UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GIUSEPE BUDNY

APLICAÇÃO *WEB* PARA REDUÇÃO DE CUSTOS NO PROCESSO DE COMPRAS NOS SUPERMERCADOS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2020

GIUSEPE BUDNY

APLICAÇÃO *WEB* PARA REDUÇÃO DE CUSTOS NO PROCESSO DE COMPRAS NOS SUPERMERCADOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Richard Duarte Ribeiro

PONTA GROSSA

2020

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

APLICAÇÃO WEB PARA REDUÇÃO DE CUSTOS NO PROCESSO DE COMPRAS NOS SUPERMERCADOS

por

GIUSEPE BUDNY

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 29 de Outubro de 2020 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Richard Duarte Ribeiro
Orientador

Prof. Dr. Augusto Foronda
Membro titular

Prof. MSc. Vinícius Camargo Andrade
Membro titular

Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

Prof. Dr. André Pinz Borges
Coordenador do curso

- O Termo de Aprovação assinado encontra-se arquivado na Secretaria Acadêmica -

AGRADECIMENTOS

Gostaria de agradecer ao Prof. Dr. Richard Duarte Ribeiro pelo apoio durante as matérias que tive com ele e também durante o desenvolvimento deste trabalho.

Agradeço também a minha família pelo apoio que sempre me deram para continuar a trilhar meu caminho de forma íntegra e respeitosa para com todos que estivessem nele.

Agradeço aos meus amigos de infância que apesar de cada um ter seguido um caminho diferente, os mesmos ainda se fazem presentes nas oportunidades que a vida nos dá.

Agradeço aos colegas de trabalho por dividirem conhecimento que também contribuiu para o enriquecimento deste projeto.

Enfim, agradeço a todos os que por algum motivo contribuíram para o meu crescimento e para o desenvolvimento do presente trabalho.

RESUMO

BUDNY, Giusepe. **Aplicação *web* para redução de custos no processo de compras nos supermercados**. 2020. 55 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2020.

A educação financeira tem sido um tema de discussão para a realidade da população brasileira e está começando a ser questionada a necessidade desta na educação básica. Dentro deste tema existem algumas questões vinculadas à gestão dos recursos financeiros relacionadas ao dia a dia da população como no processo de compras em supermercados. Com a crescente utilização de ferramentas tecnológicas e inovadoras, é possível identificar um nicho para implementação de um sistema que permita desenvolver noções de pesquisa de preços para identificar pontos possíveis para redução de custos ao realizar compras em supermercados. Este trabalho apresenta o desenvolvimento de uma aplicação *web* que permita a comparação de preços em todo o processo de compras, desde o cálculo de distância até o estabelecimento, bem como a comparação dos preços dos produtos disponíveis nos supermercados.

Palavras-chave: Aplicações *Web*. Compras. Educação Financeira.

ABSTRACT

BUDNY, Giusepe. **Web application to reduce costs in the supermarket shopping process**. 2020. 55 p. Work of Conclusion Course (Graduation in Technology in Systems Analysis and Development) - Federal Technology University - Paraná. Ponta Grossa, 2020.

Financial education has been a topic of discussion for the reality of the Brazilian population and its need for basic education is beginning to be questioned. Within this theme there are some issues related to the management of financial resources related to the daily life of the population, such as the supermarket shopping process. With the increasing use of technological and innovative tools, it is possible to identify a niche for the implementation of a system that allows the development of price research notions to identify possible points for cost reduction when shopping in supermarkets. This work proposes the development of a web application that allows the price comparison in the whole shopping process, from the distance calculation to the establishment and the comparison of the prices of the products available in the supermarkets.

Keywords: Web Applications. Shopping. Financial (Financial literacy).

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

LISTA DE SIGLAS

BNCC	*Base Nacional Comum Curricular*
OCDE	*Organização de Cooperação e de Desenvolvimento Econômico*
SQL	*Structured Query Language*
UI	*User Interface*
HTML	*Hypertext Markup Language*
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
URL	*Uniform Resource Locator*
SPA	*Single Page Application*
HTTP	*Hypertext Transfer Protocol*
CPU	*Central Process Unit*

LISTA DE ACRÔNIMOS

CONEF	*Comitê Nacional da Educação Financeira*
ENEF	*Estratégia Nacional de Educação Financeira*
ANSI	*American National Standard Institute*
JSON	*JavaScript Object Notation*
ACID	*Atomicidade, Consistência, Isolamento e Durabilidade*
REST	*Representational State Transfer*
CORS	*Cross-Origin Resource Sharing*

SUMÁRIO

1 INTRODUÇÃO

Em 2015, uma pesquisa realizada pela Proteste – Associação Brasileira de Defesa do Consumidor apontou que o preço de um mesmo produto pode ter uma discrepância de 168% em diferentes supermercados (PROTESTE, 2015). Em 2017, em pesquisa realizada pelo Procon, na cidade de Maringá no Paraná, mostrou que o mesmo produto pode possuir uma variação de 299% em seu preço (PROCON, 2017).

Em 2019, uma pesquisa realizada pela CNDL/SPC Brasil e Banco Central mostrou que os hábitos de consumo dos brasileiros estão mudando. Um destes hábitos é a consulta de preços antes da compra dos produtos e representa 59% da população brasileira (SPC BRASIL E CNDL, 2019).

Com estes dados é possível identificar uma oportunidade no mercado para a implementação de um aplicativo *web*, em que o mesmo permitirá calcular os gastos e economias relacionados ao processo de comprar produtos em um supermercado. Acredita-se que por meio da comparação dos preços e da disponibilidade dos mesmos, os mercados utilizarão este meio como divulgação e promoção do próprio estabelecimento, divulgando promoções e melhores preços para os consumidores.

A aplicação proposta neste trabalho tem como objetivo facilitar o acesso as informações dos produtos, assim como seus preços e a sua disponibilidade no estabelecimento. Além de calcular qual o custo de realizar as compras incluindo a distância de deslocamento até o supermercado em questão.

Neste contexto, este trabalho propôs o desenvolvimento da aplicação *web* citada em formato de um protótipo e o levantamento de informações referentes a educação financeira. O trabalho também apresenta informações referentes as tecnologias utilizadas para o desenvolvimento da aplicação.

1.1 ESCOPO E MOTIVAÇÃO

A motivação que levou ao desenvolvimento deste trabalho surge a partir do momento em que a sociedade apresenta um aumento no interesse acerca do tema educação financeira, desta forma, buscando novos meios de economizar (SAVOIA; SAITO; SANTANA, 2007).

Em 2018, mais de 1,3 mil iniciativas de educação financeira foram cadastradas no 2º Mapeamento Nacional das Iniciativas de Educação Financeira, realizado pelo Comitê Nacional da Educação Financeira (Conef) (BRASIL/ENEF, 2018).

Em 2014 ocorreu a homologação das diretrizes da Base Nacional Comum Curricular (BNCC). Esta, por sua vez, torna obrigatório, a partir de dezembro de 2019, que as instituições de ensino brasileiras estejam adaptadas para o estudo de conceitos básicos de economia e finanças, visando à educação financeira dos alunos (MINISTÉRIO DA EDUCAÇÃO, 2019).

Com estes dados, é possível identificar uma necessidade por parte da população brasileira, em relação a aprimorar seus conhecimentos referentes a educação financeira e, desta forma, fomentar a busca por soluções que visam reduzir os gastos da população.

Como resultado do presente trabalho foi possível criar um aplicativo *web* que permite que os clientes dos supermercados possam identificar produtos e ofertas que sejam mais interessantes para o seu orçamento.

1.2 OBJETIVOS

O Objetivo Geral do presente trabalho foi desenvolver uma aplicação *web* para ser utilizada pelos clientes dos supermercados para comparar preços e identificar os supermercados com produtos mais acessíveis.

Como objetivos específicos foram descritos a seguir:

- Estudar as tecnologias e ferramentas utilizadas para o desenvolvimento do projeto;
- Elencar as funcionalidades para o sistema;
- Fundamentar a geração do relatório.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho apresenta no primeiro capítulo uma introdução sobre o tema abordado, as motivações para o desenvolvimento do mesmo e os objetivos gerais e específicos.

O segundo capítulo apresenta o referencial teórico necessário para compreensão dos conteúdos abordados neste trabalho com os tópicos referentes à educação financeira, o uso de tecnologias disponíveis para fomentar a educação financeira, e também as tecnologias utilizadas para o desenvolvimento do *website*, como linguagens de programação, bancos de dados e *frameworks*.

O terceiro capítulo descreve a metodologia abordada para o desenvolvimento do presente trabalho.

O quarto capítulo apresenta o desenvolvimento da aplicação, incluindo as etapas necessárias para alcançar os objetivos propostos inicialmente.

O quinto capítulo aborda os resultados obtidos além de apresentar as considerações finais, as quais discutem a possibilidade de implementar novas rotinas e processos dentro da mesma aplicação com intuito de dar sequência no desenvolvimento do projeto e aprimorar o produto final.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos necessários para embasar o projeto desenvolvido, incluindo as ferramentas necessárias e os motivos das escolhas das mesmas.

Os assuntos abordados neste capítulo são: educação financeira; ferramentas e meios para incentivo da educação financeira; as tecnologias utilizadas para desenvolvimento do lado do servidor e do lado cliente; ferramentas para armazenamento de dados; *frameworks* utilizados no desenvolvimento do aplicativo *web*; e por último, os motivos da escolha das ferramentas abordadas.

2.1 EDUCAÇÃO FINANCEIRA

A educação financeira refere-se ao conjunto de conhecimentos que fundamenta a tomada de decisões, garantindo segurança no gerenciamento das finanças pessoais, segundo Savoia, Saito e Santana (2007).

Para Jacob et al. (2000), o termo “financeira” refere-se a todas as atividades rotineiras que envolvem o dinheiro, tanto para a gestão dos gastos em seu orçamento mensal, quanto para um empréstimo no banco, o uso do cartão de crédito, bem como para investimentos. Por outro lado, o termo “educação” comporta o conhecimento de conceitos, práticas, leis e normas sociais inerentes as tarefas financeiras básicas.

Segundo a Organização de Cooperação e de Desenvolvimento Econômico – OCDE (2005), define-se educação financeira como:

> O processo pelo qual consumidores e investidores melhoram seu entendimento sobre os conceitos e os produtos financeiros e, através da informação, instrução e/ou conselhos objetivos, desenvolvam as habilidades e a confiança para conhecer melhor os ricos e a oportunidades financeiras, e assim tomarem decisões fundamentadas que contribuem para melhorar seu bem-estar financeiro (OCDE, 2005, p.13).

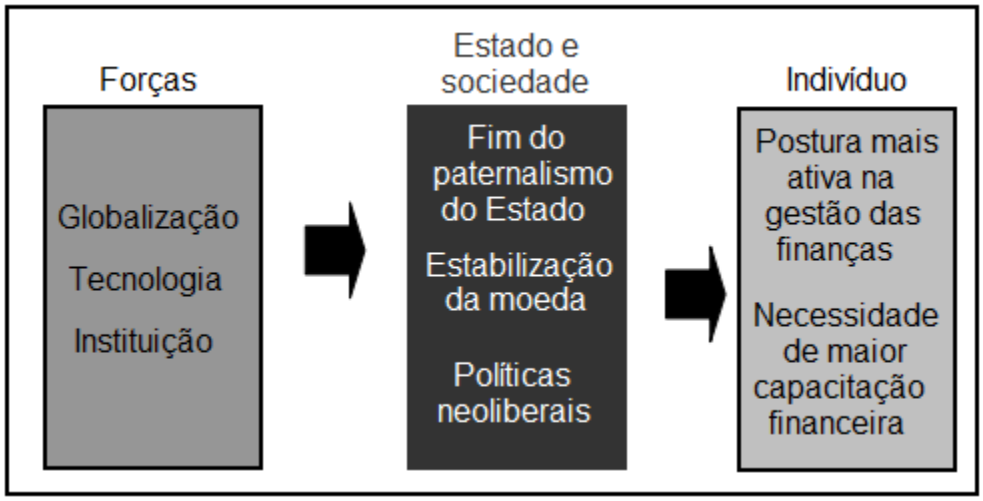
Estes conceitos tornaram-se presentes na vida do indivíduo da sociedade contemporânea devido à necessidade de acompanhar o desenvolvimento e a sua inserção no ambiente em que vive e nas relações que ele mantém. Esta evolução é

decorrente das mudanças tecnológicas, econômicas e também regulatórias que tornaram a complexidade dos serviços financeiros elevada (SAVOIA; SAITO; SANTANA, 2007).

