Não devem ser apresentados na listagem produtos que estejam indisponíveis na loja indexada.
RN03	Preço atualizado	Os preços dos produtos devem estar atualizados conforme o preço do lojista.
RN04	Índice de Confiabilidade	Cada lojista terá uma nota própria de confiabilidade chamada IBC (Índice Buscapeça de Confiabilidade) gerada por uma fórmula, composta pela nota do comerciante na plataforma do Reclame Aqui e a quantidade total avaliações.
RN05	Informações Precisas	As retornadas devem ser precisas conforme as informações apresentadas no lojista, no momento do *scraping*.
RN06	Comparação de preços	O comparativo de preços e formato de ordenação dos produtos levará em conta o valor à vista de cada produto.
RN07	Múltiplos alvos (Lojas Indexadas)	A plataforma deverá possuir um conjunto mínimo de 3 lojas indexadas operacionais para poder exibir os resultados.

Fonte: Acervo do Autor.

4.1.4 Requisitos Funcionais

Os requisitos funcionais são declarações detalhadas que descrevem as funcionalidades e o comportamento específico que o sistema deverá ter para atender às necessidades e aos objetivos propostos. Esses requisitos definem as ações, operações e tarefas que o protótipo deve ser capaz de executar. No quadro 3 serão listados os requisitos funcionais do protótipo.

Quadro 3 - Requisitos funcionais

Número	Nome	Descrição	RN
RF01	Campo de Pesquisa de Produtos	Os usuários devem poder informar critérios de buscas de produtos.	RN01
RF02	Exibir Resultado da Busca	A plataforma deverá exibir o resultado das buscas da seguinte forma: 1 - Exibir o nome da loja 2 - Exibir o nome do produto 3 - Preço atualizado do produto (a vista)	RN02, RN03, RN04
RF03	Comparação de preços	A plataforma deve fornecer uma comparação de preços para os produtos pesquisados de diferentes lojistas	RN06
RF04	Detalhes do Produto	A plataforma deve exibir as informações de título, preço e imagem para cada produto listado.	RN05
RF05	Múltiplos Lojistas	A plataforma deve integrar dados de múltiplos lojistas, garantindo que os preços e informações de produtos estejam atualizados.	RN08
RF06	Nota do Lojista	Em cada produto deverá ser apresentada uma nota do lojista que o está ofertando (IBC - Índice Buscapeça de Confiabilidade)	RN04
RF07	Acesso ao produto	Na listagem de produtos, a aplicação deve apresentar um botão em formato de link para direcionar ao usuário à página de compra na loja.	RN09

Fonte: Acervo do Autor.

Os requisitos não funcionais são critérios que descrevem características e atributos que não estão diretamente relacionados às funcionalidades específicas da aplicação, mas que são igualmente importantes a sua qualidade, desempenho e usabilidade. Esses requisitos se concentram em como a aplicação deve realizar suas funções. No quadro 4 serão listados os requisitos não funcionais.

Quadro 4 - Requisitos não funcionais

Número	Nome	Descrição
RNF01	Banco de dados	Os dados devem ser persistidos em um banco de dados SQLite.
RNF02	Back-end	Os resultados do *Crawler* devem ser servidos através de uma API externa em Node.js.
RNF03	Usabilidade	A aplicação deve ser responsiva, com layout adaptável para dispositivos móveis e tablets.
RNF04	Layout	O padrão de identidade visual da aplicação deve ser inteiramente na cor vermelha.

Fonte: Acervo do Autor.

4.2 DIAGRAMAS

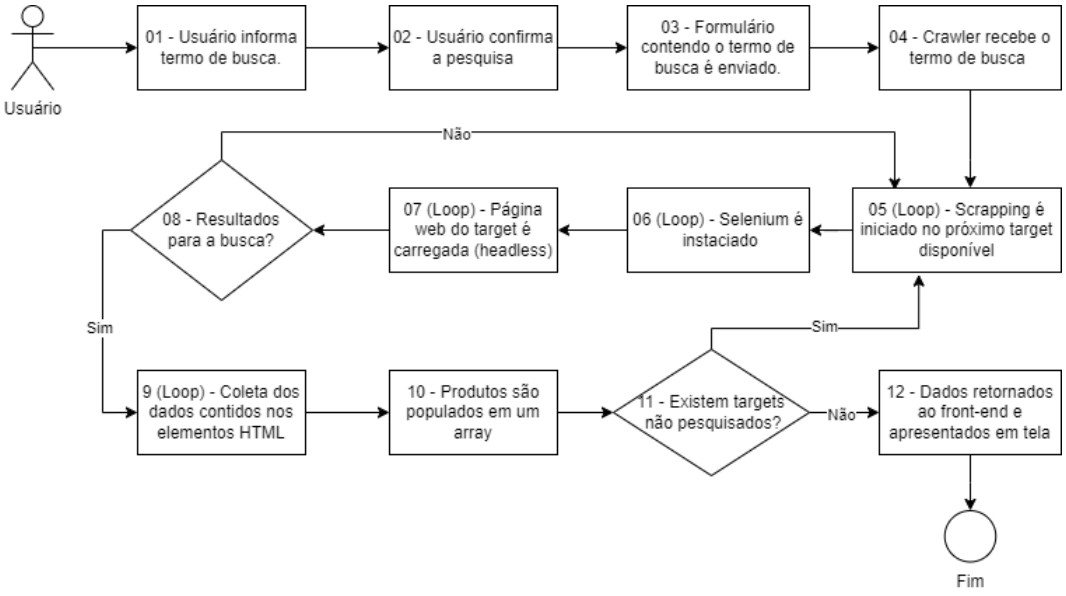
Os diagramas são usados para comunicar a estrutura e o comportamento de determinados processos do protótipo de forma compreensível. Estas representações ajudam a

identificar possíveis problemas, gargalos de desempenho e pontos de falha na arquitetura do sistema, contribuindo para uma melhor compreensão e otimização durante o processo de desenvolvimento e manutenção. Os diagramas desenvolvidos nos tópicos abaixo, são ferramentas importantes para documentar, analisar e aprimorar a arquitetura e a funcionalidade do protótipo.

