INSTITUTO FEDERAL DO ESPÍRITO SANTO
CURSO SUPERIOR DE SISTEMAS DE INFORMAÇÃO

EDUARDO ALVES FIGUEIREDO

CONSTRUÇÃO DE UMA BASE DE DADOS PARA PESQUISA DE PREÇOS NA ÁREA VAREJISTA DE SUPERMERCADOS E HIPERMERCADOS DO ESTADO DO ESPÍRITO SANTO

Serra
2023

EDUARDO ALVES FIGUEIREDO

CONSTRUÇÃO DE UMA BASE DE DADOS PARA PESQUISA DE PREÇOS NA ÁREA VAREJISTA DE SUPERMERCADOS E HIPERMERCADOS DO ESTADO DO ESPÍRITO SANTO

Trabalho de Conclusão de Curso apresentado à Coordenadoria do Curso de Sistemas de Informação do Instituto Federal do Espírito Santo, Campus Serra, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Daniel Ribeiro Trindade

Serra
2023

Dados Internacionais de Catalogação na Publicação (CIP)

F475c 2023

Figueiredo, Eduardo Alves
Construção de uma base de dados para pesquisa de preços na área varejista de supermercados e hipermercados do estado do Espírito Santo / Eduardo Alves Figueiredo. - 2023.
54 f.; il.; 30 cm

Orientador: Prof. Me. Daniel Ribeiro Trindade.
Monografia (graduação) - Instituto Federal do Espírito Santo, Coordenadoria de Informática, Curso de Bacharelado em Sistemas de Informação, 2023.

1. Banco de dados. 2. Interface de programas aplicativos (Software). 3. Supermercados. 4. Comércio varejista - Espírito Santo (Estado). 5. Notas fiscais eletrônicas. I. Trindade, Daniel Ribeiro. II. Instituto Federal do Espírito Santo. III. Título.

CDD 005.74

Bibliotecária Rogeria Gomes Belchior - CRB6/ES 417

EDUARDO ALVES FIGUEIREDO

CONSTRUÇÃO DE UMA BASE DE DADOS PARA PESQUISA DE PREÇOS NA ÁREA VAREJISTA DE SUPERMERCADOS E HIPERMERCADOS DO ESTADO DO ESPÍRITO SANTO

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de Bacharel em Sistemas de Informação, do curso de Bacharelado em Sistemas de Informação do Instituto Federal do Espírito Santo.

Aprovado em 12 de dezembro de 2023.

COMISSÃO EXAMINADORA

_____

Prof. Me. Daniel Ribeiro Trindade (Orientador)
Instituto Federal do Espírito Santo - Campus Serra

_____

Prof. Dr. Francisco de Assis Boldt
Instituto Federal do Espírito Santo - Campus Serra

_____

Prof. Dr. Vitor Faiçal Campana
Instituto Federal do Espírito Santo - Campus Serra

Emitido em 13/12/2023

FOLHA DE APROVAÇÃO-TCC Nº 2/2023 - SER - CCTMSI (11.02.32.01.08.02.09)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 14/12/2023 00:00)
DANIEL RIBEIRO TRINDADE
PROFESSOR DO ENSINO BASICO TECNICO E TECNOLOGICO
SER - CCTMSI (11.02.32.01.08.02.09)
Matrícula: 2277933

(Assinado digitalmente em 14/12/2023 13:15)
FRANCISCO DE ASSIS BOLDT
PROFESSOR DO ENSINO BASICO TECNICO E TECNOLOGICO
CMPCA (11.02.32.01.07.08)
Matrícula: 1304946

(Assinado digitalmente em 14/12/2023 09:31)
VITOR FAICAL CAMPANA
PROFESSOR DO ENSINO BASICO TECNICO E TECNOLOGICO
SER-CGEN (11.02.32.01.08.02)
Matrícula: 2383546

Visualize o documento original em https://sipac.ifes.edu.br/documentos/ informando seu número: **2**, ano: **2023**, tipo: **FOLHA DE APROVAÇÃO-TCC**, data de emissão: **14/12/2023** e o código de verificação: **6f51c619e5**

À Deus
À minha mãe
Ao meu pai
À minha irmã
À todos aqueles que me acompanharam nessa jornada

AGRADECIMENTOS

Agradeço a Deus por me dar força e perseverança para continuar mesmo com as adversidades que surgiram pelo caminho.

Agradeço a minha família por todo apoio e carinho que me deram e por terem me motivado nessa caminhada.

Agradeço ao meu orientador, Prof. Me. Daniel Ribeiro Trindade por toda a paciência, esforço, auxílio e disponibilidade.

Por fim, agradeço a todos os amigos e professores que estiveram presentes durante toda a minha jornada acadêmica.

RESUMO

O comércio varejista é um setor relevante para a economia brasileira pela geração de receitas, oferta de empregos e no atendimento das demandas de consumo das famílias. O comércio varejista abrange o setor de Hipermercados e Supermercados, no qual este será utilizado como instrumento de estudo neste trabalho. Este setor provê as necessidades básicas e diárias do consumidor. É possível observar uma variação diária dos preços dos produtos ofertados pelos Hipermercados e Supermercados. Levando em consideração esta variação de preços, foi proposta uma solução para criar uma aplicação que forneceria de maneira simples e eficaz os preços atuais dos produtos nos supermercados mais próximos da pessoa que procura. O objetivo deste trabalho é a construção de uma base de dados que armazene o histórico de preços de produtos, os preços destes produtos, as informações dos produtos e as informações dos supermercados advindos de uma nota fiscal eletrônica. As informações dos produtos e dos supermercados são obtidos de forma colaborativa através dos usuários da aplicação cliente. Foram utilizadas técnicas e metodologias de análise de requisitos, construção de diagramas de entidade-relacionamento e padrões de projeto. Ao final do projeto, foi possível afirmar que os requisitos levantados durante o processo de análise foram desenvolvidos com sucesso e as ferramentas escolhidas foram eficientes. Como resultado final foi criada uma API pública, onde esta realiza a consulta e gravação das informações referentes aos produtos, os preços destes produtos e as informações dos supermercados.