Para Braunstein e Welch (2002), quando os integrantes do meio econômico se tornam mais capacitados, o mercado acompanha o desenvolvimento e acaba realçando um ambiente mais competitivo e eficiente. Desta forma, consumidores se tornam mais exigentes, direcionando suas necessidades e obtendo um retorno condizente com elas.

Segundo Savoia, Saito e Santana (2007), é possível identificar a origem desta crescente demanda do indivíduo de dominar habilidades e conhecimentos financeiros que surgiu devido a três forças, conforme a Figura 1.

Figura 1 - Forças propulsoras da Educação Financeira



Fonte: Savoia et al. (2007)

É possível identificar na Figura 1 que o conceito de educação financeira teve grande impacto após o surgimento da globalização, das tecnologias e da instituição. Levando em consideração que as políticas neoliberais ganharam força no Brasil em 1985, segundo Gros (2002), e com a influência da globalização e as inovações tecnológicas que geraram uma reorientação por parte do governo, estima-se que o interesse em educação financeira surgiu em meados de 1990 (SAVOIA; SAITO; SANTANA, 2007).

Com a crescente necessidade de desenvolvimento acerca do tema educação financeira, em 2005 a OCDE divulgou princípios e recomendações para

este tema e que foram compilados por Savoia, Saito e Santana (2007) descritos como:

- A educação financeira não deve levar em consideração interesses particulares;
- Os programas de educação financeira devem levar em consideração a necessidade regional.
- A educação financeira deve ser utilizada como meio para o crescimento e desenvolvimento econômico da região.
- Durante o processo de educação financeira, O envolvimento das instituições financeiras no processo deve ser estimulado.
- A educação financeira deve ser um processo adaptativo conforme a evolução dos mercados.
- A mídia deve ser utilizada como ferramenta para o incentivo da educação financeira.
- A educação financeira deve começar na escola.
- As instituições financeiras devem garantir que os clientes tenham conhecimento dos serviços financeiros adquiridos.
- O foco dos programas de educação financeira deve ser o planejamento financeiro pessoal.
- Os programas devem ser personalizados conforme as necessidades dos grupos especificamente.

Como resultado da preocupação acerca desta temática, em dezembro de 2010, a presidência da república divulgou um decreto em que instituiu a Estratégia Nacional da Educação Financeira (ENEF), com o intuito de promover a educação financeira (BRASIL, 2011).

A OCDE, por meio do *Brazil Economic Snapshot* (2019), apresentou as prioridades da reforma de 2019. Com estas prioridades foi possível garantir mais flexibilidade para as escolas para adaptarem-se às necessidades dos alunos.

Desta forma, nota-se a crescente preocupação com a educação financeira dentro do território brasileiro.

2.2 MEIOS E FERRAMENTAS PARA INCENTIVO A EDUCAÇÃO FINANCEIRA

Acredita-se que para divulgar a educação financeira são necessários meios e ferramentas que estejam compatíveis com o período que vive a sociedade e as necessidades da mesma.

Como incentivo a educação financeira por parte dos órgãos públicos, o ENEF apresentou os seguintes programas:

- Programa de educação financeira para adultos;
- Programa de educação financeira para mulheres beneficiárias do Programa Bolsa Família (PBF);
- Programa de educação financeira para aposentados;

Além destes programas, estão disponíveis guias de recomendações elaborados pelo próprio ENEF, no qual são direcionados para os setores financeiro, público e também para os grupos integrantes dos programas de educação financeira (ENEF, 2017).

Levando esta análise para o presente trabalho, o projeto desenvolvido procura auxiliar o indivíduo que possui interesse em aplicar os conhecimentos adquiridos em educação financeira, como por exemplo, nas questões de redução de gastos com compras.

2.3 FERRAMENTAS PARA DESENVOLVIMENTO DA APLICAÇÃO *WEB*

Nesta seção serão apresentadas as tecnologias para o desenvolvimento do servidor da aplicação, os *frameworks* utilizados para desenvolver a parte do cliente, a ferramenta utilizada para armazenamento de dados, além dos conceitos inerentes à esta aplicação.

O conjunto das principais tecnologias utilizadas neste projeto compõe um agrupamento denominado MEVN *stack,* sendo as tecnologias o MongoDB, o Express, o Vue.js e o NodeJS.

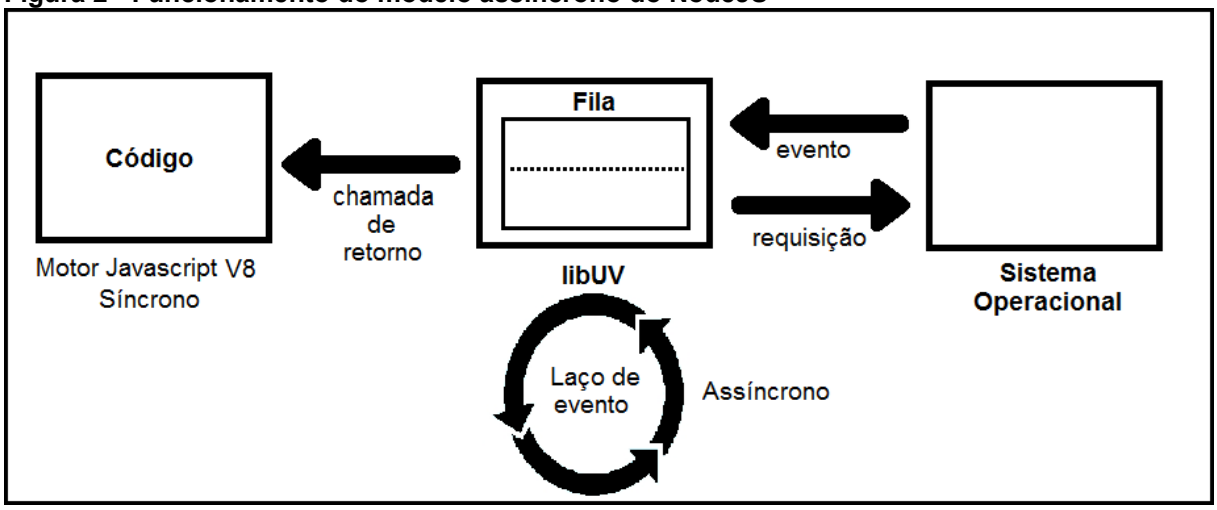
2.3.1 NodeJS (versão 12.7.0)

NodeJS, criado por Ryan Dahl em 2009, é uma plataforma que combina um motor JavaScript V8[1] da Google com o modelo orientado a eventos. Ele também possui entrada e saída de dados de baixo nível. Isso significa que este modelo depende de troca de informações simples e enxutas, o que garante grande desempenho (TEIXEIRA, 2012). Esta entrada e saída de dados de baixo nível fica a cargo da biblioteca libUV (LIBUV, 2020).

Esta plataforma permite utilizar a linguagem JavaScript no servidor utilizando o modelo assíncrono. Esse modelo prevê a execução de funções de forma independente entre si, desta forma garantindo escalabilidade. O resultado é a possibilidade de expandir o sistema garantindo alta performance para executar funções de forma rápida e eficaz (NODEJS, 2019).

Na Figura 2 está ilustrado o processo onde o sistema operacional encaminha para uma fila de controle de eventos gerenciado pelo libUV onde este encaminhará para o Motor Javascript V8 e realizará as requisições.

Figura 2 - Funcionamento do modelo assíncrono do NodeJS



Fonte: Adaptado de Singh (2018)

Para reduzir o tempo de desenvolvimento deste projeto, foi utilizado o npm, um gerenciador de pacotes disponível para o NodeJS. Com ele é possível inserir em

[1] O motor JavaScript V8 é um interpretador de JavaScript desenvolvido pela Google. Ele é chamado de motor porque recebe código JavaScript e o executa.

um projeto bibliotecas já desenvolvidas por outros desenvolvedores, o que simplifica o processo de desenvolvimento de um *software* (NPM, 2020).

Utilizando o npm foi possível adicionar os seguintes módulos (NPM, 2020) no presente projeto:

- Body-parser 1.19.0: permite manipular a estrutura do corpo de uma requisição;
- CORS 2.8.5: permite habilitar as configurações do CORS (*Cross-Origin Resource Sharing*) no servidor. Esta é uma tecnologia utilizada nos navegadores para permitir a troca de informações entre domínios distintos;
- Dotenv 8.2.0: permite o uso de variáveis de ambiente que podem ser carregadas em um arquivo separado;
- Jsonwebtoken 8.5.1: permite o uso de *tokens* para realizar autenticações entre cliente e servidor;
- Nodemon 1.19.3: simplifica o processo de desenvolvimento de aplicações em NodeJS, permitindo que a aplicação recarregue toda vez que surgir uma alteração em seu código fonte;
- Express 4.17.1: *framework* para o NodeJS que reduz a quantidade de códigos escritos e torna a compreensão das estruturas dos códigos mais simples;
- Mongoose 5.7.5: permite a conexão entre a aplicação NodeJS e o banco de dados MongoDB.
- Axios 0.19.2: biblioteca para o lado do cliente que permite realizar requisições *Hypertext Trasnfer Protocol* (HTTP) e receber as respostas para estas requisições.

Acredita-se que o sucesso de uma plataforma pode ser medido pelo número de seus *downloads*, já que isso potencialmente mostra a adoção do *software* pelo público.

No caso do NodeJS, em 2018 ele alcançou a marca de 35 milhões de *downloads* por mês, levando a um total de mais de 365 milhões em um ano. Esses números mostram o aumento na quantidade de pessoas participando da comunidade de usuários do NodeJS (NODESOURCE, 2018).

2.3.2 Banco de dados

Esta seção aborda os principais conceitos referentes a Banco de Dados relacionais e não relacionais. Também apresenta o modelo orientado a documentos, seguido pela descrição do *software* MongoDB, utilizado para o desenvolvimento deste trabalho.

2.3.2.1 Modelo Relacional

O Banco de Dados relacional é representado por um conjunto de tabelas que relacionam-se entre si. Estas tabelas são utilizadas para armazenar as informações que representam um objeto. Cada tabela é composta por linhas e colunas, na qual as colunas representam os tipos de dados armazenados e as linhas compõe o conjunto de atributos de um objeto. Cada linha da tabela pode ser identificada por um atributo único simples ou composto, chamado chave primária, que garante a identidade de um objeto (AMAZON RDS, 2019).

A interface primária de interação com os dados é a *Structured Query Language* (SQL). O SQL foi reconhecido pelo American National Standard Institute (ANSI) como padrão em 1986. Esta linguagem é utilizada para incluir, alterar, apagar, consultar e também gerenciar os dados e as tabelas dentro de um banco de dados relacional (AMAZON RDS, 2019).