4.2.1 - Fluxograma de Funcionamento do Protótipo

Trata-se de uma representação gráfica que descreve o fluxo de funcionamento ou a lógica de operação da aplicação. Seu objetivo é fornecer uma versão visual e clara de como a aplicação realiza suas funções e mostra os diferentes componentes, módulos, processos ou funções interagem entre si para alcançar um objetivo específico.

Figura 5- Fluxograma de Funcionamento



Fonte: Acervo do Autor.

4.2.1.1 Pesquisa de Produtos

Conforme representado na Figura 5, a pesquisa de produtos é iniciada por meio de um campo de pesquisa localizado na página inicial da aplicação web (Atividade 01). Este campo será responsável por receber o termo de busca que, posteriormente, será enviado ao *crawler*. Quando o usuário informa o termo de busca desejado, o formulário é enviado (Atividade 03), e o termo de busca é passado para o *back-end*, onde a ferramenta de *scraping* está operando. A

pesquisa dos produtos é realizada através do *crawler* Selenium (Atividade 05), esta ferramenta é utilizada para realizar a busca de dados na web de forma automatizada. O *crawler* recebe as configurações iniciais, como as URLs de partida de cada loja indexada, os critérios de filtragem, compostos por seletores CSS, e os parâmetros de busca, e inicia uma instância de um navegador para poder realizar o *scraping* (Atividade 07).

Para cada loja indexada, o *crawler* irá aplicar o mesmo procedimento de pesquisa, alterando apenas os parâmetros de seleção dos resultados no HTML. Após a resposta contendo o HTML da página, os critérios de filtragem são aplicados e as informações referentes aos produtos são extraídas e adicionada em um *array* (Atividade 10). Por fim, caso não haja mais lojas para a realização da busca, o resultado contido no *array* de produtos é enviado novamente ao *front-end*, mapeado e apresentado ao usuário (Atividade 12).

4.2.1.2 Avaliação de Lojistas

A avaliação dos resultados da busca na aplicação é fundamental para proporcionar aos usuários uma experiência com resultados confiáveis. A ferramenta oferece uma lista detalhada de produtos com base na pesquisa do usuário, apresentando o Índice Buscapeça de Confiabilidade (IBC) como uma métrica de destaque. Este índice é composto pela média das avaliações dos usuários, variando de 1 a 5 estrelas, juntamente com a pontuação atribuída pelos consumidores na plataforma Reclame Aqui, é uma maneira de fornecer uma visão abrangente sobre a confiabilidade de cada item listado, lavando em conta tanto as avaliações da própria aplicação, quanto a avaliação do Reclame Aqui.

Para garantir a precisão e acessibilidade desses dados, as classificações de 1 a 5 estrelas dos usuários, juntamente com a nota do lojista no Reclame Aqui, são armazenadas localmente em um banco de dados SQLite. O armazenamento SQLite é um sistema de gerenciamento de banco de dados relacional, porém, se diferencia dos sistemas tradicionais por ser um banco de dados leve, autônomo e sem necessidade de configurações ou servidores separados. Esse modelo de armazenamento foi escolhido por ser é amplamente utilizado em situações em que se requer um armazenamento local de dados básicos. Além disso, o armazenamento local das avaliações dos usuários garante a segurança e acessibilidade desses dados para futuras consultas.

Essa abordagem de avaliações não apenas permite os usuários a entenderem a qualidade e confiabilidade dos lojistas listados na plataforma, mas também pode ajudar a estimular uma tomada de decisão de compra do usuário.

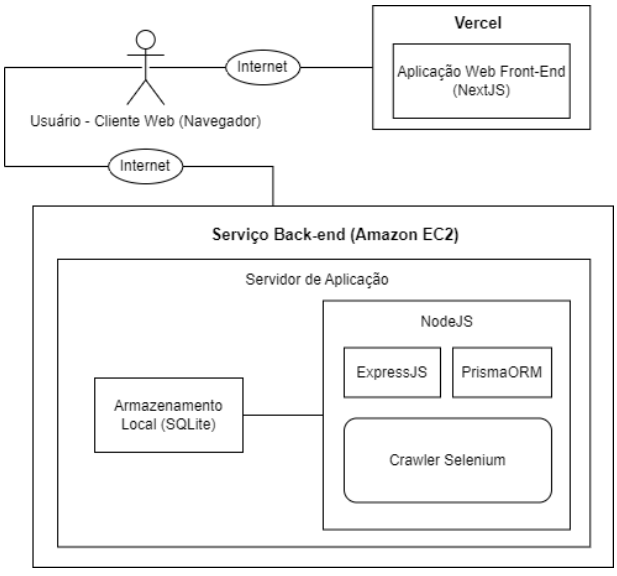
4.2.2 Disposição dos Componentes

A arquitetura do protótipo é composta por uma aplicação *front-end*, desenvolvida com o *framework* Next.js, e um serviço *back-end* executado em Node.js. O Next.js facilita a criação da interface do usuário com renderização eficiente e componentes reutilizáveis, o serviço *back-end* em Node.js gerencia a lógica, processa dados e interage com o banco de dados. Para o armazenamento, o protótipo adota o SQLite. Essa estrutura híbrida possibilita uma interação facilitada entre a interface do usuário e o manuseio de dados.