Palavras-chave: API pública. base de dados. Hipermercados e Supermercados. Nota fiscal eletrônica.

ABSTRACT

Retail trade is a relevant sector for the Brazilian economy for generating revenue, offering jobs and meeting family consumption demands. Retail trade covers the Hypermarkets and Supermarkets sector, which will be used as a study instrument in this work. This sector provides the consumer with basic and daily needs. It is possible to observe a daily variation in the prices of products offered by Hypermarkets and Supermarkets. Taking this price variation into account, a solution was proposed to create an application that would simply and effectively provide the current prices of products in the supermarkets closest to the person searching. The objective of this work is to build a database that stores the history of product prices, the prices of these products, product information and supermarket information from an electronic invoice. Product and supermarket information is obtained collaboratively by users of the client application. Requirements analysis techniques and methodologies, construction of entity-relationship diagrams and design patterns were used. At the end of the project, it was possible to affirm that the requirements raised during the analysis process were successfully developed and the tools chosen were efficient. As a final result, a public API was created, where it queries and records information regarding products, the prices of these products and supermarket information.

Keywords: Public API. data base. Hypermarkets and Supermarkets. Electronic invoice.

LISTA DE FIGURAS

Figura 1 – Uma nota fiscal do município de Vitória do Supermercado Extraplus . 13
Figura 2 – Exemplo de um diagrama de entidade-relacionamento 17
Figura 3 – Exemplo da arquitetura cliente-servidor 18
Figura 4 – Exemplo de requisição HTTP 19
Figura 5 – Exemplo do Entity Framework 20
Figura 6 – Exemplo de chamada para API REST 21
Figura 7 – Tela principal do aplicativo ByPantry 23
Figura 8 – Visão Geral da aplicação 25
Figura 9 – Exemplo de um diagrama de entidade-relacionamento 29
Figura 10 – Criar um novo projeto no Visual Studio 32
Figura 11 – Estrutura do modelo de projeto API Web do ASP.NET Core 32
Figura 12 – Estrutura e sintaxe do código da linguagem de programação 33
Figura 13 – Código em C# que apresenta a forma de realizar uma requisição HTTP com *RestSharp* 34
Figura 14 – Página *Web* da nota fiscal de consumidor eletrônica 35
Figura 15 – Página *Web* da nota fiscal de consumidor eletrônica com dados gerais . 36
Figura 16 – Código em C# que apresenta a forma de realizar uma requisição HTTP usando o *Selenium* 36

SUMÁRIO

1 INTRODUÇÃO 11
1.1 OBJETIVOS 14
1.1.1 Objetivo Geral 14
1.1.2 Objetivo Específicos 14
1.2 ESTRUTURA DO TEXTO 14
2 REFERENCIAL TEÓRICO 16
2.1 ENGENHARIA DE SOFTWARE 16
2.1.1 Requisitos Funcionais 16
2.1.2 Requisitos Não Funcionais 16
2.1.3 Diagramas de Entidade-Relacionamento 16
2.2 DESENVOLVIMENTO WEB 17
2.2.1 Arquitetura Cliente-Servidor 17
2.2.2 Http 18
2.2.3 Front-End 19
2.2.4 Back-End 19
2.2.4.1 Api Web Asp.Net Core 20
2.2.4.2 Entity Framework 20
2.2.4.3 Api Rest 20
2.3 WEB CRAWLER 21
2.3.1 Selenium 21
2.4 TRABALHOS CORRELATOS 22
3 DESENVOLVIMENTO 25
3.1 VISÃO GERAL 25
3.2 REQUISITOS FUNCIONAIS 27
3.3 REQUISITOS NÃO FUNCIONAIS 28
3.4 DIAGRAMA DE ENTIDADE-RELACIONAMENTO 29
3.5 TECNOLOGIAS E BIBLIOTECAS UTILIZADAS 29
3.6 DESENVOLVIMENTO DA API DE COMUNICAÇÃO 31
3.6.1 Construção e Estrutura da Api 31
3.7 OBTENÇÃO DOS DADOS DA NOTA FISCAL 33
4 RESULTADOS 38
4.1 DOCUMENTAÇÃO DOS MÉTODOS DA API 38
4.1.1 Métodos de Envios (Http Post) 38
4.1.1.1 LinkNotaFiscal 38
4.1.1.2 ProdutosEscolhidosCarrinho 39
4.1.2 Métodos de Consultas (Http Get) 40
4.1.2.1 SupermercadosProximos 40
4.1.2.2 ProdutosCategoriaSupermercados 41

4.1.2.3 ProdutosCategoria 41
4.1.2.4 HistoricoPrecoGeral 42
4.1.2.5 CategoriasProdutos 42
4.1.2.6 SupermercadosProduto 43
4.1.2.7 HistoricoPrecoSupermercado 44
4.1.2.8 RelatorioInfoMetodo 44
4.1.2.9 RelatorioNotaFiscal 45
4.2 ANÁLISE DO COMPORTAMENTO DOS MÉTODOS DA API 46
4.2.1 Método de LinkNotaFiscal 46
4.2.2 Método de SupermercadosProximos 47
4.2.3 Método de ProdutosCategoriaSupermercados 48
4.2.4 Método de HistoricoPrecoSupermercado 49
4.2.5 Método de SupermercadosProduto 50
5 CONSIDERAÇÕES FINAIS 51
5.1 CONCLUSÃO 51
5.2 TRABALHOS FUTUROS 51
REFERÊNCIAS 53

1 INTRODUÇÃO

O comércio varejista é um setor relevante para a economia brasileira pela geração de receitas, oferta de empregos e no atendimento das demandas de consumo das famílias, como é descrito em Pauli (2019). Esta afirmação fica enfatizada ao se analisar o volume de vendas do comércio varejista. A Tabela 1 apresenta indicadores de volume de vendas do comércio varejista e comércio atacadista a partir do mês de dezembro de 2022, onde a primeira coluna apresenta uma divisão por tipo de produto (Atividades de Divulgação), a segunda, a variação com relação ao mês anterior (1) e a terceira, as variações mensais (2) (PMC. . . , 2023). Na quarta linha, item 2.1 Hipermercados e Supermercados e coluna Mensal (2), apresenta valores percentuais positivos de 3,2% em dezembro de 2022, 2,6% em janeiro de 2023 e 1,8% em fevereiro de 2023.