Os bancos de dados relacionais possuem transações, que são uma sequência de operações que formam uma unidade lógica. Estas transações devem estar em conformidade com as propriedades Atomicidade, Consistência, Isolamento e Durabilidade (ACID), listadas a seguir (AMAZON RDS, 2019):

- Atomicidade: a transação deve ocorrer por inteiro ou, no caso de falhas, não deve realizar nenhuma operação;
- Consistência: as transações devem garantir que um dado, quando alterado passe de um estado consistente para outro também consistente;
- Isolamento: toda transação é independente por si só, desta forma garantindo que ela não seja afetada por outras transações;

- Durabilidade: é necessário que os dados após a alteração da transação continuem da mesma forma.

Exemplos de bancos de dados relacionais são: PostgreSQL[2], Microsoft SQL Server[3], Oracle[4] e o MySQL[5].

Neste modelo há vantagens e desvantagens. As vantagens, segundo Nayak et al. (2013), são:

- Possui uma linguagem de manipulação padrão, o que garante universalidade de conceitos entre os diferentes bancos de dados relacionais;
- Possui a possibilidade de implementar as propriedades ACID que proporcionam os atributos atomicidade, consistência, isolamento e durabilidade;
- Possui uma interface padrão nas ferramentas de gestão de bancos de dados.

Segundo Nayak et al. (2013), em contraponto, as desvantagens do modelo relacional são:

- Após modelado o banco de dados, o mesmo possui uma estrutura padrão e todos os registros quando inseridos devem seguir este padrão;
- Para dados em larga escala, as operações não são tão eficazes, o que torna alguns processos lentos;
- Não possuem uma escalabilidade simplificada.

2.3.2.2 Modelo Não Relacional

O banco de dados não relacional, ou NoSQL, foi utilizado pela primeira vez em 1998 em um banco de dados relacional em que foi omitido o uso da SQL. Porém este termo ganhou popularidade em 2009 nas conferências de bancos de dados não relacionais (STRAUCH, 2009).

[2] https://www.postgresql.org/
[3] https://www.microsoft.com/pt-br/sql-server/
[4] https://www.oracle.com/br/index.html
[5] https://www.mysql.com/

Este modelo não foi criado para substituir o modelo relacional, mas sim para resolver problemas que o modelo relacional não é capaz de fazê-lo adequadamente. Um exemplo deste caso é no processamento de dados em larga escala, que é realizado com maior eficiência em bancos de dados não relacionais (NTNU, 2014).

O NoSQL possui foco em escalabilidade, agilidade e flexibilidade, devido à necessidade de lidar com dados de grandes proporções (STRAUCH, 2009).

Os quatro tipos principais de bancos de dados não relacionais são: orientado a chave-valor, orientado a documentos, orientado a colunas e orientado a grafos.

O modelo orientado a chave-valor é composto por objetos. Dentro destes são armazenados um conjunto de caracteres que representam a chave, além de dados referentes ao seu valor. Para acessar estas informações é gerado um mapa que armazena as chaves de cada registro. Este modelo é recomendado para alta escalabilidade, mas não para cenários em que os dados podem possuir esquemas distintos (NAYAK et al., 2013).

Já o modelo orientado a colunas, diferentemente do modelo relacional, é armazenado em colunas, ou seja, cada coluna representa um atributo. Para que estas colunas representem os dados de um mesmo registro, é necessário criar relacionamentos entre as mesmas. Pela distribuição dos dados em colunas é possível realizar agregação dos dados de forma eficaz, desta forma este modelo é recomendado para mineração de dados (NAYAK et al., 2013).

O modelo orientado a grafos é representado por estruturas desse tipo. Tais estruturas consistem em nós e arestas, no qual cada nó representa um dado e cada aresta um relacionamento que possui uma direção. Pelo fato deste modelo possuir foco nos relacionamentos, ele é recomendado para aplicações como redes sociais, segurança e controle de acesso, redes e gestão de nuvem (NAYAK et al., 2013).

Por fim, o modelo orientado a documentos armazena seus dados em um documento, que em conjunto forma uma coleção. Cada documento possui atributos definidos por um nome e um valor. O valor pode ser representado por um tipo simples, por outros campos e também por outros documentos (ALVERMANN, 2011).

No modelo orientado a documentos, é perceptível a semelhança com o modelo relacional, porém existem diferenças nas estruturas dos registros, que podem possuir uma distinção entre si. Isso se torna uma vantagem devido à sua flexibilidade de estruturas.

Como desvantagem nota-se uma complexidade quando comparado com o modelo chave-valor, pois permite inserir os pares de valores-chave dentro do documento (NAYAK et al., 2013).

O modelo utilizado neste trabalho é o orientado a documentos, e a ferramenta gerenciadora utilizada é o MongoDB. Esta ferramenta aborda o modelo da orientação a documentos.

A opção pelo NoSQL deve-se à necessidade de se manter um projeto flexível, com fácil adaptação e maleabilidade para manutenções evolutivas do projeto, o que é proporcionado pelo modelo orientado a documentos (LÓSCIO, PONTES, OLIVEIRA, 2011).

A escolha pelo uso do MongoDB neste projeto deve-se a uma série de fatores, como a facilidade de uso, possibilidade de escalabilidade, além do propósito geral de permitir criação, atualização, leitura e exclusão de maneira simples (CHODOROW, 2013). O MongoDB também permite o uso de índices, agregação, tipos especiais de coleção e armazenamento de arquivos (CHODOROW, 2013), o que também pode ser visto como vantagens adicionais.

2.3.2.3 MongoDB 4.2.8

Essa ferramenta surgiu de uma tentativa de se desenvolver uma plataforma como serviço em nuvem de código aberto, porém não foi encontrado um banco de dados que suprisse essa necessidade. No entanto, a empresa viu a oportunidade de desenvolver um banco de dados orientado a documentos, que foi denominado MongoDB (ALVERMANN, 2011).

O MongoDB utiliza o formato BSON (JSON binário) para estruturar os documentos. Este formato é caracterizado pela eficiência e redução do espaço necessário para armazenamento que é realizado por meio de documentos, que são arquivos BSON contidos dentro de coleções, e podem ser representadas por diretórios. Cada coleção possui uma nomenclatura para agrupar os documentos semelhantes. Este modelo pode ser comparado com arquivos salvos dentro de um diretório no sistema operacional (ALVERMANN, 2011).

No Quadro 1 é possível identificar um exemplo de documento que contém informações referentes a uma coleção denominada “Produtos”, em um banco de dados orientado a documentos.

Quadro 1 – Modelo de documento armazenado

```
{
     "_id: ObjectId("5eb8106de6274904d4dc9b8b")
      nome: "Maçã Gala"
      descricao: "Maçã Gala"
      preco: 10
      desconto: 0
      gtin: "11111111111"
      __v:0
}
```

Fonte: Autoria própria

Entre as vantagens do MongoDB, é possível identificar que o mesmo fornece recursos como consistência, tolerância a falhas e persistência. Também é possível utilizar recursos de agregação, consultas e indexação, o que permite realizar buscas mais completas e mais rápidas (NAYAK et al., 2013).

Em contrapartida, este recurso de indexação acaba ocupando muito espaço dentro da estrutura do banco de dados, desta forma, aumentando a quantidade de recurso computacional necessário para otimizar determinadas buscas (NAYAK et al., 2013).

2.3.3 Vue.js (versão 2.6.11)

Vue.js é uma biblioteca JavaScript de prototipação rápida, com o objetivo de criar interfaces, que são elementos de comunicação entre sistemas, para o desenvolvedor de maneira eficaz.

O Vue.js foi criado por Evan You, quando trabalhava no laboratório de criação da Google. Evan precisava prototipar grandes interfaces de usuário sem que fosse necessário repetir estruturas HTML. Desta forma, viu a oportunidade de desenvolver uma ferramenta que suprisse sua necessidade (FILIPOVA, 2016).

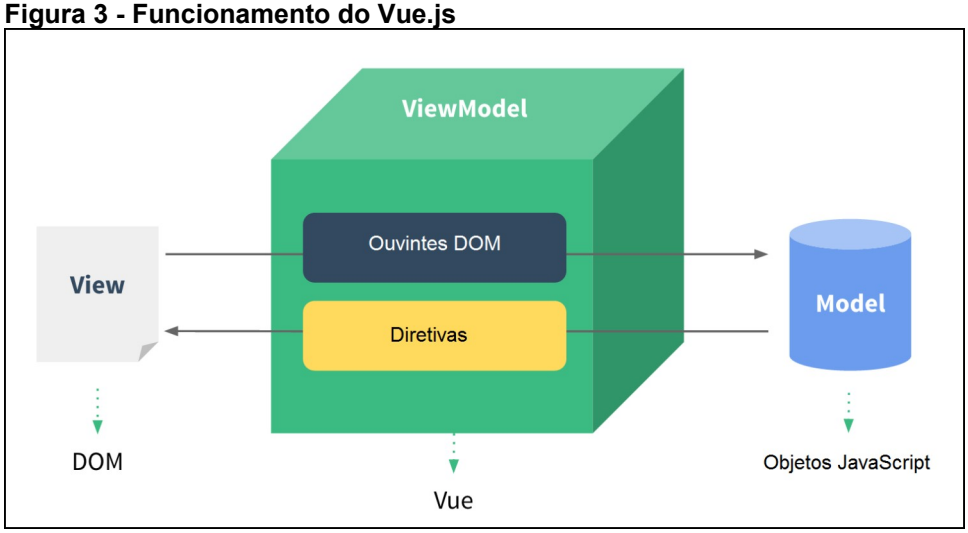
Para a escolha desta ferramenta no desenvolvimento deste trabalho, foram considerados alguns tópicos constatados pela própria equipe Vue.js, que realizou um comparativo com outros *frameworks* de grande representatividade no mercado de desenvolvimento *web* (VUE.JS, 2019). São estes:

- O Vue.js utiliza HTML, porém de maneira reduzida e eficaz, desta forma, facilita a migração de aplicações desenvolvidas apenas com HTML.
- Como é necessário usar código HTML no *framework*, a compreensão, aprendizado e contribuição dos desenvolvedores *web* iniciantes em um projeto utilizando Vue.js, ocorre de forma facilitada.
- Quando o objetivo é desenvolver aplicações mais robustas, é necessário utilizar soluções de roteamento[6]. Em comparação ao *framework* React[7], o Vue.js possui soluções oficiais para estas questões. Isso é proposital, pois o intuito é manter estas preocupações com a comunidade, preservando a ferramenta em um ecossistema[8] mais fragmentado.
- Ao comparar com o *framework* Angular[9], um dos principais pontos a se destacar é a curva de aprendizado. Percebe-se que para iniciar com o Angular é necessário muito mais esforço, tornando a curva de aprendizado maior comparado ao Vue.js.

A Figura 3 apresenta o funcionamento do *framework* Vue.js, que atua na camada ViewModel a qual interage com as camadas View e Model.

[6] Roteamento é definido pelo processo de direcionar as interfaces do usuário através das URLs no navegador.
[7] https://pt-br.reactjs.org
[8] Ecossistema é definido por tudo aquilo que envolve a tecnologia em questão, sejam ferramentas de apoio à esta tecnologia ou também à comunidade de desenvolvedores que a utiliza.
[9] https://angular.io

Figura 3 - Funcionamento do Vue.js



Fonte: Vue (2020)

Na Figura 3 é possível identificar a nomenclatura DOM que é definida pela interface que renderiza documentos de extensão HTML (MOZILLA, 2019).

O Vue.js garante vantagens como reuso de componentes e permite que sejam ligados os modelos de dados à camada de representação. Os modelos são objetos JavaScript puros e podem ser utilizados em qualquer parte da camada de visualização da sua aplicação.