Na aplicação web foi utilizado o Next.js, que oferece todas as vantagens do React.js, ajudando na criação de interfaces dinâmicas e reativas, permite a criação de componentes reutilizáveis, evitando a repetição de código. O Next.js, adiciona ao protótipo recursos como renderização do lado do servidor (SSR) e geração estática (SSG), fazendo com que a aplicação possua um bom tempo de carregamento. No servidor de aplicação, foi utilizado o Node.js, este *framework* Javascript foi utilizado para a criação da API O ecossistema de pacotes do Node.js, disponibilizado pelo NPM, oferecendo módulos e bibliotecas prontas para uso. Umas destas bibliotecas utilizadas foi o Express.js, responsável executar a criação e gerenciar os *endpoints* da aplicação.

O componente mais importante da arquitetura é o *crawler*, este serviço está localizado no servidor de aplicação, dentro de um *endpoint* que ficará responsável por receber as informações do produto pesquisado pelo usuário (cliente web). Esta ferramenta é responsável por realizar a coleta de informações nas lojas indexadas para a pesquisa.

Figura 6 - Diagrama de Distribuição



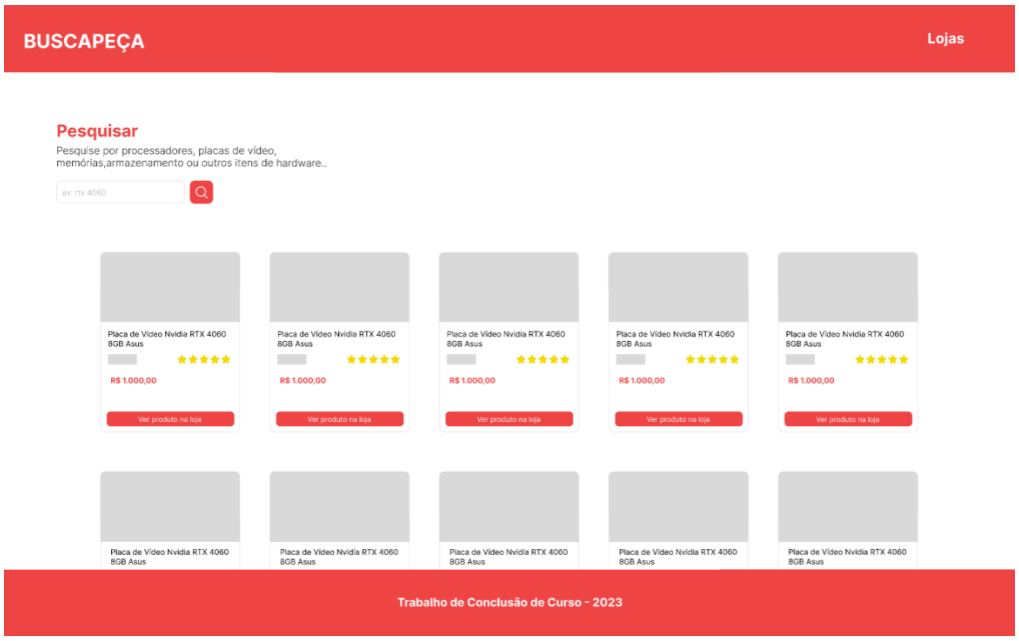
Fonte: Acervo do Autor.

Na Figura 6, está representado o diagrama de distribuição, detalhando partes da aplicação e como elas interagem para fornecer uma experiência de usuário, este diagrama permite a visualização da arquitetura do sistema, apresentando seus componentes e a forma com que eles se comunicam.

4.3 PROTOTIPAÇÃO

Para facilitar o desenvolvimento da aplicação, foi realizada a prototipação da tela principal utilizando a ferramenta Figma. No protótipo visual foi aplicada a paleta de cores do *design* previsto do projeto, assim como espaçamentos, estilo de componentes como *inputs*, botões. No protótipo visual também foi prevista distribuição dos produtos em formato de *grid*, contendo as informações relativas a cada produto, a imagem principal do produto, e um botão que servirá como redirecionamento para a página da loja.

Figura 7 - Protótipo visual realizado na ferramenta Figma



Fonte: Acervo do Autor.

O desenvolvimento prévio de um protótipo visual no Figma foi uma prática que trouxe alguns benefícios, oferecendo uma visão antecipada e tangível do produto final. O protótipo visual serviu como um guia, fornecendo diretrizes visuais e funcionais que precisam ser implementadas. Além disso, permitiu que revisões e alterações pudessem ser feitas mais prontamente, permitindo ajustes antes do desenvolvimento.

4.4 DESENVOLVIMENTO

No desenvolvimento do protótipo, foram adotados *frameworks* e tecnologias baseadas em Javascript, uma linguagem amplamente utilizada no processo de construção de aplicativos web. O Javascript está presente em cada etapa do desenvolvimento, desde o gerenciamento das páginas até a estrutura e comportamento visual, além de estar presente no *crawler*, encarregado de realizar buscas e capturar informações sobre os produtos, e também no gerenciamento dos *endpoints* da API e no envio dos dados para o armazenamento local.