Tabela 1 – Tabela pesquisa Mensal de Comércio

Atividades de Divulgação	Mês/Mês anterior (1)			Mensal (2)		
	DEZ	JAN	FEV	DEZ	JAN	FEV
Volume de vendas do comércio varejista (5)	**-2,8**	**3,8**	**-0,1**	**0,4**	**2,6**	**1,0**
1. Combustíveis e lubrificantes	-2,0	1,4	-0,3	23,8	26,7	19,7
2. Hipermercados, supermercados, produtos alimentícios bebidas e fumo	-0,8	1,3	-0,7	2,5	2,2	1,0
2.1. Hipermercados e supermercados	-0,8	1,9	-0,9	3,2	2,6	1,8
3. Tecidos, vestuário e calçados	-6,6	27,9	-6,3	-11,9	2,3	-9,2
4. Móveis e eletrodomésticos	-1,7	1,2	-1,7	0,3	3,4	-1,9
4.1. Móveis	-	-	-	-9,9	-6,8	-14,6
4.2. Eletrodomésticos	-	-	-	5,4	7,2	4,9
5. Artigos farmacêuticos, médicos, ortopédicos, de perfumaria e cosméticos	-0,5	-1,1	1,4	0,8	-7,6	-0,7
6. Livros, jornais, revistas e papelaria	-1,0	-4,3	4,7	0,3	15,2	-9,5
7. Equipamentos e materiais para escritório, informática e comunicação	-0,6	7,9	-10,4	0,1	14,8	-4,5
8. Outros artigos de uso pessoal e doméstico	-3,0	1,6	-2,0	-8,4	-6,5	-12,9
Volume de vendas do comércio varejista ampliado (6)	**0,8**	**0,2**	**1,7**	**-0,6**	**0,5**	**-0,2**
9. Veículos, motocicletas, partes e peças	2,4	0,0	1,4	-1,8	4,4	-1,5
10. Material de construção	1,9	2,8	-2,0	-7,1	1,1	-5,9
11. Atacado especializado em produtos alimentícios, bebidas e fumo	-	-	-	-	-0,9	-9,5

Fonte: PMC. . . (2023).

Levando em consideração as afirmações a cerca da Tabela 1 e a própria Tabela 1, é possível realizar um projeto que manipule as informações obtidas em relação aos preços de produtos dos supermercados. Isto porque as variações mensais do volume de vendas do setor de Hipermercados e Supermercados se encontram positivos em todos os meses analisados. Isto fica evidenciado na Tabela 1.

O setor de Hipermercados e Supermercados da área varejista é o setor que possui um destaque e uma alta relevância, pois provê as necessidades básicas e diárias do consumidor, como: produtos alimentícios, higiênicos e assim por diante. A ida frequente dos consumidores aos hipermercados e supermercados para realizarem suas compras, apenas reforça o motivo deste setor ser significativo. Entre os 53 serviços essenciais para a população, os supermercados se encontram entre estes serviços, mesmo nos momentos mais agudos da pandemia, de acordo com (PINHO, 2021).

Ao realizar uma pesquisa sobre o preço de um determinado produto, é possível perceber uma variação deste preço em diferentes supermercados, sendo esta variação alta ou não. Como existe a oscilação de preços diária dos produtos oferecidos pelos supermercados, os consumidores podem não saber em qual supermercado mais próximo à eles possui o menor preço oferecido dos produtos que eles desejam comprar. Isto acaba acarretando na realização de compras dos produtos em um supermercado cujos valores estão mais elevados. Ocasionando na maioria das vezes, que o cliente saiba posteriormente que os produtos comprados por ele estavam com preços mais acessíveis em outros supermercados. Gerando ao cliente mais opções de compra. Como exemplo, considerando o valor da cesta básica com 20 produtos, o Procon de Maringá verificou uma diferença de até 294,8% nos preços dos itens (IORE, 2023).

Os preços nos supermercados do Espírito Santo podem variar até 202% , segundo reportagem de 2023 do site "Tribuna Online" (TOVAR, 2023). Isto faz com que os consumidores capixabas, em muitas das vezes, comprem produtos que possuem valores mais elevados em relação a outros supermercados. Isto se dá devido a alguns fatores, como exemplo: (i) A categorização dos supermercados e hipermercados, onde o fator da classe social dos consumidores onde o estabelecimento se encontra localizado influencia nos preços dos produtos, (ii) O poder de negociação que existe entre o fornecedor e o estabelecimento, determinando assim, um alto ou baixo poder de negociação quando leva-se em consideração o período de tempo atuante no mercado e o porte do estabelecimento e (iii) a lei da oferta e demanda, onde um produto pode ter o seu preço elevado pelo fato dos clientes o buscarem além do esperado ou, diminuído pelo fato do produto não ter a procura como deveria. Sendo assim, o conhecimento dos preços exercidos pelos supermercados próximos, seria uma informação fundamental a ser usada na tomada de decisão de compra do consumidor, o auxiliando no planejamento da compra antes de sair de casa ou mesmo no momento de realização de sua compra dentro do supermercado.

Existem organizações que estabelecem e gerenciam padrões específicos para a realização de uma técnica ou método. Quando outras organizações desejam utilizar estas técnicas ou métodos, estas precisam seguir estes padrões já estabelecidos. O código de barras, que é uma sequência de números representada por várias barras com diferentes espessura e posicionamento (LEãO, 2022), é um retrato deste uso. Sendo a GS1 Brasil a empresa oficial responsável por fornecer códigos de barras no Brasil e no mundo (REIS, 2022). Para que o registro de um código de barras seja feito, é necessário procurar a GS1 e realizar o cadastro da empresa, enviar a documentação necessária e efetuar o pagamento de um boleto (GS1, 2018). O código de barras é um sistema que auxilia e facilita a utilização dos produtos, pois ele permite que os produtos sejam comercializados usando a mesma identificação em todos os Estados do Brasil.