Como desvantagens, é possível identificar que o Vue.js possui uma comunidade menor, portanto possui menos recursos disponíveis para consulta. Além disso, a equipe de desenvolvimento do Vue.js possui uma quantidade reduzida de desenvolvedores comparado a outras equipes tecnologias, deste modo, projetos de grande porte podem não ter suporte corporativo (FILIPOVA, 2016).

As comparações realizadas entre o Vue.js com outros *frameworks* leva em consideração os principais *frameworks* disponíveis na atualidade, sendo eles, o ReactJS e o Angular (VUE.JS, 2019).

As demais tecnologias e ferramentas apresentadas no presente trabalho, aliadas ao Vue.js contemplam o conjunto denominado MEVN *Stack*. Este conjunto possui este nome devido às iniciais das principais ferramentas utilizadas que são o MongoDB, Express, VueJS e NodeJS (VUE.JS, 2019).

Existem também outras *stacks* conhecidas como MEAN *Stack* e MERN *Stack* que são diferentes da atual apenas pela terceira letra do acrônimo que é representado pelos *frameworks* Vue.js, Angular e ReactJS. Porém, a escolhida para

o presente trabalho deve-se pela comparação entre estes *frameworks*, levando em consideração que as outras tecnologias utilizadas nas *stacks* são as mesmas.

2.3.4 Express (versão 4.17.1)

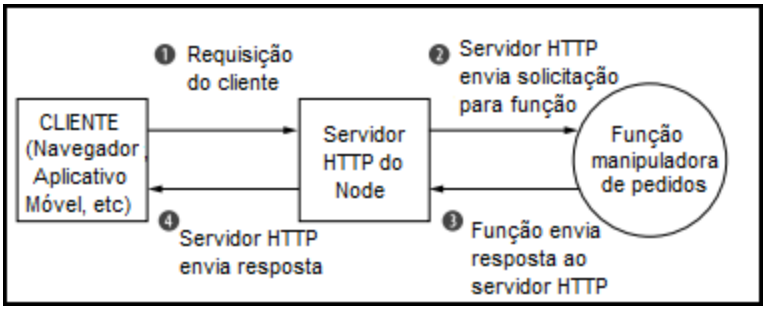
Express é um *framework* relativamente pequeno baseado nas funcionalidades do NodeJS como servidor *web.*

Este *framework* possui o objetivo de simplificar a utilização de APIs, além de possuir outras funcionalidades que tornam o desenvolvimento de uma aplicação servidora com NodeJS mais rápido (HAHN, 2016).

Quando uma aplicação é desenvolvida apenas com o NodeJS, é necessário criar uma função JavaScript no lado do servidor que tratará as informações recebidas. Esta é chamada de *Request Handler* (HAHN, 2016). Esta função é responsável pela lógica para responder a requisição do cliente.

Na Figura 4 é apresentado o fluxo das requisições de uma aplicação NodeJS onde existe um cliente, representado por uma interface de usuário, que permite o envio de requisições para o servidor do NodeJS. Dentro deste servidor estão disponíveis funções para manipular as informações das requisições e a partir disto retornará uma resposta do servidor para o cliente.

Figura 4 - Fluxo de requisição em aplicação NodeJS

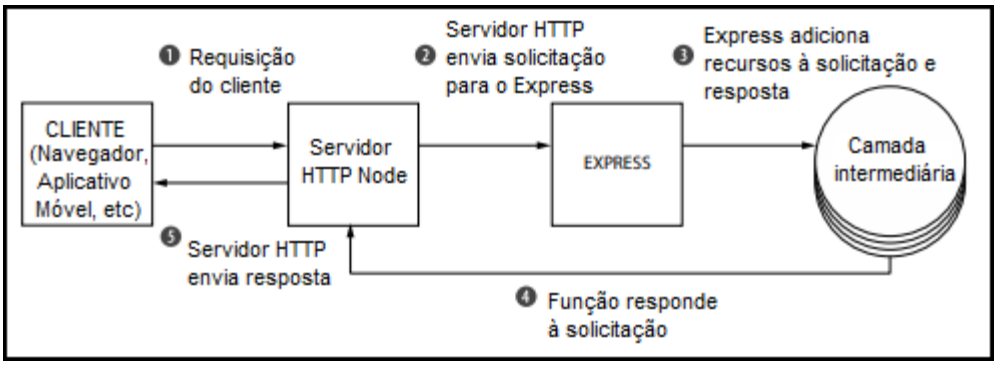


Fonte: HAHN (2016)

Ao utilizar o *framework* Express, uma camada a mais é adicionada entre o servidor HTTP e a função manipuladora de requisição e respostas, fornecendo funcionalidades aos serviços de requisição e de resposta. Ele também permite a criação de vários *Request Handler's* menores, garantindo assim mais facilidade na manutenção e na modularização (HAHN, 2016).

A Figura 5 apresenta a camada onde será inserido o Express. Em comparativo com a Figura 4, o Express é adicionado entre as etapas 2 e 3.

Figura 5 - Fluxo de requisição em aplicação com Express



Fonte: HAHN (2016)

Como vantagens do Express é possível citar o seu formato minimalista com uma sintaxe simplificada. Além disso, ele apresenta uma estrutura baseada no recebimento de requisições HTTP por um cliente e retorno de respostas também por HTTP (BROWN, 2020).

Em contrapartida, vale ressaltar que o Express não é eficiente quando se trata de grandes processamentos de *Central Process Unit* (CPU), como por exemplo, em requisições envolvendo áudios, vídeos e processamento gráfico. O Express também não possui maturidade em comparação com outras tecnologias, desta forma, apresentando várias falhas (PADIA, 2018).

2.3.5 OpenStreetMap

Para o cálculo das distâncias entre os estabelecimentos foi necessário fazer requisições à API do OpenStreetMap[10]. O OpenStreetMap é uma base de dados geográficos gratuita, que é mantida por uma comunidade que popula tal base inserindo informações referentes às ruas, rotas de ônibus, construções, praias e qualquer informação pertinente usada para localização geográfica (BENNETT, 2010).

As vantagens do OpenStreetMap, segundo BENNETT (2010), são:

- Projeto de código aberto, ou seja, acessível para ajustes e modificações;

[10] www.openstreetmap.org

- Não possui restrições de uso, inclusive para uso comercial;
- Comparado às outras bases de dados geográficos, o OpenStreetMap recebe atualizações com mais frequência;
- Pode ser utilizado por meio de APIs padronizadas.

Em contrapartida, as desvantagens são:

- Por ser um projeto de código aberto, ainda faltam informações sobre algumas regiões;
- Algumas informações podem estar desatualizadas.

Por meio da API do OpenStreetMap foi possível calcular as distâncias entre o estabelecimento de venda e a localização do usuário enviando os pontos de partida e de chegada, desta forma, retornando o valor referente à distância entre estes dois pontos.

3 MÉTODOS

Neste capítulo é apresentado a análise de *softwares* similares ao desenvolvido neste projeto, e um comparativo do funcionamento da ferramenta em relação a este projeto.

3.1 SISTEMAS SIMILARES

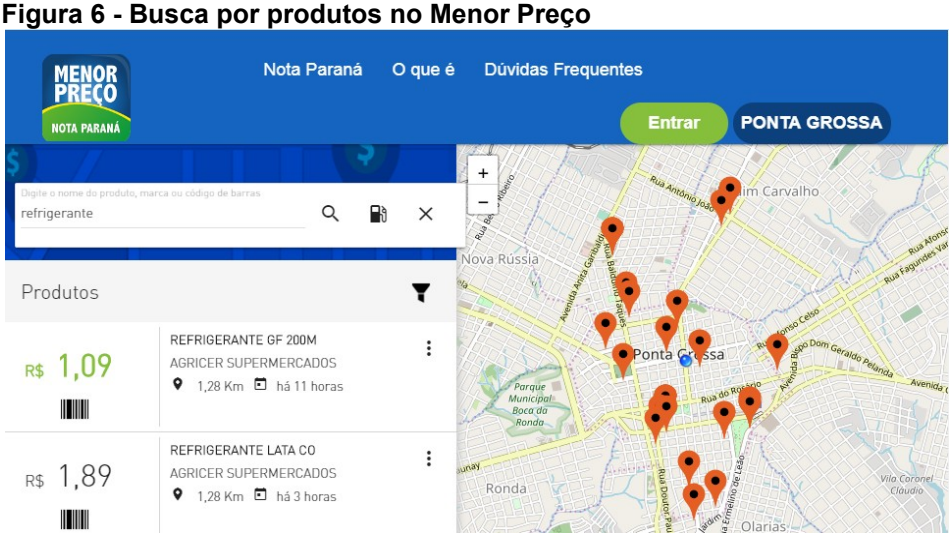
A partir de buscas por sistemas que possuem a proposta de permitir o comparativo de preços entre os produtos dos supermercados, foram encontradas duas ferramentas com o mesmo resultado final.

O primeiro aplicativo é o Menor Preço[11], que permite a criação de uma lista de compras e também apresenta os estabelecimentos por meio de um mapa.

Na tela de busca de produtos é apresentado um mapa conforme a região selecionada e um campo de texto para inserir o nome do produto que se procura. Ao realizar a busca é apresentada uma lista de categorias dos produtos da qual uma deve ser selecionada. Após essa seleção é apresentada uma lista com produtos disponíveis nos supermercados da região com seus respectivos preços.

A Figura 6 apresenta um exemplo da busca de produtos.

Figura 6 - Busca por produtos no Menor Preço



Fonte: Autoria própria.

[11] https://menorpreco.notaparana.pr.gov.br

Após o cadastro no *website* do desenvolvedor é possível a criação de uma lista de compras com os produtos que se compra regularmente. A partir dessa lista será comparado o preço total da compra, bem como a localização dos estabelecimentos que possuem os produtos, com ordenação por custo total da lista.

As informações sobre os produtos são obtidas por meio dos dados da nota fiscal emitida no ato do pagamento nos estabelecimentos. Esses dados são enviados para a Secretaria de Estado da Fazenda do Paraná, permitindo assim que os preços estejam atualizado conforme são vendidos os produtos.

Outro sistema que possui a mesma proposta final é o Pinngo[12]. Este é um aplicativo para dispositivos móveis que apresenta um objetivo semelhante ao Menor Preço.

Com ele é possível buscar por produtos e comparar preços, porém valores dos produtos são atualizados pelos próprios usuários do aplicativo. Para isto, o usuário pode utilizar o *smartphone* e escanear o QR Code[13] da sua nota fiscal após a compra e enviar por meio do aplicativo. Desta forma, atualizará a base de dados do sistema.

Os sistemas Menor Preço e Pinngo possuem a proposta de permitir o comparativo de preços, assim como este projeto. Contudo, a diferença entre os dois aplicativos e o presente trabalho pode ser identificada na forma de alimentar a base de dados.

Neste trabalho, os dados são alimentados através de uma página no próprio *site*. O estabelecimento pode acessar a página e por meio do mesmo cadastrar e atualizar os produtos. A principal vantagem utilizando desta forma de alimentar o banco de dados é a garantia de que o próprio estabelecimento terá controle das informações. Já a desvantagem é que o trabalho acabaria sendo manual para os responsáveis em manter os produtos atualizados.

3.2 MÉTODO DE DESENVOLVIMENTO

Para o desenvolvimento do presente projeto foi utilizado o modelo Kanban (OHNO, 2013).

[12] www.pinngo.com.br
[13]Código barrametrico bidimensional semelhante a um código de barras.

Em relação ao modelo Kanban, segundo Corona e Pani (2013), ele é propício para manter a organização das atividades em processo. Ele permite a visualização das atividades pendentes e atividades concluídas, desta forma, organizando o cronograma das etapas do projeto.