Utilizando *frameworks*, como Next.js, foi possível criar uma estrutura organizada e modular, dividindo a aplicação em componentes. Além disso, o Javascript desempenhou um papel crucial no comportamento visual do protótipo. Por meio de bibliotecas como React.js, foi possível criar interfaces responsivas e interativas, proporcionando uma experiência de usuário fluida e dinâmica. Os recursos oferecidos por essas tecnologias contribuíram para a construção de uma interface amigável e atraente para os usuários.

O Javascript também foi aplicado no desenvolvimento do *crawler*, a função responsável por realizar pesquisas em diversos sites, coletando informações sobre os produtos. Além disso, todo o gerenciamento da API e do envio dos dados para o armazenamento local foi realizado utilizando tecnologias Javascript.

4.4.1 Front-End

No desenvolvimento da estrutura *front-end*, foi utilizado o Next.js, um *framework* baseado em React.js, que em conjunto com o React, desempenhou um papel importante na organização e na construção da estrutura visual da aplicação, permitindo uma abordagem mais modular e escalável por meio da divisão da interface em componentes. Neste protótipo, foram criados componentes distintos para diferentes partes da aplicação, por exemplo, o menu de navegação e para o rodapé da aplicação.

A área de pesquisa do protótipo também foi separada, este componente desempenha um papel central ao permitir que os usuários realizem consultas e chamadas para o *endpoint* responsável pela listagem dos produtos. Ao interagir com esse componente, os usuários podem buscar os produtos, desencadeando a chamada ao serviço de *web scraping* que fornecerá os resultados.

As funcionalidades de reatividade do React proporcionam um controle e manipulação dos dados na aplicação conforme ações do usuário. Ao utilizar estados, foi possível controlar o

fluxo de requisições. Quando o usuário realiza a requisição para o *back-end*, um estado é acionado para sinalizar o carregamento dos produtos está em andamento. Isso permite que a interface responda visualmente, exibindo uma tela de carregamento. Quando a API retorna a lista de produtos após a execução do *crawler* no *back-end*, os dados resultantes são salvos em outro estado separado. Este segundo estado atua como um repositório dos produtos, armazenando a lista obtida a partir da requisição. Esta lista será posteriormente utilizada para alimentar o componente de listagem na interface.

Ao mapear esse estado que contém a lista de produtos, os dados são iterados e apresentados por meio de *cards* individuais no componente de listagem. Cada *card* representa um produto específico, exibindo informações como nome, imagem, preço.

Figura 8 - Chamada para o Crawler por parte do front-end

```
async function getProducts(e: any) : Promise<void>  {
    e.preventDefault();
    setIsLoading( value: true)

    axios.get( url: `${getApiHost()}products?searchItem=${e.target.searchItem.value}`)
        .then((response: any) : void  => {
            setProductList(response.data)
        }).catch((error: any) : void  => {
                console.error("Error:", error);
        }).finally( onfinally: () : void  => {
            setIsLoading( value: false)
        })
}
```

Fonte: Acervo do Autor.

Na figura 8, é possível visualizar o trecho de código que compõe a chamada do *crawler* a partir do *front-end*. Esse processo inicia com a exibição de uma tela de carregamento, um indicativo visual para o usuário de que a chamada está em andamento e realizando o carregamento dos produtos. Para realizar a requisição ao *crawler*, foi utilizada a biblioteca Axios, uma ferramenta que facilita a execução de requisições HTTP no ambiente Javascript. Através do Axios, é especificado qual método HTTP será utilizado, assim como o *endpoint* específico para a chamada ao *back-end*, onde o *crawler* está operando.

Quando a requisição é concluída, é feito o acesso ao conteúdo da resposta fornecida pelo *back-end*. Essa resposta contém os resultados da busca realizada pelo *crawler*, ou seja, a lista de produtos obtidos através do *scraping* dos lojistas. Esses dados são cruciais para a etapa subsequente, na qual a interface é atualizada e os produtos capturados são exibidos ao usuário.

Esta etapa do processo é essencial para a funcionalidade da aplicação, permitindo que o *front-end* interaja de forma eficaz com o *back-end*, buscando e exibindo os produtos de maneira clara e ordenada na interface do usuário. Essa abordagem oferece uma experiência mais controlada ao usuário, além de fornecer acesso rápido e eficiente aos dados coletados pelo *crawler* nos diversos sites de referência de produtos de hardware.

4.4.2 Back-End

Para o desenvolvimento do *back-end* da aplicação utilizou-se o Node.js e tem como função principal o *crawler* de pesquisa, que foi feito utilizando a biblioteca Selenium. O Selenium se destaca por sua capacidade de automatizar a interação com navegadores, permitindo a navegação e a interação com os elementos das páginas da web. Isso o torna uma ferramenta ideal para realizar buscas e coletar dados em várias lojas virtuais, e garante uma eficiência na obtenção das informações dos produtos.

Além da funcionalidade do *crawler*, o *back-end* também compreende rotas na API que possibilitam a avaliação individual de cada lojista. Essas rotas permitem interações específicas, como a coleta da nota das avaliações relacionadas à confiabilidade e satisfação dos usuários com cada loja. Essas informações são cruciais para que os usuários possam tomar decisões informadas durante suas buscas. Adicionalmente, há a interação necessária com o ORM Prisma para armazenar os dados dessas avaliações no armazenamento local SQLite. O Prisma atua como uma interface para o banco de dados, simplificando e agilizando o processo de armazenamento de informações. Essa integração com o SQLite, um sistema de gerenciamento de banco de dados leve e eficiente, permite o armazenamento seguro e local das informações e avaliações dos lojistas.