A proposta deste trabalho é o desenvolvimento de uma base de dados de domínio público, onde irá conter as informações dos preços dos produtos ao longo do tempo em diferentes supermercados, podendo ser utilizada tanto por entidades públicas quanto por entidades privadas. A forma pela qual esta base de dados será alimentada será através da colaboração dos consumidores do estado do Espírito Santo para a leitura de um *QR code* de uma nota fiscal, tal como na Figura 1. No Espírito Santo existe uma padronização da emissão da nota fiscal, onde contém um *QR code* que retorna um *link* para uma página HTML da SEFAZ (Secretária da Fazenda Estadual) com um espelho digital da nota fiscal, com a descrição dos itens comprados, seu valor unitário, a quantidade comprada e o valor total da compra, como é apresentado na Figura 1.

Figura 1 – Uma nota fiscal do município de Vitória do Supermercado Extraplus

Fonte: elaborado pelo próprio autor (2023)

O estudo de caso deste trabalho é a coleta de dados via notas fiscais dos supermercados de Vitória/ES, como, Supermercados Carone, Perim, Epa e Extraplus. O período de coleta é de 01 de outubro de 2023 até 31 de outubro de 2023, através de compras de conhecidos próximos dos membros da equipe e pelos próprios membros da equipe. Um cuidado a ser tomado na coleta de dados é que o mesmo item exista em mais de um supermercado e em tempos diferentes, para se apresentar a variação do preço no tempo.

O objetivo do trabalho é a construção de uma base de dados que armazene o histórico de preços de produtos relacionados a área varejista de supermercados e hipermercados. Esta base de dados deve ser alimentada de forma colaborativa pelos clientes dos supermercados

da região do estado do Espírito Santo. De modo que esta base de dados seja formada ao longo do tempo de forma colaborativa. Sendo o objetivo da base de dados permitir a pesquisa da variação de preços de um determinado produto na rede de varejo para supermercados e hipermercados do Espírito Santo. Juntamente com a criação de uma API (Interface de programação de aplicações) pública que permita que diferentes entes, tanto públicos como privados, sejam capazes de realizar a pesquisa de preços e histórico de preços de um determinado produto. Uma vez estabelecida a base de dados e a API pública, será possível que os entes públicos sejam capazes de definir mais facilmente suas métricas, como, a inflação dos produtos e por fim auxiliando no projeto de políticas públicas. Além disso, é possível que os consumidores sejam capazes de realizar a pesquisa do histórico de preços dos produtos sem a necessidade de se locomover.

As tecnologias utilizadas para a implementação deste trabalho são: SQL Server, API WEB ASP.NET CORE, ENTITY FRAMEWORK e REST (Representational State Transfer). Sendo as linguagens: C# e SQL (Structure Query Language).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

A construção de uma base dados que troque informações com uma API pública fornecendo os preços e o histórico de preços de um determinado produto.

1.1.2 Objetivo Específicos

- Modelar e criar a base de dados para armazenar o histórico de preços de produtos no estado do Espírito Santo.
- Desenvolver o método para leitura de dados de notas fiscais através da página HTML da SEFAZ do Espírito Santo.
- Criar um site ou aplicativo que permita a realização de testes em relação a base de dados e o início da alimentação da base dados.
- Projetar e criar um conjunto de métodos de uma API (*Application Programming Interface*) que recebe e fornece dados a cerca do projeto.

1.2 ESTRUTURA DO TEXTO

Os próximos capítulos estão organizados da seguinte forma:

- Capítulo 2 – Referencial Teórico: apresenta os principais conceitos e tecnologias utilizados.

- Capítulo 3 – Desenvolvimento: descreve como o sistema foi desenvolvido para alcançar os objetivos propostos para resolução do problema

- Capítulo 4 – Resultados: apresenta e discute os resultados alcançados com o desenvolvimento do sistema

- Capítulo 5 – Considerações Finais: realiza uma síntese do trabalho e cita alguns possíveis projetos futuros que podem ser desenvolvidos com base neste.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados todas as ferramentas e bibliotecas que serão utilizadas para o desenvolvimento do projeto.

2.1 ENGENHARIA DE SOFTWARE

A engenharia de software é uma área da engenharia e também da computação que tem por objetivo a realização da especificação, desenvolvimento e manutenção de software, utilizando as tecnologias necessárias e as práticas de gerência de projeto para alcançar organização, produtividade e qualidade do software (ENGENHARIA..., 2023). Ao iniciar o desenvolvimento de software é preciso definir as especificações, que seriam deixar documentado quais as funções que o software terá (requisitos funcionais) e quais as suas restrições (requisitos não funcionais). Ademais, existe outro diagrama que é comumente utilizado e auxilia no momento do desenvolvimento e que faz parte da especificação, sendo este o diagrama de entidade-relacionamento. As tecnologias e as práticas de gerência citadas anteriormente abrangem a linguagem de programação, o banco de dados, as ferramentas e as bibliotecas que serão utilizadas no projeto.

2.1.1 Requisitos Funcionais

Um requisito funcional é uma declaração de como um sistema ou aplicação deve se comportar, ou seja, a definição do que o sistema ou aplicação irá fazer. Geralmente é definido um conjunto de requisitos funcionais, indicando todas as funcionalidades que será provido. Exemplo de um requisito funcional: O sistema deve permitir a inserção de um pedido de um usuário.