O motivo que levou a escolha pela utilização do modelo Kanban refere-se ao fato de que o presente projeto foi desenvolvido por um único integrante e pela quantidade de recursos disponíveis reduzida, além da necessidade de definir metas para o desenvolvimento de cada funcionalidade.

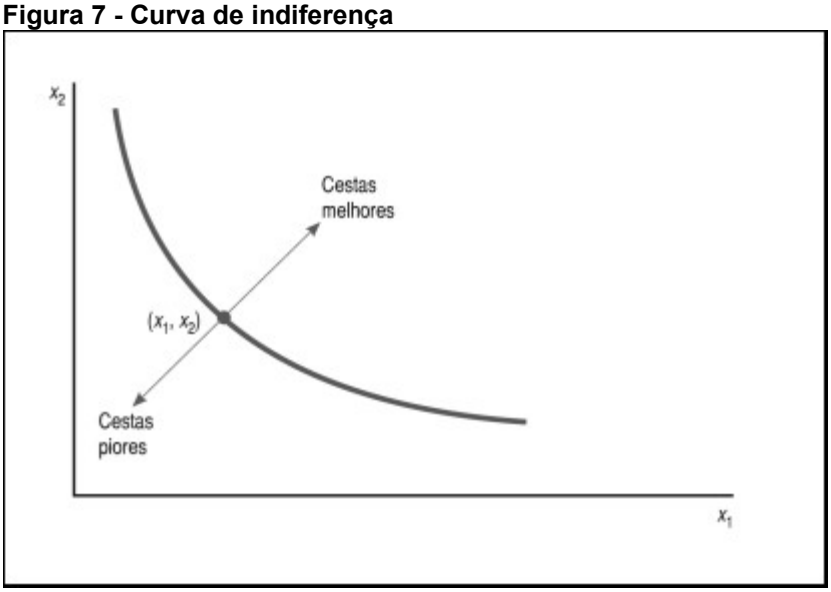
3.3 FUNDAMENTAÇÃO PARA GERAÇÃO DO ALGORITMO DE MELHOR CUSTO X BENEFÍCIO

Para o desenvolvimento do algoritmo disponível no Apêndice A, foram utilizados os conceitos de preferências definidos por Varian (2012).

Segundo Varian (2012), o comportamento do consumidor é definido pelas escolhas dos melhores produtos que os mesmos podem pagar. Para isso, a definição de melhores produtos é descrita como preferência.

Esta preferência foi descrita por meio de um gráfico em que apresenta a curva de indiferença, como mostrado na Figura 7. Esta curva é formada pelas combinações de consumo que proporcionam o mesmo nível de satisfação para o consumidor que é definida na microeconomia como utilidade. Uuma curva de indiferença deve apresentar as funções de combinação de consumo que possuem o mesmo grau de utilidade (MANKIW, 2015).

Figura 7 - Curva de indiferença



Fonte: VARIAN (2012)

A curva de indiferença nos permite analisar o quão disposto o cliente está a trocar um produto por outro. Portanto, é possível encontrar curvas de indiferença que são representadas por retas. Estas retas mostram que os produtos que compõe a curva de indiferença são substitutos perfeitos, ou seja, eles podem ser substituídos sem perder seu valor final (MANKIW, 2015).

Para exemplificar este cenário, é possível apresentar uma situação em que estão presentes dois pacotes com moedas de 5 e 10 centavos. Pouco importa a quantidade de moedas de cada valor, contanto que cada pacote possua o mesmo valor final. Ou seja, 1 moeda de 10 centavos sempre poderá ser substituída por 2 moedas de 5 centavos e a recíproca também é válida (MANKIW, 2015).

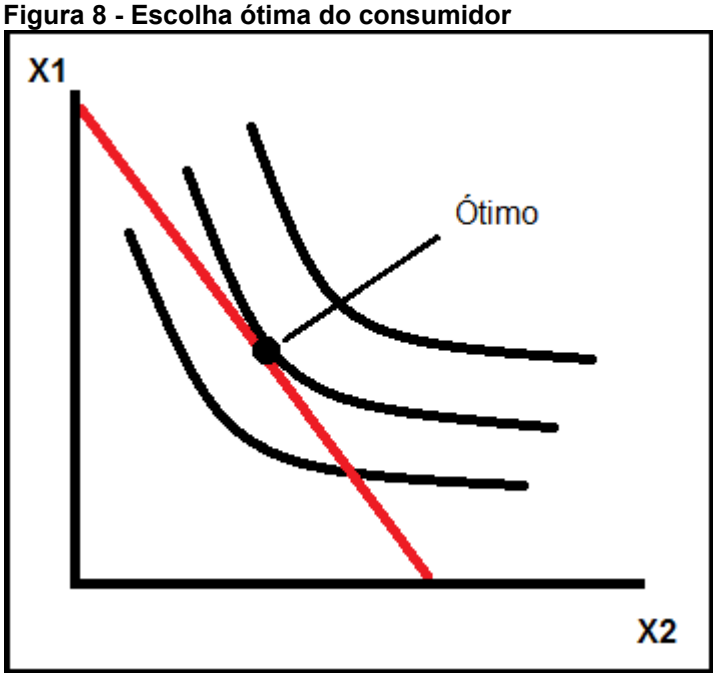
As curvas de indiferença normalmente são convexas e podem apresentar variações na sua curvatura. Quanto menos convexa a curva for, mais facilmente substituíveis são os bens em comparação. Quando uma curva é muito convexa, os bens são dificilmente substituíveis. Isto deve-se à taxa marginal de substituição, que representa quanto o consumidor precisa de um produto para substituir por outro (VARIAN, 2012).

Além dos conceitos acima, é necessário compreender que para a otimização da escolha do consumidor compreende-se o conceito de restrição orçamentária. Esta é definida pelo quanto o mesmo pode gastar. Esta quantidade é formada por

uma função de primeiro grau que em um gráfico é visto como uma reta (MANKIW, 2015).

Para encontrar a melhor escolha do consumidor, é necessário encontrar o ponto tangente entre a reta orçamentária e a curva de indiferença, desta forma, resultando na escolha ótima (MANKIW, 2015), como é ilustrado na Figura 8.

Figura 8 - Escolha ótima do consumidor



Fonte: Adaptado de Mankiw (2015)

Portanto, pode-se concluir que a melhor escolha do consumidor está limitada a sua restrição orçamentária, logo quanto mais de um produto ele possa pagar pelo menor preço melhor (VARIAN, 2012).

Para este projeto, leva-se em consideração que a restrição orçamentária é imposta pelo próprio consumidor ao selecionar os produtos que o mesmo deseja comparar, desta forma, fica em função do algoritmo apenas selecionar os produtos dos estabelecimentos que possuem o menor custo na somatória dos preços.

Além desta fundamentação para geração do relatório, também é necessário compreender como serão comparados os produtos. Para isso é utilizado um campo identificador chamado GTIN (*Global Trade Item Number).* Este campo é um padrão criado pela GS1 Brasil, organização que desenvolve e mantém padrões de comunicação, e foi diretamente criado para facilitar o processo de comparação de produtos em plataformas de venda e comparadores de preço (GS1 Brasil, 2020).

4 DESENVOLVIMENTO

Este capítulo aborda o processo de desenvolvimento do projeto além de mostrar a relação entre as ferramentas utilizadas. Também são apresentados os resultados obtidos por meio de ilustrações do próprio produto final.

4.1 ABORDAGEM INICIAL

O início do desenvolvimento do projeto ocorreu por meio da elaboração das funcionalidades esperadas que o sistema deveria possuir. Estas funcionalidades são:

- Cadastro de usuário;
- Alteração dos dados dos usuários;
- Login com autenticação;
- Cadastro de produtos;
- Alteração dos dados dos produtos;
- Pesquisa dos produtos;
- Cadastro de lista de compras;
- Geração de relatório de melhor custo x benefício para a lista de compras.

Estas funcionalidades foram elencadas com base nos sistemas similares citados no Capítulo 3.1.

Como este projeto é dividido em uma API do lado servidor utilizando NodeJS e uma API no lado cliente utilizando Vue.js, foi possível segmentar o desenvolvimento de cada funcionalidade entre o lado servidor e o lado cliente, desta forma, gerando um *Minimum Viable Product* (MVP)[14] ou Produto Mínimo Viável para cada funcionalidade.

Para o desenvolvimento do lado do servidor foi necessário criar os *controllers* e *models* para cada funcionalidade. O *controller* é um *script* que possui

[14] Minimum Viable Product é uma versão pequena do produto com funcionalidades básicas de forma a satisfazer as necessidades do cliente e obter uma opinião do mesmo para futuras melhorias (RIES, 2009).

um conjunto de funções referentes a um contexto com o objetivo de comunicar a requisição recebida do cliente e realizar um processamento em dados recebidos, bem como comunicar com o banco de dados, se necessário.

Models são objetos previamente definidos que mantêm a estrutura padrão dos dados. Neste projeto, os *models* foram definidos a partir do framework Mongoose que utiliza esquemas de dados, desta forma garantindo as operações de inserção, atualização, exclusão e leitura no banco de dados MongoDB.

Para iniciar o desenvolvimento listou-se as atividades conforme o Quadro 2.

Quadro 2 – Atividades de desenvolvimento do projeto

Atividade	Objetivo
Instalação e configuração das ferramentas de desenvolvimento	Instalar as ferramentas Vue.js, NodeJS, Express, Mongoose e Bootstrap
Cadastro de usuário	Desenvolver no lado servidor o *controller* e o *model* referente ao cadastro de usuário. Desenvolver no lado cliente a página *web* de cadastro de usuário.
Login com autenticação	No lado servidor, implementar a geração e a validação de *token* de *autenticação.* No lado cliente, implementar a tela de login e armazenamento de *token* de autenticação no navegador.
Alteração dos dados dos usuários	No lado servidor, implementar as rotas para alteração de dados dos usuários no Banco de Dados. No lado cliente, implementar a tela para receber os dados do usuário a serem alterados.
Cadastro de produtos	No lado servidor, implementar as rotas de cadastro das informações dos produtos. No lado cliente, implementar a tela para o cadastro das informações dos produtos. Essa função é permitida apenas para o perfil dos estabelecimentos.
Alteração de produtos	No lado servidor, implementar as rotas

	para atualização dos dados dos produtos. No lado cliente, implementar a tela para realizar a atualização dos produtos. Essa função é permitida apenas para o perfil dos estabelecimentos.
Pesquisa dos produtos	Implementar no lado servidor, rota para realizar a busca dos produtos. Implementar o componente de busca, no lado cliente, para consumir a rota de busca de produtos.
Cadastro de lista de compras	Realizar o desenvolvimento do *controller* e *model* da lista de compras no lado servidor. No lado cliente, desenvolver a página de cadastro de lista de compras.
Geração do relatório de melhor custo benefício	Implementar geração de relatório no lado servidor e desenvolver a página de relatório no lado cliente.

Fonte: Autoria própria

4.2 INSTALAÇÃO E CONFIGURAÇÃO DAS FERRAMENTAS

Esta etapa foi dividida em duas partes. A primeira consistiu na instalação e configuração do lado servidor. A segunda referente a instalação e configuração do lado cliente. Esta distinção dos dois lados permite com que os mesmos executem de maneira independente e em caso de alterações em um dos lados não afetará diretamente o outro.

No lado servidor foi instalado o NodeJS, junto com o Mongoose. Além disso, foram instaladas as bibliotecas body-parser, CORS e jsonwebtoken.

O body-parser permitiu que a comunicação por meio de arquivos JSON fosse simplificada, pois o mesmo permite a obtenção dos dados do corpo do arquivo JSON de forma direta por intermédio de um atributo do próprio objeto.

O mecanismo CORS, por meio de sua própria biblioteca permite com que recursos restritos em uma determinada página *web* sejam solicitados por um domínio externo.