4.4.2.1 EXPRESS.JS

O Express.js foi o *framework* web para Node.js escolhido para gerenciar o roteamento da aplicação, esta ferramenta simplifica o desenvolvimento de aplicativos web e APIs. A estrutura básica é relativamente simples e consiste em alguns componentes principais que trabalham juntos para lidar com solicitações HTTP. Após a instalação do Express no projeto Node.js, foi definida uma rota do tipo GET com o nome "/producs", este *endpoint* é o responsável por receber o termo de busca enviado do *front-end* e fazer a chamada da função *crawler*, que por sua vez, irá fazer o *scraping* dos dados conforme o termo de busca.

4.4.2.2 CRAWLER

Os dados necessários para selecionar os elementos HTML específicos dos lojistas são organizados e armazenados em uma constante separada. Durante o processo de *web scraping*, esses dados são mapeados, o que implica em identificar e associar os elementos HTML relevantes para a extração de informações específicas de cada lojista. Essa estrutura organizada facilita o processo de coleta de dados, permitindo que o *crawler* identifique, localize e extraia as informações de interesse de forma eficiente, proporcionando assim uma busca detalhada e precisa dos produtos disponíveis nos sites monitorados.

Na figura 9, é possível visualizar a constante que armazena as informações e seletores necessários para a extração dos dados de cada loja indexada. Esta constante é responsável por armazenar informações como o nome da loja, a URL de busca, que será utilizada para a carregar a página de pesquisa durante o processo de carregamento da loja indexada, e por fim, os seletores CSS, estes seletores serão usados durante o processo de scraping pra extrair as informações necessárias do HTML resultante de cada loja indexada.

Figura 9 - Dados de seleção para crawling das lojas indexadas

```
export const lojaData : {id: number, name: string, url... = [
    {
        id: 1,
        name: 'kabum',
        url: 'https://www.kabum.com.br/busca/',
        cardSelector: '.productCard',
        imageSelector: '.imageCard',
        nameSelector: '.nameCard',
        priceSelector: '.priceCard',
    }, {
        id: 2,
        name: 'pichau',
        url: 'https://www.pichau.com.br/search?q=',
        cardSelector: 'a[data-cy="list-product"]',
        imageSelector: '.MuiPaper-root > div > div > div > img',
        nameSelector: 'h2.MuiTypography-root',
        priceSelector: '.MuiCardContent-root > div > div:nth-child(1) > div > div:nth-child(3)',
    }, {
        id: 3,
        name: 'gkInfostore',
        url: 'https://www.gkinfostore.com.br/buscar?q=',
        cardSelector: '.listagem-item',
        imageSelector: '.imagem-produto > img:first-child',
        nameSelector: 'a.nome-produto',
        priceSelector: '.desconto-a-vista',
    }
]
```

Fonte: Acervo do Autor.

A função de *crawling* usa a instância do driver do Selenium para extrair o HTML contido na página de pesquisa de cada loja indexada. Posteriormente, a biblioteca Cheerio é empregada para realizar uma seleção de forma simplificada dos elementos contidos nesse HTML. Dessa maneira, para cada elemento *card* de produto identificado no lojista durante o momento da pesquisa, são selecionados os elementos como imagens, título, preço e link correspondentes a este produto. Esses dados são então organizados e populados em um array de produtos, posteriormente enviados de volta para o *front-end*. Na Figura 10, é possível visualizar o código correspondente a esse processo.

Figura 10 - Dados de seleção para crawling das lojas indexadas

```
await driver.getPageSource().then((html: any) : void  => {
    const $ : CheerioAPI  = load(html)

    $(selector).each( fn: (i : number , el : AnyNode ) : void  => {
        const image : string | undefined  = $(imageSelector, el).attr( name: 'src')
        const title : string  = $(titleSelector, el).text()
        const price : string  = $(priceSelector, el).text()
        const link : string  = String($( selector: '> a',el).attr( name: 'href'))

        if(price) {
            products.push({
                lojaId: idLoja, image, title, price, link
            })
        }
    })
})
```

Fonte: Acervo do Autor.

4.4.3 Deploy

Após a conclusão do desenvolvimento, a aplicação foi efetivamente disponibilizada em um ambiente de produção, garantindo que usuários de diferentes locais pudessem acessar e utilizar a versão final do protótipo. Ao realizar o *deploy* da aplicação, foi possível proporcionar uma experiência pronta para uso, permitindo que usuários finais explorem e interajam com a plataforma. Além disso, a disponibilização em ambiente de produção viabilizou a identificação de melhorias no processo de pesquisa dos produtos, e possibilitou testes do protótipo de maneira mais simplificada.

4.4.3.1 Vercel

Na escolha de hospedar o *front-end*, a Vercel se mostrou a plataforma ideal para a implantação, oferecendo uma série de funcionalidades no processo. A funcionalidade de *deploy*

automático é uma dessas características destacáveis, permitindo uma implementação contínua e imediata de alterações no código-fonte, resultando em atualizações dinâmicas da aplicação. A integração direta com o GitHub garante uma sincronização entre o repositório e o ambiente de produção. A Vercel também fornece ferramentas robustas para o monitoramento de métricas e o gerenciamento eficaz da infraestrutura da aplicação.