2.1.2 Requisitos Não Funcionais

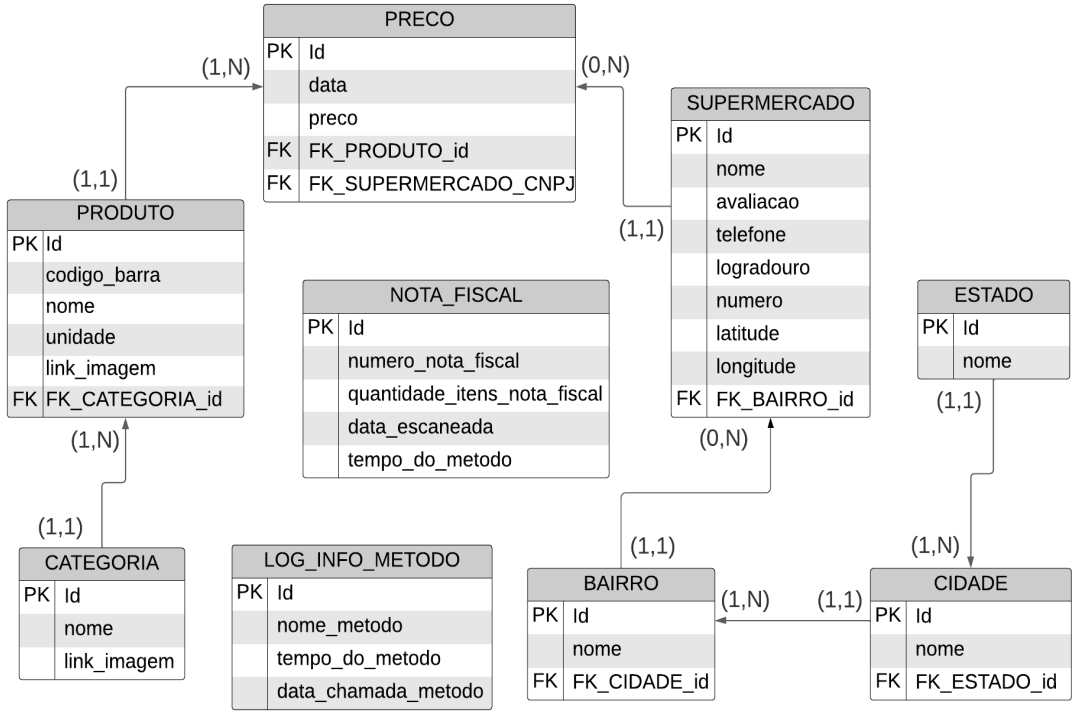
Um requisito não funcional declara uma restrição que um sistema ou aplicação contém. Além de definirem questões relacionadas ao uso da aplicação ou sistema em termos de desempenho, usabilidade, confiabilidade, segurança e assim por diante. Geralmente é definido um conjunto de requisitos não funcionais, indicando todas as restrições que terão no sistema ou aplicação. Exemplo de um requisito não funcional: O sistema será desenvolvido utilizando C# como linguagem de programação.

2.1.3 Diagramas de Entidade-Relacionamento

Um diagrama de entidade-relacionamento é um tipo de fluxograma que ilustra como "entidades", por exemplo, pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema (LUCIDCHART, 2023). Um diagrama de entidade-relacionamento orienta

e auxilia o desenvolvedor no momento de desenvolvimento do modelo físico de banco de dados. A Figura 9 apresenta um exemplo de um diagrama entidade-relacionamento.

Figura 2 – Exemplo de um diagrama de entidade-relacionamento



Fonte: elaborado pelo próprio autor (2023)

Os termos “PK” e “FK” que aparecem na Figura 9 significam respectivamente, a chave primária e a chave estrangeira. Estas chaves são necessárias para acontecer a relação entre as tabelas. Os números entre parêntesis indicam a cardinalidade, ou seja, o número de ocorrências de uma entidade que está associado com ocorrências de outra entidade. Existem três tipos de relacionamentos, sendo eles: o de Um para Um, de Um para Muitos e o de Muitos para Muitos.

2.2 DESENVOLVIMENTO WEB

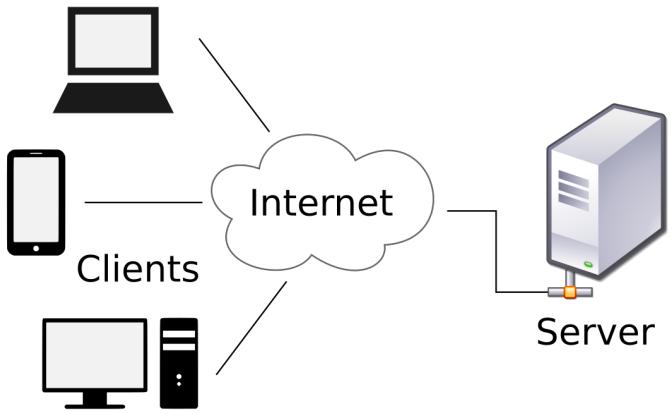
O desenvolvimento web se refere a construção e desenvolvimento de sites ou aplicações que são disponibilizadas na *internet* ou *intranet*. No desenvolvimento web existem basicamente três tipos de desenvolvimento, sendo eles: O *front-end*, voltado para o lado do cliente, o *back-end*, voltado para o lado do servidor e o *full-stack*, que seria realizar os dois tipos de desenvolvimento *front-end* e *back-end*. O projeto está voltado na utilização do desenvolvimento *back-end*.

2.2.1 Arquitetura Cliente-Servidor

A arquitetura cliente-servidor é uma arquitetura de aplicação distribuída, onde as máquinas exercem diferentes funções. A máquina servidora provê serviços que serão consumidos

pela máquina cliente. Um exemplo ilustrativo pode ser observado na Figura 3.

Figura 3 – Exemplo da arquitetura cliente-servidor



Fonte: Sampaio (2021)

O cliente (*Clients*) representado pelo *notebook*, *smartphone* ou computador *desktop* se conecta à internet, representada pela nuvem, e através dela realiza a requisição de um recurso que está disponível no servidor (*Server*), sendo este na maioria das vezes um computador *desktop* potente. O servidor devolve o recurso através de uma resposta também utilizando a internet. Ao final, o cliente obtém este recurso que requisitou.

2.2.2 Http

O protocolo HTTP (*Hypertext Transfer Protocol*) é um protocolo de comunicação utilizado para realizar transferências de arquivos hipermídia, como, textos, imagens, vídeos e áudio. Este protocolo é a base para a comunicação de arquivos na *internet*. O protocolo HTTP contém os verbos definidos para utilização, os verbos mais utilizados são o *GET*, *POST*, *PUT* e *DELETE*. O verbo *GET* faz a requisição de um arquivo ou documento para o servidor, o verbo *POST* envia um arquivo ou documento para o servidor, o verbo *PUT* realiza uma alteração no servidor e o verbo *DELETE* faz a exclusão de um arquivo ou documento no servidor. Os verbos mais utilizados no projeto são o *GET* e o *POST*. Um exemplo de uma requisição HTTP pode ser observado na Figura 4.