A biblioteca Jsonwebtoken foi utilizada na etapa de login com autenticação para garantir uma maior segurança aos dados do usuário e reduzir os riscos de terceiros estarem utilizando credenciais de usuários legítimos.

No lado cliente foi instalado o Vue.js, junto com as bibliotecas Fortawesome, Axios, Vee-validate, Vue-router e Vuex.

O Fortawesome foi utilizado para simplificar o desenvolvimento das páginas, de forma a permitir uma interface mais atraente e com maior usabilidade para o usuário.

Para realizar as requisições HTTP foi utilizada a biblioteca Axios que apresenta uma sintaxe simplificada para a comunicação com o lado servidor. No Quadro 3 é possível identificar uma requisição feita utilizando o método POST.

Quadro 3 – Função POST da biblioteca axios

```
1    axios.post('http://localhost:3000/listacompras/', {
2        produtos: self.compras
3    })
4    .then(function(response) {
5        Console.log(response);
6    })
7    .catch(function(error) {
8        Console.log(error);
9    });
```

Fonte: Autoria própria

Na linha 1 do Quadro 3, o axios é um objeto instanciado a partir da biblioteca do próprio axios e através dele é chamada a função *post*. Ao executar esta função é realizada uma requisição POST no endereço que é passado como primeiro parâmetro deste método. Após a requisição é então executada a função descrita a partir da linha 4. Em caso de erro será executada a linha 7.

A biblioteca Vuerouter foi utilizada para permitir que a aplicação do lado cliente seja desenvolvida com rotas. Estas rotas permitem que a aplicação navegue

entre as páginas do aplicativo de forma simplificada. O Quadro 4 apresenta um exemplo do uso do Vuerouter por meio de um *link* que, ao ser clicado, fará com que a página carregue a nova página indicada pela rota.

Quadro 4 – Estrutura de rota do vuerouter

```
      <router-link to =”/>Home</router-link>
      <router-link to =”/login”>Login</router-link>
```

Fonte: Autoria própria

A tag <router-link> é semelhante à tag HTML <a> que permite definir um *hyperlink* dentro de um texto. Neste caso, ao clicar no texto *Login*, o navegador carregará a página referente a rota definida dentro do código.

A biblioteca Vuex foi utilizado para suprir a demanda de desenvolver uma aplicação *Single Page Application* (SPA). Uma SPA é uma aplicação que atualiza o conteúdo de uma página de acordo com uma programação definida.

Desta forma, a aplicação não necessita renderizar toda a estrutura da página ao redirecionar para uma outra página da aplicação. Como por exemplo, páginas com cabeçalhos fixos não precisam renderizar o cabeçalho toda vez que redirecionar para outra página que também possua este mesmo cabeçalho, garantindo uma transição mais rápida dentro da aplicação.

4.3 CADASTRO DE USUÁRIO

No desenvolvimento do cadastro de usuário do lado servidor foi necessário implementar um *controller* com os métodos GET, POST, PATCH e DELETE para que, desta forma, fosse possível selecionar, inserir, atualizar e apagar um usuário, como é possível identificar no Quadro 5.

Quadro 5 – Método GET Usuário

```
1    router.get(‘/’, async (req, res) => {
2         Try{
3              Const usuarios = await Usuario.find();
4                   Res.json(usuarios);
5              }catch(err){
```

```
6                   Res.json({message: err});
7               })
8         });
```

Fonte: Autoria própria

Na linha 1 do Quadro 5 identifica-se a chamada da função get do objeto router. A execução desta função permite que quando a aplicação receber uma requisição de método GET no endereço passado no primeiro parâmetro será executada a linha 2.

No Quadro 6 pode ser observado um exemplo do método POST.

Quadro 6 – Método POST Usuário

```
1    router.post('/', (req, res) => {
2               const usuario = new Usuario({
3                    nomeCompleto: req.body.nomeCompleto,
4                    email: req.body.email,
5                    senha: req.body.senha,
6                    perfil: req.body.perfil
7               });
8               usuario.save()
9               .then(data => {
10                   res.json(data)
11              })
12              .catch(err => {
13                   res.json({message: err});
14              })
15        });
```

Fonte: Autoria própria

Na linha 1 do Quadro 6 é possível identificar a execução da função post do objeto router. Quando esta aplicação receber uma requisição no endereço passado no primeiro parâmetro e do tipo POST, será então executada a linha 2.

Na linha 2 inicia-se a criação de um novo objeto Usuario tendo como base a classe Usuario. Então na linha 8 este novo objeto é salvo no banco de dados através

da função save. Caso não ocorram erros, a linha 10 será executada e retornará como resposta da requisição o objeto criado.

No Quadro 7 é apresentado um exemplo do método PATCH.

Quadro 7 – Método PATCH Usuário

```
1    router.patch('/:usuarioId', async (req, res) => {
2         try{
3              const updatedUsuario = await Usuario.updateOne(
4                   {_id: req.params.usuarioId},
5                   {$set:           {           nomeCompleto:
req.body.nomeCompleto,
6                   email: req.body.email,
7                   senha: req.body.senha,
8                   perfil: req.body.perfil }}
9              );
10             res.json(updatedUsuario);
11        }catch(err){
12             res.json({message: err)};
13        }
14   });
```

Fonte: Autoria própria

Na linha 1 do Quadro 7 é possível identificar a execução da função patch do objeto router. Esta função é executada quando o sistema recebe uma requisição do tipo PATCH no endereço passado no primeiro parâmetro. Assim que é recebida a requisição executa-se a linha 2.

Na linha 3 é realizada a atualização do objeto que em questão com os novos dados recebidos no corpo da requisição. Estes dados podem ser acessados através do identificador req.params.

Caso não ocorram erros, a linha 10 executará retornando à requisição o objeto atualizado.

No Quadro 8 pode ser observado um exemplo do método DELETE.

Quadro 8 – Método DELETE Usuário

```
1    router.delete(‘/:usuarioId’ async (req, res) => {
2         try{
3              const removedUsuario = await Usuario.deleteOne(
4                   {_id: req.params.usuarioId});
5              Res.json(removedUsuario);
6         }catch(err){
7              Res.json({message: err});
8         }
9    });
```

Fonte: Autoria própria

Na linha 1 do Quadro 8 pode-se identificar a execução da função delete do objeto router. Esta função é executada quando a aplicação recebe uma requisição do tipo DELETE no endereço referente ao primeiro parâmetro.

Na linha 3 então é executada a função deleteOne do objeto Usuario onde a mesma irá excluir um dado do banco conforme o valor passado na linha 1 identificado por :usuarioId.

Assim que executada a função, caso não ocorram erros será retornado à requisição o retorno da função.

Também foi necessário implementar um esquema do Mongoose apresentando os atributos do usuário, incluindo o tipo do atributo e demais validações para este dado. No Quadro 9 pode ser observado um exemplo do esquema para uso do Mongoose.

Quadro 9 – Esquema Usuário para Mongoose

```
1    const mongoose = require(‘mongoose’);
2    const UsuarioSchema = mongoose.Schema({
3         nomeCompleto: {
4              type: String,
5              required: true
6         },
7         email: {
8              type: String,
```

```
9               required: true,
10              unique: true
11          },
12          senha: {
13              type: String,
14              required: true,
15              select: false
16          },
17          perfil: {
18              Type: Number,
19              Required: true
20          }
21      });
22      UsuarioSchema.plugin(uniqueValidator);
23      module.exports = mongoose.model('Usuarios',UsuarioSchema);
```

Fonte: Autoria própria

Na linha 1 do Quadro é importada a biblioteca do mongoose. Já na linha 2 é instanciado o schema que representa o formato dos objetos da classe Usuario para o mongoose.

Para a implementação no lado cliente foi elaborada uma página utilizando as funcionalidades do Vue.js junto com a biblioteca Axios para realizar as requisições ao servidor.

4.4 LOGIN COM AUTENTICAÇÃO

O diferencial encontrado para desenvolver esta etapa encontra-se no uso da biblioteca Jsonwebtoken (JWT) como forma de autenticação. O JWT é um método RCT 7519 padronizado pela indústria. Este método possibilita a autenticação entre duas partes por meio de um *token* que é representado por uma *string* em formato Base64.

Para isto, foi criado um ponto de acesso no lado servidor para receber a requisição de login e validar as credenciais do usuário. Sendo estes válidos, o

servidor retorna um *token* para o lado cliente armazenar e validar todas as requisições que realizará.

O Quadro 10 apresenta um exemplo de autenticação das credenciais do usuário que retorna o *token* para o lado cliente.

Quadro 10 – Exemplo de geração do token

```
1    app.post('/auth/signin', async (req, res) => {
2         const { email, password } = req.body;
3         const user = await
Usuario.findOne({email}).select('+senha');
4         if(!user) return res.sendStatus(400);
5         if(!(password == user.senha)) return
res.sendStatus(400);
6         const accessToken = jwt.sign(user.id,
process.env.ACCESS_TOKEN_SECRET);
7         res.json({accessToken:accessToken});
8    });
```

Fonte: Autoria própria

Na linha 1 do Quadro 10, é possível identificar a execução da função post do objeto app. O primeiro parâmetro desta função representa o valor da URL que estará recebendo as requisições.

Quando recebida uma requisição, a linha 2 é executada recebendo os valores das variaveis email e password através do corpo da requisição.

Na linha 3 é executada a função findOne do objeto Usuario para identificar se aquela conta já existe no banco de dados. Caso o usuário não seja encontrado na linha 4, será retornado o *status* 400, conforme os padrões HTTP.

Na linha 5 consta uma estrutura de decisão *if* para identificar se a senha do usuário armazenada no banco de dados é igual a senha enviada na requisição. Caso sejam iguais o token é gerado na linha 6 e retornado à requisição. Caso sejam diferentes será retornado um *status* 400.

No Quadro 11 é possível identificar a função que é inserida como intermediária nos pontos de acesso da aplicação e que valida o *token* que o lado cliente possui armazenado.

Quadro 11 – Exemplo de autenticação do token

```
1    app.get('/listacompras', authenticate(), (req, res, next)
=> { console.log('Autenticado');
2    function authenticate(req, res, next){
3         const authHeader = req.headers['authorization'];
4         const token = authHeader && authHeader.split(' ').
[1];
5         if(token == null return res.sendStatus(401);
6         jwt.verify(token,    process.env.ACCESS_TOKEN_SECRET,
(err, user) => {
7              if(err) return res.sendStatus(403);
8              req.user = user;
9              next();
10        })
11   });
```

Fonte: Autoria própria

Na linha 1 do Quadro 11 é possível identificar a chamada do método get do objeto app. O primeiro parâmetro desta função representa a URL que estará recebendo a requisição.

Na linha 3 é atribuído à variável authHeader o valor presente no cabeçalho da requisição indicado pelo atributo authorization. Na linha 4 então é ajustado o valor da variável authHeader para remover espaços em branco.

Já na linha 5 é verificado se o valor recebido no cabeçalho é nulo. Caso seja, será retornado um *status* 401 como resposta à requisição. Na linha 6 então é apresentada uma verificação do *token* caso o mesmo não seja nulo.

4.5 CADASTRO LISTA DE COMPRAS

Para o cadastro da lista de compras no lado do servidor foi necessário implementar o *controller* com os pontos de acesso para os métodos GET, POST, PATCH e DELETE. Também implementou-se o esquema do Mongoose apresentando seus atributos e suas validações.

No lado do cliente foi desenvolvida a página com um campo de busca dos produtos cadastrados na parte superior da tela. Abaixo do campo de busca está disponível uma lista que armazenará os produtos adicionados pelo usuário. Com base nesta lista será gerado o relatório.