4.4.3.2 Amazon EC2

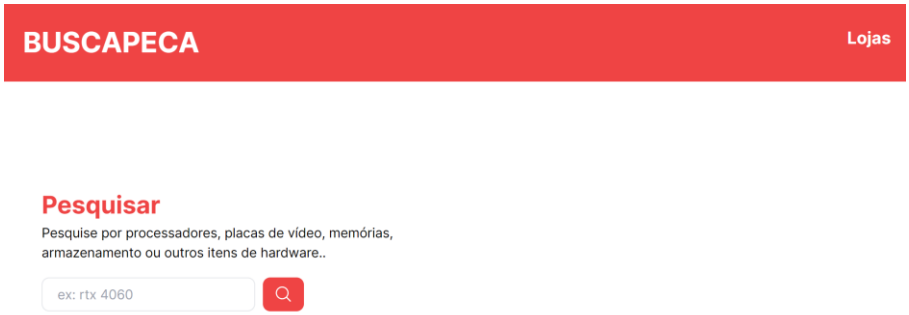
A aplicação *back-end* Node.js, que compõe tanto o *crawler* quanto a API para comunicação com o *front-end*, foi clonada e executada em uma instância no serviço EC2 da Amazon. A hospedagem da aplicação no EC2 envolveu etapas como a configuração da instância, instalação de dependências, e a exposição da aplicação uma porta para que pudesse ser acessada externamente. Por fim, a instância foi vinculada a um domínio pessoal personalizado, o que contribuiu para a identidade da aplicação e proporcionou ao *front-end* uma comunicação facilitada.

4.5 FUNCIONAMENTO

Nesta seção, serão abordados os detalhes do funcionamento da aplicação, demonstrando o fluxo de pesquisa de produtos realizado do usuário. Este detalhamento será composto por imagens e uma descrição de cada passo da utilização e funcionamento da aplicação.

O protótipo foi desenvolvido com uma funcionalidade principal, que é a pesquisa de produtos de lojas diversas com foco em hardware de computador. Quando os usuários acessam a página principal da aplicação, eles encontram um campo de pesquisa imediatamente visível e acessível. Este campo é o ponto de partida para a comunicação entre o usuário e o *back-end* da aplicação, que envia consultas aos *crawlers* para buscar informações relevantes sobre produtos de hardware e seus preços. Os termos de busca desejados são inseridos pelos usuários neste campo, permitindo-lhes pesquisar uma ampla gama de produtos e comparar preços em tempo real.

Figura 11 - Página inicial - Pesquisa de produtos (RF01)

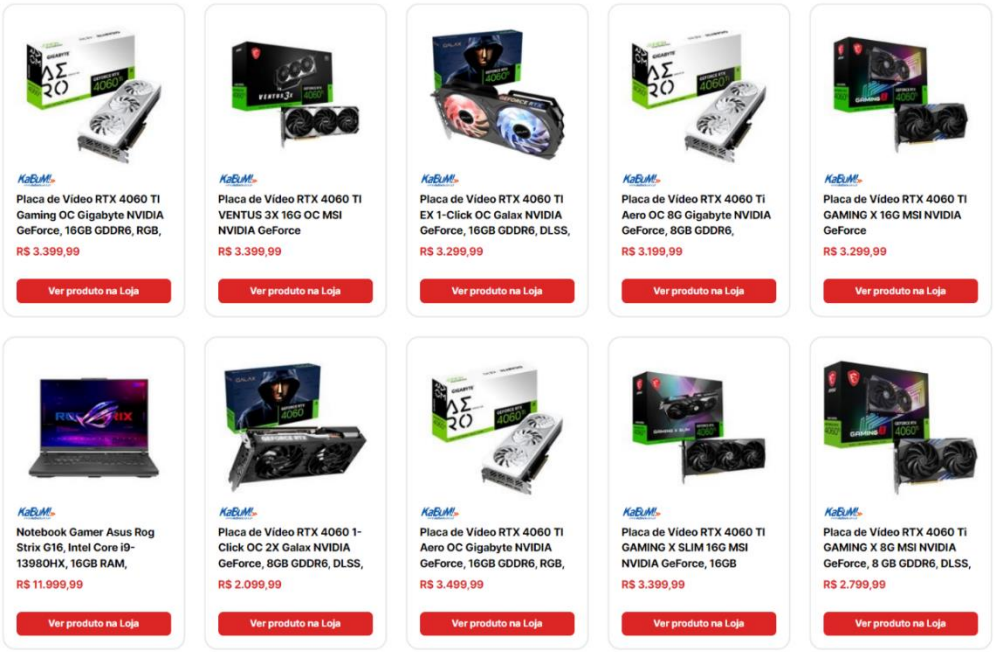


Fonte: Acervo do Autor.

Após o usuário inserir os termos de pesquisa e enviá-los ao *back-end*, a aplicação apresentará uma lista de produtos exibidos em um formato de *grid*, o que facilita a visualização e a comparação. Os resultados são classificados automaticamente em ordem crescente com base no menor preço principal disponível, tornando a comparação mais conveniente para o usuário.

Cada *card* de produto na lista inclui informações importantes. Primeiramente, há uma imagem ilustrativa do produto, que ajuda os usuários a identificarem visualmente o item que estão procurando. A logomarca da loja à qual o produto pertence é exibida, permitindo os usuários saberem de qual varejista estão considerando a compra. Além disso, a interface mostra a avaliação da loja atribuída pelos próprios usuários, fornecendo um feedback sobre a reputação da loja. Na parte inferior do *card* do produto, os detalhes essenciais são apresentados de forma clara. Primeiro, o nome do produto é destacado, permitindo a identificação do item desejado. Em seguida, o preço do produto é exibido, possibilitando a comparação direta entre diferentes ofertas. Por fim, um botão de direcionamento à página da loja é fornecido, permitindo que os usuários acessem a página da loja online para obter mais detalhes e concluir a compra. A listagem dos itens pode ser observada na figura 12.

Figura 12 - Listagem dos resultados de busca (RF02, RF03, RF04, RF06, RF07)



Fonte: Acervo do Autor.