Figura 4 – Exemplo de requisição HTTP

Fonte: Jr (2020)

O método *POST* especifica que esta requisição irá enviar informações ao servidor. A URI informa a este servidor qual o caminho desta informação que o cliente requisitante deseja, ou seja, qual o recurso que este deseja obter. A versão especifica qual é a versão do protocolo HTTP que está sendo utilizada, no caso da Figura 4 é a versão 1.1. O cabeçalho é uma parte da requisição que informa informações adicionais, no caso da Figura 4, o *Host* está informando qual o endereço do servidor no qual se deseja realizar a requisição, o *Connection* indica que a conexão será duradoura, não irá ficar se desconectando, o *Content-type* indica qual será o formato da informação que será enviada. Já o corpo possui as informações que serão enviadas para o servidor.

2.2.3 Front-End

Front-end é a parte visual, onde são visualizados os arquivos hipermídia na máquina cliente da arquitetura cliente-servidor. Geralmente, estes arquivos hipermídia são formados por uma linguagem de marcação chamada HTML (*HyperText Markup Language*), onde são especificados ou adicionados os textos, imagens, vídeos e áudios. Continuando, o CSS (*Cascading Style Sheets*), que é um mecanismo para adição de estilos e o *JavaScript* que é uma linguagem de programação. Existem também *frameworks* que são utilizados para o desenvolvimento *front-end*, sendo os mais utilizados, *VUE.JS*, *React* e o *Angular*.

2.2.4 Back-End

Back-end é a parte lógica, onde são recebidos, processados e retornados as requisições que chegam no servidor. No *back-end* são utilizados a estrutura lógica para tratar as requisições utilizando uma linguagem de programação, banco de dados para armazenar informações a cerca da aplicação e um mecanismo de ligação da parte lógica com o banco de dados. As linguagens de programação mais utilizadas atualmente são: *Python*, *Rails*, *Java*, *C#* e assim por diante. Já para o banco de dados são: *Oracle*, *SQL Server*, *MySQL* e assim por diante. Para o projeto foram utilizados a linguagem de programação *C#*, o banco de dados *SQL Server* e o mecanismo de mapeamento objeto-relacional *Entity Framework*. Estas tecnologias serão detalhadas nas próximas seções.

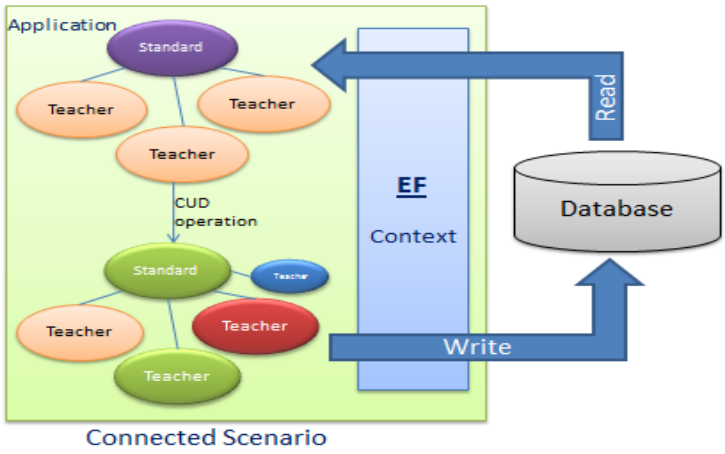
2.2.4.1 Api Web Asp.Net Core

O .NET é uma plataforma única desenvolvida pela *Microsoft* para desenvolvimento e execução de sistemas e aplicações. Já o ASP.NET também é uma plataforma para desenvolvimento de aplicações *WEB*. A escolha destas plataformas se dá devido ao uso e ao conhecimento já adquirido anteriormente pelo estudante.

2.2.4.2 Entity Framework

O *Entity Framework* é uma ferramenta presente na plataforma .NET de mapeamento de dados objeto-relacional e de persistência. O *Entity Framework* realiza a ponte entre o servidor da aplicação e o banco de dados, persistindo os dados no banco de dados no formato relacional e recuperando estes dados no formato de objeto para o servidor. Estas operações só são possíveis por que o próprio *Entity Framework* realiza estas transformações. A Figura 5 apresenta um exemplo do funcionamento da aplicação utilizando o *Entity Framework*, onde o *Application*, representado pelos módulos *Standard* e *Teacher* dentro do bloco verde, faz o uso do *Entity Framework* (EF) para realizar as operações de leitura (*Read*) ou gravação (*Write*) no *Database*.