4.6 GERAÇÃO DO RELATÓRIO DE MELHOR CUSTO BENEFÍCIO

Esta é a funcionalidade principal do projeto, pois é a partir do algoritmo de análise dos produtos que será gerado o relatório com os 3 estabelecimentos com os melhores custo benefício, baseados na disponibilidade e preço dos produtos que compõe a lista de compras cadastrada pelo usuário. Além disso, foi a etapa mais extensa de desenvolvimento, pois foi necessário implementar o algoritmo de validação da preferência do consumidor, baseado na disponibilidade do produto no estabelecimento e no seu preço.

O algoritmo desenvolvido para esta etapa consta no Apêndice A e leva em consideração a disponibilidade dos produtos e seus preços. Por exemplo, ao cadastrar uma lista com quatro produtos, existem os seguintes cenários possíveis:

- Um estabelecimento possui todos os produtos da lista de compras;
- Um estabelecimento possui apenas alguns dos produtos da lista;
- Um estabelecimento possui nenhum dos produtos da lista.

Caso o estabelecimento não possua nenhum dos produtos cadastrados na lista de compras, o mesmo não será considerado para análise de custo benefício.

Caso todos os estabelecimentos analisados possuam todos os produtos cadastrados na lista de compras, a análise será feita através de um comparativo com o somatório dos preços de cada estabelecimento. O estabelecimento que possuir o menor valor total será apresentado como o melhor custo benefício. Em caso de empate, o estabelecimento mais próximo, conforme dados informados no cadastro do usuário pessoa física e do usuário pessoa jurídica, terá a preferência.

4.7 RESULTADO DO PROJETO

O resultado final da aplicação desenvolvida é apresentado nas Figuras 9, 10, 11 e 12:

a) Para utilizar a função de cadastro de compras e geração de relatório é necessário realizar um cadastro utilizando um email e uma senha. A Figura 9 demonstra a tela de cadastro do usuário:

Figura 9 - Exemplo de cadastro de usuário

Fonte: Autoria própria

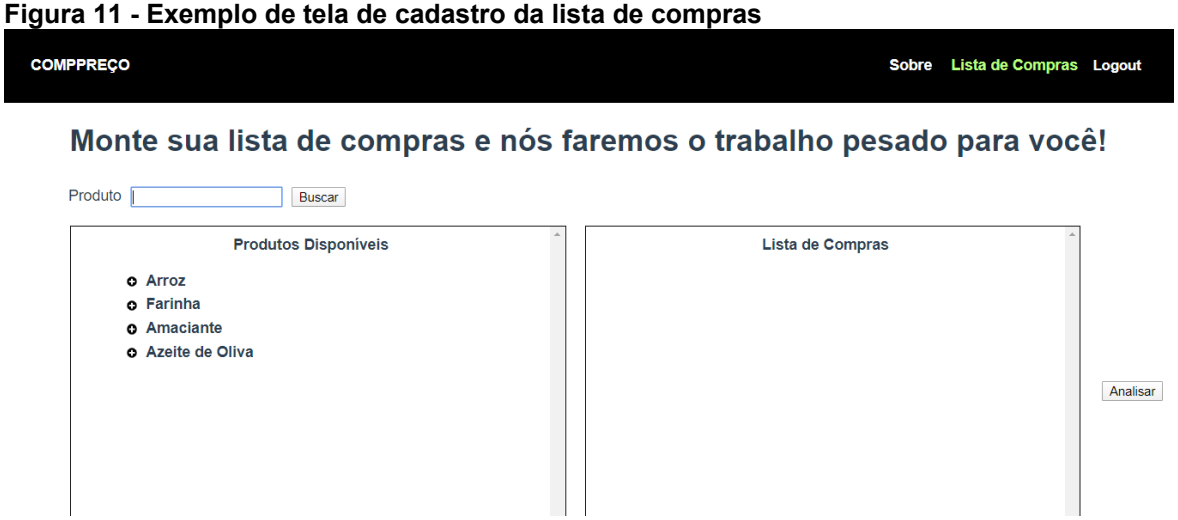
b) Após realizar o cadastro será possível realizar o login por intermédio da tela apresentada na Figura 10:

Figura 10 - Exemplo de tela de login

Fonte: Autoria própria

c) Com o acesso realizado com sucesso, será disponibilizada a página para cadastro da lista de compras, que possui um campo de buscas para os produtos e duas listas divididas em “Produtos Disponíveis” e “Lista de Compras”. Ao final da seleção dos produtos é possível clicar no botão “Analisar”, conforme ilustrado na Figura 11:

Figura 11 - Exemplo de tela de cadastro da lista de compras



Fonte: Autoria própria

d) Após clicar no botão “Analisar”, será apresentada a página de relatório com os estabelecimentos que apresentam o melhor custo benefício, como apresentado na Figura 12:

Figura 12 - Exemplo relatório de custo benefício

Fonte: Autoria própria

5 CONCLUSÃO

Neste capítulo são apresentadas as observações finais referentes ao projeto, expondo algumas das dificuldades encontradas durante o desenvolvimento, assim como as possibilidades de trabalhos futuros.

5.1 CONSIDERAÇÕES FINAIS

O objetivo principal deste projeto foi auxiliar no processo de compras em supermercados de maneira a reduzir os custos da atividade total, levando em consideração a disponibilidade dos produtos e o valor total da lista de compras. Um processo demorado para quem não dispõe de tanto tempo livre para obter todos os dados dos produtos desejados e somar manualmente os preços dos mesmos. Processo este que poderia ser simplificado utilizando os recursos adequados.

As ferramentas que foram utilizadas neste projeto são de código aberto e livres, desta forma, foi possível realizar uma economia de recursos e agilizar o processo de desenvolvimento da aplicação.

Estas ferramentas foram escolhidas de acordo com o desempenho, curva de aprendizagem e facilidade para escalabilidade da aplicação.

Em consideração ao MEVN *stack* é possível afirmar que possui uma comunidade abrangente se comparado a outros conjuntos de ferramentas como MEAN *stack* (MongoDB, Express, Angular e NodeJS) e MERN *stack* (MongoDB, Express, React e NodeJS), sendo assim, possível encontrar diversos conteúdos em fóruns pela Internet e também nos próprios sites das ferramentas.

A proposta inicial deste trabalho incluiu a integração dos dados dos produtos dos estabelecimentos para que assim fosse possível que a informação disponível dos produtos na aplicação não ficasse defasada. Tal questão demandaria contratos com os estabelecimentos, além de envolvimento de acompanhamento jurídico, o que acarretaria em um custo elevado para a validação da ferramenta. Porém, a aplicação desenvolvida possui uma estrutura para a integração com as informações proveniente dos estabelecimentos de venda, o que permite que trabalhos futuros

usem o cadastro de produtos por parte dos estabelecimentos, garantindo uma base de dados fiel à realidade do mercado.

5.2 DIFICULDADES ENCONTRADAS

Dentro das dificuldades encontradas pode-se citar a questão da grande frequência de atualização das ferramentas utilizadas. Consequentemente, ao buscar as informações necessárias das ferramentas escolhidas foram encontradas soluções para versões diferentes, o que tornou a busca das informações mais demorada.

5.3 TRABALHOS FUTUROS

Durante o desenvolvimento do código fonte deste sistema, o mesmo já foi desenvolvido levando em consideração escalabilidade e adição de novos recursos. Esta modelagem leva em consideração o uso de API’s do servidor para buscar os dados. Portanto qualquer comunicação com os dados seja cadastro de produtos, ou geração de relatório é possível utilizando qualquer linguagem de programação que possua bibliotecas para realizar requisições HTTP. Novos recursos que podem ser adicionados futuramente:

- Integração dos dados do banco junto aos sistemas dos supermercados;
- Desenvolvimento de um aplicativo móvel aproveitando o lado servidor já desenvolvido.

REFERÊNCIAS

ALVERMANN, Markus. **Einführung in MongoDB.** Javaspektrum, 1 ed. 2011. Disponível em: <https://sigs-datacom.de/uploads/tx_dmjournals/alvermann_JS_01_11.pdf>. Acesso em: 8 out. 2019.

AMAZON RDS. **Relational database**. 2019. Disponível em: <https://aws.amazon.com/pt/relational-database/>. Acesso em: 7 out. 2019.

BASSIL, Youssef. **A simulation model for the waterfall software development life cycle.** International Journal of Engineering & Technologu (iJET). Vol. 2, No. 5, 2012. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1205/1205.6904.pdf>. Acesso em: 29 jun. 2020.

BENNETT, Jonathan. **OpenStreetMap.** Birmingham: Packt Publishing Ltd, 2010.

BERVERLY, Sondra G.; BURKHALTER, Emily K. **Improving the Financial Literacy and Practices of Youths**. Children & Schools, Vol. 27. n. 2, Abr/2005.

BRASIL/ENEF. **Estratégia Nacional de Educação Financeira:** plano diretor da ENEF. 2011. Disponível em <http://www.vidaedinheiro.gov.br/legislação/Default.aspx.> Acesso em: 5 out. 2019.

BRASIL/ENEF. **Estratégia Nacional de Educação Financeira**: parcerias e patrocínios. 2018. Disponível em: < http://www.vidaedinheiro.gov.br/parcerias-e-patrocinios/para-adultos/>. Acesso em: 5 out. 2019.

BRAUNSTEIN, S.; WELCH, C. **Financial literacy**: an overview of practice, research, and policy. Federal Reserve Bulletin, Nov. 2002.

BROWN, Ethan. **Web development with Node & Express.** 2 ed. Sebastopol: O'Reilly Media Inc, 2020.

CHODOROW, Kristina. **MongoDB:** the definitive guide. Sebastopol: O'Reilly. Mai. 2013. Disponível em: < https://books.google.com.br/books?hl=pt-BR&lr=&id=uGUKiNkKRJ0C&oi=fnd&pg=PP1&dq=why+choose+mongodb&ots=h9ptOecSwc&sig=u1pASadKG21KHulywPZOC2gd1WU&redir_esc=y#v=onepage&q=why%20choose%20mongodb&f=false>. Acesso em: 8 out. 2019.

CORONA, Erika; PANI, Filippo Eros. **A review of Lean-Kanban approaches in the software development.** Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari. 2013. Disponível em: < https://pdfs.semanticscholar.org/0725/482b6ced393863440f7e063c268e3790d18c.pdf>. Acesso em: 29 jun. 2020.

FIELDING, Roy T. **Architectural styles and the design of network-based software architectures.** 2000. 180 f. Dissertação (Doutorado) – University of California. Irvine, 2000.

FIELDING, Roy T.; TAYLOR, Richard N. **Principled design of the modern web architecture.** ACM Trans. Inter Tech. 2(2): 115-120, 2002.

FILIPOVA, Olga. **Learning Vue.js 2.** Birmingham: Packt. Dec. 2016. Disponível em: < https://books.google.com.br/books?hl=pt-BR&lr=&id=nszcDgAAQBAJ&oi=fnd&pg=PP1&dq=vue+js&ots=9nLbGhOlnO&sig=0SH16uCr0jn8iWLvgMRlHBwIRIA&redir_esc=y#v=onepage&q=vue%20js&f=false>. Acesso em: 8 out. 2019.

GROS, Denise Barbosa. **Institutos liberais e neoliberalismo no Brasil da Nova República**. 2002. 235 f. Dissertação (Doutorado) – Universidade Estadual de Campinas. Campinas, 2002.

GS1 Brasil. **GTIN:** número global do item comercial. Disponível em: <https://gs1br.org/codigos-e-padroes/chaves-de-identificacao/gtin>. Acesso em: 18 nov. 2020.