A plataforma também inclui uma página dedicada especificamente à exibição das avaliações dos lojistas indexados, acessível através do menu superior. Essa página fornece uma visão abrangente de todos os lojistas presentes na pesquisa realizada na plataforma, apresentando informações como a classificação do Índice de Buscapeça de Confiabilidade (IBC) de cada lojista, juntamente com o total de avaliações registradas para cada um deles. Essa visualização detalhada permite aos usuários terem uma noção clara do desempenho e reputação de cada lojista presente na plataforma, oferecendo informações para a tomada de decisão sobre a escolha de um vendedor, baseando-se na confiabilidade e nas avaliações dos demais usuários. Na figura 12 é possível visualizar a apresentação da página de avaliações.

Figura 13 - Página de apresentação dos lojistas (RF05)

Fonte: Acervo do Autor.

4.6 AVALIAÇÃO DE RESULTADOS DE BUSCA

Após a conclusão das etapas envolvendo o processo de desenvolvimento da plataforma web do Buscapeça, o próximo passo foi realizar uma etapa de comparação entre os resultados oferecidos pelo protótipo e as ferramentas disponíveis consideradas como estado da arte. Tanto o Buscapé quanto o Zoom são ferramentas relevantes no mercado e reconhecidas por sua capacidade de comparação. No Quadro 5, foram registradas as análises comparativas entre os menores preços oferecidos por cada plataforma com base em um determinado termo de pesquisa.

Quadro 5 - Comparação de resultados de pesquisa com estado da arte

Termo de Busca	**Buscapeça**	**Zoom**	**Buscapé**
SSD Kingston NV2 1TB	R$ 279,00	R$ 301,93	R$ 301,93
RTX 4060	R$ 1.898,00	R$ 2.209,15	R$ 2.209,15
GTX 1660	R$ 1.129,00	R$ 1.328,87	R$ 1.328,87
CORE i5 12400F	R$ 928,99	R$ 928,99	R$ 928,99
AMD Ryzen 5600X	R$ 1.093,41	R$ 1.168,22	R$ 1.168,22

Fonte: Acervo do Autor.

Essa etapa de foi importante para avaliar o desempenho e a eficácia do protótipo recém-desenvolvido em relação aos padrões estabelecidos pelas ferramentas já consolidadas no mercado. As análises contemplaram não apenas a precisão na comparação de preços e extensão dos catálogos, mas também demonstraram uma vantagem nos preços finais oferecidos pelo Buscapeça em comparação com as plataformas já existentes.

5. CONSIDERAÇÕES FINAIS

Este trabalho constitui-se no desenvolvimento de um protótipo que tem como objetivo realizar a busca e comparação de preços de produtos com foco em hardware de computador, permitindo que usuários pesquisem por um produto e a aplicação web faça a busca dos produtos utilizando uma técnica de levantamento de dados chamada de *scraping*.

Com base no levantamento bibliográfico fora realizada a seleção dos recursos tecnológicos adotados no desenvolvimento do protótipo, incluindo bibliotecas, *frameworks* e todo o ecossistema Javascript, que desempenharam um papel fundamental e facilitaram todas as etapas do processo de criação. Essas tecnologias foram essenciais para a conclusão bem-sucedida do protótipo.

O levantamento de regras de negócio e consequente especificação dos requisitos funcionais e não funcionais, fora realizado e especificado no terceiro tópico do trabalho o que permitiu materializar todo o escopo necessário.

Durante a construção do *front-end* do protótipo, a combinação do *framework* Next.js com a biblioteca React.js desempenhou um papel fundamental na criação de uma interface altamente estruturada e reutilizável. Para a estilização, foi adotado o Tailwind CSS, um conjunto de classes utilitárias que agiliza a criação de estilos, já no *back-end*, o Node.js e as bibliotecas Express.js e Prisma ofereceram agilidade para o processo de desenvolvimento, permitindo a implementação das funcionalidades e rotas da aplicação. Além disso, o formato de armazenamento de dados local SQLite, se mostrou funcional para o caso de uso da aplicação.

A função central do protótipo consiste na execução do processo de coleta de dados (*scraping*), realizada por meio de uma técnica denominada *crawler*. Esse recurso desempenha o papel de receber e processar informações relacionadas às lojas, termos de busca e seletores, os quais determinam quais partes do conteúdo HTML devem ser extraídas de cada loja. Uma vez que esses dados estão disponíveis, o *crawler* inicia instâncias de navegadores em cada loja indexada, realizando a extração dos dados relevantes. Durante o desenvolvimento dessa etapa, a biblioteca Selenium foi empregada para gerenciar e coordenar integralmente o funcionamento do *crawler*, permitindo a automação e interação com os navegadores, viabilizando a extração precisa e organizada das informações desejadas.

Ao término do processo de desenvolvimento da plataforma Buscapeça e sua subsequente comparação com as ferramentas consideradas como referência no mercado, é possível destacar a relevância e o potencial desta ferramenta de comparação de preços de hardware. A análise comparativa evidenciou não apenas a capacidade do protótipo em

equiparar-se a plataformas consolidadas, como Buscapé e Zoom, mas também sua vantagem competitiva em relação aos menores preços encontrados em determinadas buscas. A etapa de comparação não se limitou à precisão na comparação de preços, mas também destacou a extensão do catálogo de produtos e, principalmente, o impacto nos preços finais oferecidos pelo Buscapeça.