Figura 5 – Exemplo do Entity Framework



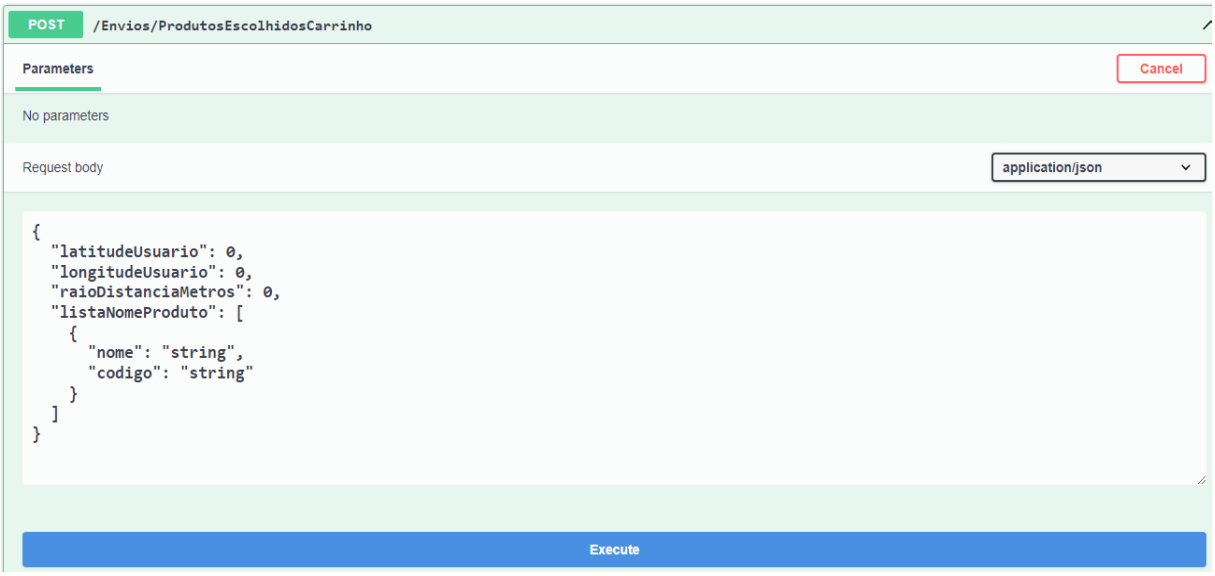
Fonte: Macoratti (2023)

2.2.4.3 Api Rest

API REST é uma interface de programação de aplicações que segue um conjunto de restrições do estilo de arquitetura REST, onde REST é uma sigla que significa *Representational State Transfer* (HAT, 2023). Uma API é composta por métodos que são consumidos por outra aplicação. Para realizar o consumo dos métodos de uma API é preciso utilizar o protocolo HTTP juntamente com o verbo. Como exemplo, para recuperar algum dado que um método da API fornece é preciso especificar o verbo *GET*. Para realizar a troca de dados entre as aplicações é recomendado a padronização do formato destes dados. O JSON (*JavaScript Object Notation*) é um formato compacto de troca de dados simples e

rápida entre sistemas diferentes. Geralmente, os sistemas possuem tecnologias e linguagens diferentes entre si, utilizando o JSON a troca de dados entre dois sistemas se torna mais simples e rápida. Na Figura 6 é possível observar uma chamada *POST* para uma API passando um JSON como corpo da requisição.

Figura 6 – Exemplo de chamada para API REST



Fonte: Elaborado pelo próprio autor

2.3 WEB CRAWLER

Web Crawler é um aplicativo de computador que realiza a navegação pela internet de forma automatizada. Existem alguns *web crawlers* presentes e atuantes atualmente, como: *Googlebot* e o *Bingbot*. O *Googlebot* é um robô que indexa páginas da *web* usado pela Google, com a intenção de construir um índice de buscas para o Google Search. O Bingbot é um algoritmo utilizado pela *Microsoft* para realizar a indexação dos conteúdos na internet. O *Selenium* é um *framework* que realiza testes nas aplicações da internet. Para este trabalho foi utilizado o Selenium para realizar *web crawling* e assim realizar as navegações automatizadas.

2.3.1 Selenium

A API utiliza o *Selenium* para realizar a navegação por uma página web, obtendo informações contidas no site, clicando em botões, *links*, *dropdows*, imagens e assim por diante. A aplicação através do *Selenium* faz a simulação da interação de um usuário com um *website*. Este *framework* foi escolhido pelo fato da disponibilidade e compatibilidade deste com o .NET.

2.4 TRABALHOS CORRELATOS

Nesta seção detalham-se duas aplicações que possuem certa semelhança com o projeto descrito neste trabalho. Uma aplicação é um aplicativo *mobile* que apresenta os preços dos produtos nos supermercados. Já a outra aplicação é um site que mostra um produto e o preço deste nas lojas em que ele é ofertado.

O ByPantry[1] é um aplicativo *mobile* que tem como objetivo apresentar os preços dos produtos nos supermercados próximos ao usuário. A obtenção dos dados relacionados aos produtos e supermercados são feitos através da leitura de uma nota fiscal ou a inserção dos produtos e dos valores destes produtos diretamente no aplicativo. Este também possui algumas outras funcionalidades, sendo: um sistema de acumular pontos para ganhar descontos em futuras compras, compartilhar a lista de compras com os amigos ou familiares, realizar o acompanhamento diário de descontos nas listas de compras e poder realizar a compra de uma lista de produtos diretamente pelo aplicativo. Ao utilizar o aplicativo pode-se observar as vantagens que ele possui, sendo: a observação e o conhecimento dos preços dos produtos dos supermercados em tempo real e a possibilidade de poder comprar seus produtos diretamente pelo aplicativo.

O ByPantry não possui certas funcionalidades que poderiam ser interessantes, tais como: a visualização do histórico de preços de um determinado produto, disponibilização de mais opções de escolha de supermercados ao olhar a lista de compras do usuário e assim por diante. Logo, as desvantagens que se encontram no aplicativo são: a manipulação dos preços e dos produtos pode ser feita diretamente pelo usuário e a grande quantidade de itens e opções que são exibidas na tela do *smartphone*, o que pode causar uma sensação de exibição de informação desnecessária.

[1] https://bypantry.com/

Figura 7 – Tela principal do aplicativo ByPantry

Fonte: elaborado pelo próprio autor (2023)

O Buscapé é um site que exibe um produto escolhido pelo usuário e também o preço deste nas lojas em que ele é ofertado, sendo que estes produtos podem ser categorizados por diversas categorias, como: eletrônicos, roupas, alimentos, móveis e assim por diante. O Buscapé realiza a captura dos dados relacionados a um produto que o usuário esteja visualizando através do fornecimento das informações dos lojistas. Os lojistas que desejam que seu produto seja exibido no *site* do Buscapé, devem pagar uma comissão para colocar o seu inventário no *site*. Além disso, o site do do Buscapé conta também com outras funcionalidades que podem atrair o consumidor e os lojistas, sendo estas funcionalidades: *cashback*, cupons, extensão para usar em navegadores e assim por diante. Em relação as vantagens que o Buscapé possui, é possível citar: a variabilidade de segmentos de produtos ofertados, a possibilidade de comparação de preços de um produto em lojas diferentes e o

alerta ao usuário quando um produto escolhido por este chega num determinado valor estipulado por este também.