HAHN, Evan M. **Express in action:** writing, building, and testing Node.js applications. Shelter Island: Manning. Disponível em: <https://www.hackerstribe.com/wp-content/uploads/2016/04/Node.js-Express-in-Action.pdf>. Acesso em: 9 out. 20019.

JACOB, Katy et al. **Tools for survival**: An analysis of financial literacy programs follower income families. Chicago: Woodstok Institute, Jan/2000.

JAKL, Michael. **REST:** representational state transfer. University of Technology Vienna, 2002. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=77CBDA0C3CF66C5634C3C0295D553F20?doi=10.1.1.97.7334&rep=rep1&type=pdf>. Acesso em: 10 out. 2019.

LIBUV. **Welcome to the libuv documentation.** Disponível em: <http://docs.libuv.org/>. Acesso em: 11 jul. 2020.

LÓSCIO, B. F.; PONTES, J. C. S.; OLIVEIRA, H. R. **NoSQL no desenvolvimento de aplicações Web colaborativas.** VIII Simpósio Brasileiro de Sistemas Colaborativos. 2011.

MANKIW, N. G. **Introdução à economia.** 6ª ed. São Paulo. Cencage Learning, 2015.

MENOR PREÇO. **Menor Preço Nota Paraná.** Disponível em: <https://menorpreco.notaparana.pr.gov.br/index>. Acesso em: 3 fev. 2020.

MINISTÉRIO DA EDUCAÇÃO. **Base Nacional Comum Curricular.** Disponível em: <http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf/>. Acesso em: 30 jul. 2019.

MINISTÉRIO DA FAZENDA. **Ações gratuitas de educação financeira duplicaram em todo o país.** 14 mai. 2018. Disponível em: <http://www.fazenda.gov.br/noticias/2018/maio/acoes-gratuitas-de-educacao-financeira-duplicaram-em-todo-o-pais>. Acesso em: 30 jul. 2019.

MOZILLA. **Modelo de Objeto de Documento (DOM).** Disponível em: <https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM>. Acesso em: 18 nov. 2020.

NAYAK, Ameya; POOJARY, Dikshay; PORIYA, Anil. **Type of NOSQL databases and its comparions with relational databases.** International Journal of Applied Information Systems (IJAIS). Foundation of Computer Science FCS, New York. V. 5 N4. 2013. Disponível em: <https://www.researchgate.net/profile/Dikshay_Poojary/publication/302557703_Article_Type_of_nosql_databases_and_its_comparison_with_relational_databases/links/5aeaa2b50f7e9b837d3c40e7/Article-Type-of-nosql-databases-and-its-comparison-with-relational-databases.pdf>. Acesso em: 29 jun. 2020.

NODE.JS FOUNDATION. **NODE.JS Evented I/O for V8 JavaScript**. Disponível em: <http://nodejs.org>. Acesso em: 5 out. 2019.

NODESOURCE. **Node.JS usage and adoption statistics 2018**. 30 mar. 2018. Disponível em: <https://nodesource.com/node-by-numbers>. Acesso em: 7 out. 2019.

NPM, Inc. **Npm.js.** Disponível em: <https://docs.npmjs.com/>. Acesso em: 27 jun. 2020.

NTNU (Norwegian University of Science and Technology). Advanced Process Simulation. **SQL vs NoSQL**. 08 dez. 2014. Disponível em: <http://folk.ntnu.no/preisig/HAP_Specials/AdvancedSimulation_files/2014/AdvSim-2014__Birgen_Cansu_Databases.pdf>. Acesso em: 8 out. 2017.

OHNO, Taiichi. **Das Toyota-Produktionssystem.** 3 ed. Campus Verlag Gmbh, 2013.

ORGANIZATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT. **Improving Financial Literacy:** Analysis of issues and policies. Paris, 2005.

ORGANIZATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT. **Brazil Economic Snapshot**. Disponível em: < http://www.oecd.org/economy/brazil-economic-snapshot/>. Acesso em: 5 out. 2019.

PADIA, Keval. **Is Node.js really better than PHP? Let's go through the pros and cons.** Disponível em: < https://datafloq.com/read/is-nodejs-better-than-php-pros-cons/5297>. Acesso em: 11 jul. 2020.

PINNGO. **Compare preços no Pinngo: monte sua lista e descubra os preços.** Disponível em: <http://www.pinngo.com.br/>. Acesso em: 5 fev 2020.

PROCON Maringá. **Produtos de Supermercado**. 1 jul. 2017. Disponível em: <*http://www2.maringa.pr.gov.br/sistema/arquivos/0a088a46fd8e.pdf*>. Acesso em: 8 jul 2019.

PROTESTE. **Supermercados:** diferença de preços pode chegar a 168%. 2 out. 2015. Disponível em: <*https://www.proteste.org.br/suas-contas/supermercado/noticia/supermercados-diferenca-de-precos-pode-chegar-a-168*>. Acesso em: 8 jul. 2019.

RIES, Eric. Minimum Viable Product: a guide. 2009. Disponível em: <startuplessonslearned.com/2009/08/minimum-viable-product-guide.html>. Acesso em: 30 set. 2020.

ROYCE, Winston Walker. **Managing the development of large software systems.** Disponível em: <http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf>. Acesso em: 11 jul. 2020.

SAVOIA, José Roberto Ferreira; SAITO, André Taue; SANTANA, Flávia de Angelis. **Paradigmas da educação financeira no Brasil.** Rev. Adm. Pública, Rio de Janeiro (RJ), v. 41, n. 6, p. 1121-1141, dez. 2007.

SINGH, Inderpreet. **How to mash up BLE, NodeJS, and MQTT to get internet of things**. 24 aug. 2018. Disponível em: <https://hackaday.com/2018/08/24/how-to-mash-up-ble-nodejs-and-mqtt-to-get-internet-of-things/>. Acesso em: 27 jun. 2020.

SPC BRASIL. **Com economia desfavorável, brasileiro muda hábitos de consumo e passa a pesquisar mais preço, apontam CNDL/SPC Brasil e Banco**

Central. 5 fev. 2019. Disponível em: <https://www.spcbrasil.org.br/pesquisas/pesquisa/5907>. Acesso em: 8 jul. 2019.

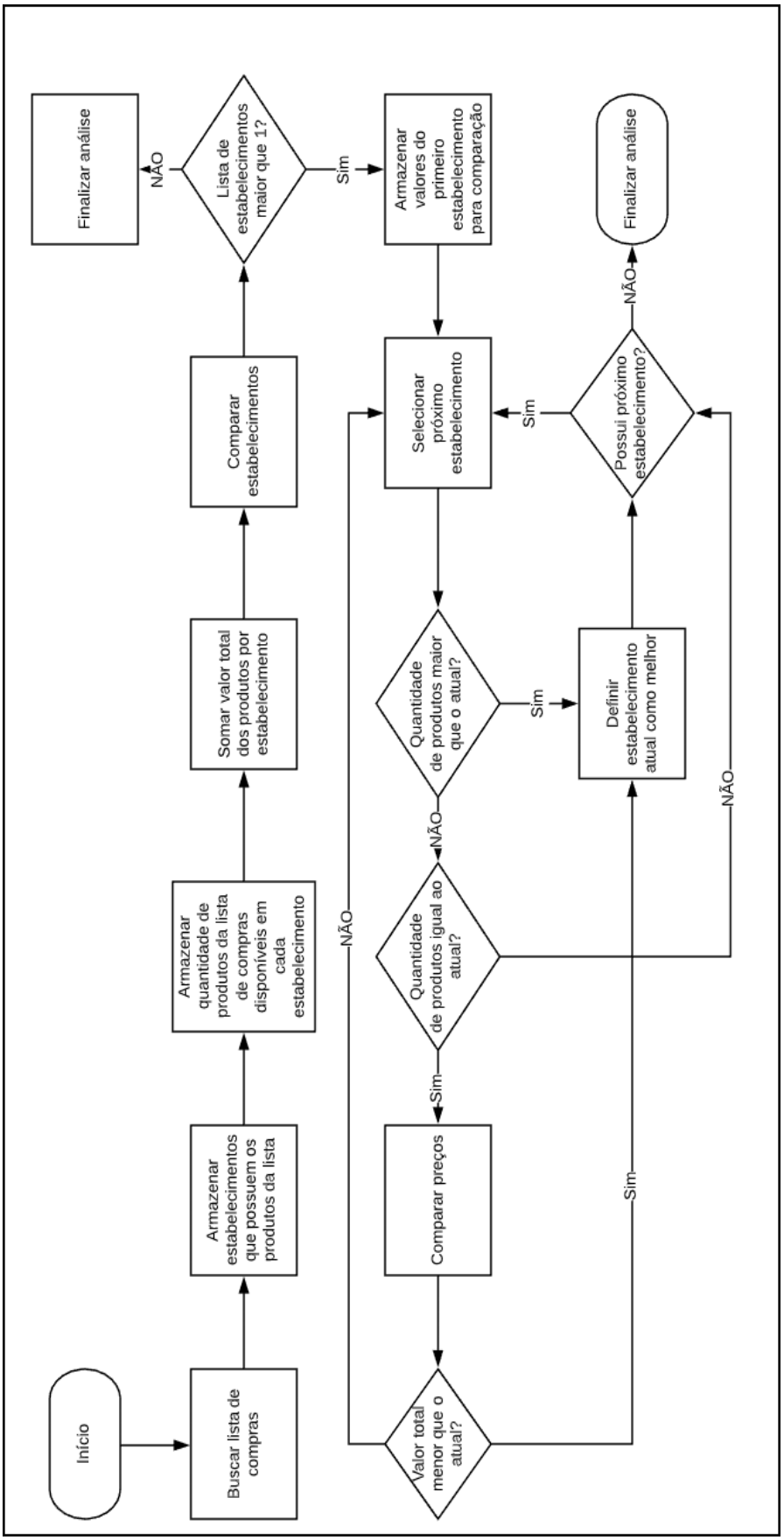
SPURLOCK, Jake. **Bootstrap.** Sebastopol: O'Reilly. May. 2013. Disponível em: < https://books.google.com.br/books?hl=pt-BR&lr=&id=LZm7Cxgi3aQC&oi=fnd&pg=PR2&dq=bootstrap+css&ots=eW2zzAGnJE&sig=lE360rXEQblCIhn4sfTrM29-LAY&redir_esc=y#v=onepage&q=bootstrap%20css&f=false>. Acesso em: 9 out. 2019.

TEIXEIRA, Pedro. **Professional Node.js**: building javascript based scalable software. Indianapolis: John Wiley and Sons, Inc, 2012.

VARIAN, Hal R. **Microeconomia:** uma abordagem moderna. 8ª edição. Rio de Janeiro: Elsevier, 2012.

VUE.JS. **Comparison with other frameworks.** Disponível em: <https://vuejs.org/v2/guide/comparison.html>. Acesso em: 8 out. 2019.

APÊNDICE A - FLUXOGRAMA DA ANÁLISE DE MELHOR ESTABELECIMENTO PARA DETERMINADA LISTA DE COMPRAS


Início
Buscar lista de compras
Armazenar estabelecimentos que possuem os produtos da lista
Armazenar quantidade de produtos da lista de compras disponíveis em cada estabelecimento
Somar valor total dos produtos por estabelecimento
Comparar estabelecimentos
Lista de estabelecimentos maior que 1?
NÃO
Finalizar análise
Sim
Armazenar valores do primeiro estabelecimento para comparação
Selecionar próximo estabelecimento
Quantidade de produtos maior que o atual?
Sim
Definir estabelecimento atual como melhor
NÃO
Quantidade de produtos igual ao atual?
NÃO
Sim
Comparar preços
Valor total menor que o atual?
NÃO
Sim
Possui próximo estabelecimento?
Sim
NÃO
Finalizar análise