Uma vez concluído o desenvolvido, a aplicação foi disponibilizada em ambiente de produção, permitindo que qualquer usuário possa acessar e utilizar a versão final do protótipo de forma facilitada. Para isso, a plataforma escolhida para hospedar o *front-end* foi a Vercel, que conta com diversas funções para facilitar o processo de implantação, como *deploy* automático e integração direta com o GitHub. O serviço *back-end*, foi incluído em uma máquina no serviço EC2 da Amazon e disponibilizado em um domínio personalizado.

Esses resultados reforçam a viabilidade e o potencial da plataforma como uma alternativa competitiva no mercado, garantindo aos usuários uma opção confiável, abrangente e com preços atrativos na busca por produtos de hardware. A análise comparativa não só validou a eficácia do protótipo, mas também ressaltou sua capacidade de fornecer um diferencial significativo no cenário atual de comparação de preços, promovendo uma escolha informada e vantajosa para os consumidores no setor de hardware.

5.1 – RECOMENDAÇÕES DE TRABALHOS FUTUROS

Como recomendação para a continuidade e desenvolvimento do protótipo, fica a implementação de um processo de armazenamento dos itens pesquisados pelos usuários, a persistência destes dados poderia agilizar o processo de pesquisa, que atualmente se torna muito custoso, visto que os navegadores são instanciados e processo de *scraping* acontece por completo no momento da busca do usuário. Esta reestruturação também iria permitir que o usuário realizasse não só a avaliação da loja, mas também avaliasse os produtos resultantes do processo de busca que estão armazenados.

Para que a plataforma fique mais completa e a pesquisa se torne mais precisa, seria viável a implementação de um processo de busca por categorias dentro de cada loja indexada no processo de pesquisa. Desta forma, seria possível realizar a comparação de múltiplos produtos de uma mesma categoria, ou até pesquisar por produtos correspondentes apenas a uma determinada categoria.

No estado atual do protótipo, também existe margem para melhora na forma de inclusão de uma nova loja no processo de pesquisa de produtos. No momento, a inclusão deve ser feita

manualmente no código fonte, visto que as lojas indexadas estão definidas em uma constante de forma fixa no código, conforme representado na Figura 9. Este processo poderia ser feito de forma facilitada em uma tela de administração dentro da plataforma, onde um usuário administrador informa as credenciais, acessa o formulário de cadastro de lojas e realiza a inclusão de uma nova loja, juntamente com todas as informações e seletores CSS necessários para pesquisa dos produtos.

REFERÊNCIAS

ABBA, Ihechikara Vincent. 2022. Disponível em: <https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/>. Acesso em: 01 dezembro 2023.

ADRIANO, Thiago. **Guia prático de TypeScript: Melhore suas aplicações JavaScript**. Casa do Código, 2021 *Ebook.*

ANDRADE, Paula. 2021. Disponível em: < https://www.treinaweb.com.br/blog/o-que-e-o-express-js>. Acesso em: 03 julho 2023.

BUSCAPE. 2023. Disponível em: < https://www.buscape.com.br/conta/privacidade>. Acesso em: 03 outubro de 2023

BUZZI, Felipe. 2022. Disponível em: < https://blog.rocketseat.com.br/prisma-uma-das-melhores-coisa-que-ja-aconteceu-no-ecossistema/>. Acesso em: 29 junho 2023.

CLOUDFLARE. 2023. Disponível em: <https://www.cloudflare.com/pt-br/learning/bots/what-is-a-web-crawler/>. Acesso em: 21 outubro de 2023

DUCKETT, Jon. **Javascript & Jquery Interactive Front-End Web Development**. Indianopolis, John Wiley & Sons, 2014.

FLANAGAN, David. **JavaScript: o guia definitivo**. Porto Alegre: Bookman, 2014 *Ebook*.

FONSECA, Elton. 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-orm>. Acesso em: 28 junho 2023.

KOWALCZYK, Krzystof. **Essential React**. Essential Programming Books, 2015.

MBATHI, Collins. 2023. Disponível em: <https://www.turing.com/kb/javascript-for-backend-development>. Acesso em: 28 junho 2023.

MEIRELLES, Fernando S. **Pesquisa do Uso da TI – Tecnologia de Informação nas Empresas**. FGVcia, 2023 *Ebook*.

MITCHELL, Ryan. **Web Scraping with Python Collecting Data From The Mordern Web**. O'Reilly Media Inc, 2015 *Ebook.*

NASCIMENTO, Felipe. 2022. **NextJS: por que usar?**. Disponível em: < https://www.alura.com .br/artigos/next-js-vantagens>. Acesso em: 22 outubro de 2023.

NEXTJS. 2023. Disponível em: <https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs/>. Acesso em: 22 outubro de 2023

REACT. 2023. Disponível em: < https://react.dev/learn>. Acesso em: 30 junho 2023

ROVEDA, Ugo. 2023. Disponível em: < https://kenzie.com.br/blog/react/>. Acesso em: 03 julho 2023.

RUNGHTA, Krishna. **Learn NodeJS in 1 Day: Complete Node JS Guide with Examples**. Ergo ebooks, 2016 *Ebook*.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo, Pearson Education, 2011 *Ebook*.
SOUTO, Mario. 2022. Disponível em: < https://www.alura.com.br/artigos/react-componentes-com-styled-components>. Acesso em: 03 julho 2023.

TAILWIND. 2023. Disponível em: < https://tailwindcss.com>. Acesso em: 21 outubro de 2023

VILLANIN, Mateus. 2023. Disponível em: <https://www.alura.com.br/artigos/figma>. Acesso em: 30 junho 2023.

ZOOM. 2023. Disponível em: < https://www.zoom.com.br/conheca-o-zoom>. Acesso em: 03 outubro de 2023