O Buscapé também não possui funcionalidades que poderiam ser agregadas, sendo a principal: a não disponibilidade de realizar a compra de um produto diretamente pelo site ou aplicativo. As desvantagens que podem ser encontradas no site ou aplicativo são: a dessincronização dos preços apresentados pelo Buscapé e pela loja ofertante do produto, existem muitas reclamações sobre a não devolução do *cashback* prometido ao realizar uma compra (TODAS. . . , 2023).

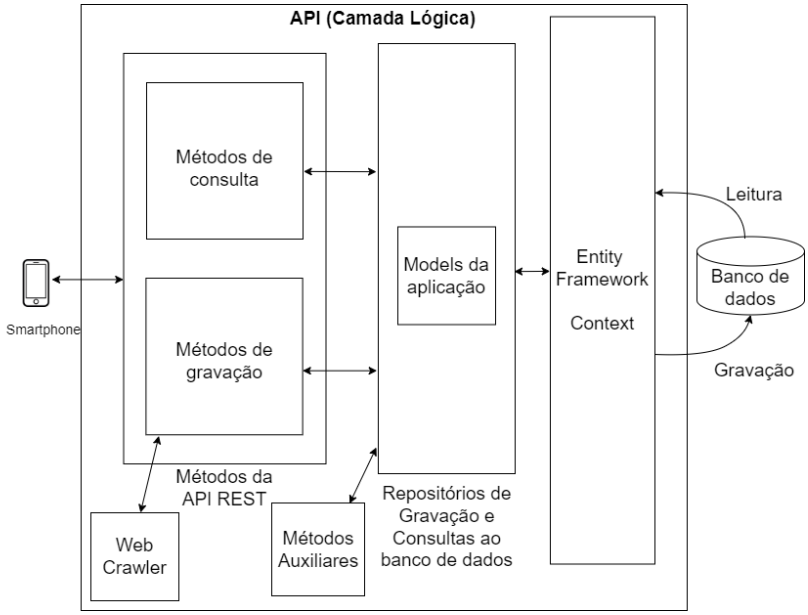
3 DESENVOLVIMENTO

Como já descrito no Capítulo 1, o objetivo deste trabalho é construir uma base dados que troque informações como uma API pública, fornecendo assim, as informações dos supermercados, os preços e o histórico de preços de um determinado produto. Nas próximas seções será descrito com mais detalhes a construção do banco de dados e da API pública.

3.1 VISÃO GERAL

A aplicação de maneira geral é composta por duas camadas: a API, também chamada de camada lógica e o banco de dados. Dentro da comada lógica se encontram quatro módulos principais, dois submódulos e dois componentes que são consumidos por estes módulos. Sendo estes módulos: Métodos da API REST, Repositórios de Gravação e Consulta ao banco de dados, Models da aplicação e o *Entity Framework*. Os submódulos que estão dentro do Métodos da API REST: Métodos de consulta e Métodos de gravação. Já os componentes são: Método Auxiliar e *Web Crawler*. A Figura 8 apresenta a visão da estrutura da aplicação.

Figura 8 – Visão Geral da aplicação



Fonte: Elaborado pelo próprio autor (2023).

A camada de banco de dados tem como objetivo receber, armazenar e fornecer as informações referentes aos supermercados, aos produtos e aos preços dos produtos nos supermercados. Existem algumas tabelas no banco de dados que servem para realizar estatísticas referentes ao uso da API e que serão detalhadas nas próximas seções. Em relação a camada lógica, o *Entity Framework* é quem realiza a mapeamento de dados e a comunicação entre os Repositórios de gravação e Consultas ao banco de dados e ao próprio banco de dados. Ele recebe os objetos advindos da API, realiza a transformação para objeto relacional e envia para o banco de dados, realizando também o processo inverso.

Os Repositórios de gravação e Consultas ao banco de dados contém classes que são responsáveis por utilizarem o *Entity Framework* para poder realizar a gravação e a obtenção de dados provenientes do banco de dados, sendo estas classes a interface de comunicação entre a API e o banco de dados. Estas classes contam com o auxílio do componente Métodos Auxiliares que tem como função capturar informações dos produtos quando necessário. Este componente será descrito com mais detalhes nas próximas seções.

Adiante, os Métodos da API REST contém dois submódulos, onde tanto o submódulo Métodos de consulta quanto o submódulo de Métodos de gravação, são responsáveis por enviar e receber dados procedentes dos Repositórios de gravação e Consultas ao banco de dados e, por fim, realizar a comunicação com o mundo externo. O submódulo de Métodos de gravação realiza o uso do componente *Web Crawler* que tem por objetivo navegar e obter dados de uma página *Web* referente a uma nota fiscal. Este componente também será descrito com mais detalhes nas próximas seções.

A parte responsável por consumir os métodos da API é um aplicativo *mobile* através de um *Smartphone*, desenvolvido por outro membro do projeto, o estudante do IFES Campus Serra Thiago Santos. Este aplicativo é responsável por apresentar as informações que o usuário necessita.

3.2 REQUISITOS FUNCIONAIS

Foram identificados os seguintes requisitos funcionais:

Tabela 2 – Requisitos Funcionais

Requisito	**Descrição**
(RF01) Requisições HTTP	A API deve receber requisições HTTP POST ou GET;
(RF02) Respostas dos métodos	A API deve retornar respostas no formato JSON para as requisições HTTP realizadas;
(RF03) Controle de produtos	A API deve armazenar e recuperar informações referente a produtos;
(RF04) Controle de preços	A API deve armazenar e recuperar informações referente a preços de produtos;
(RF05) Controle de supermercados	A API deve armazenar e recuperar informações referente a supermercados;
(RF06) Requisições a outras APIs	A API deve realizar consultas a outras APIs ou sites para obter dados;

Fonte: Elaborado pelo próprio autor (2023).

3.3 REQUISITOS NÃO FUNCIONAIS

Foram identificados os seguintes requisitos não funcionais:

Tabela 3 – Requisitos Não Funcionais

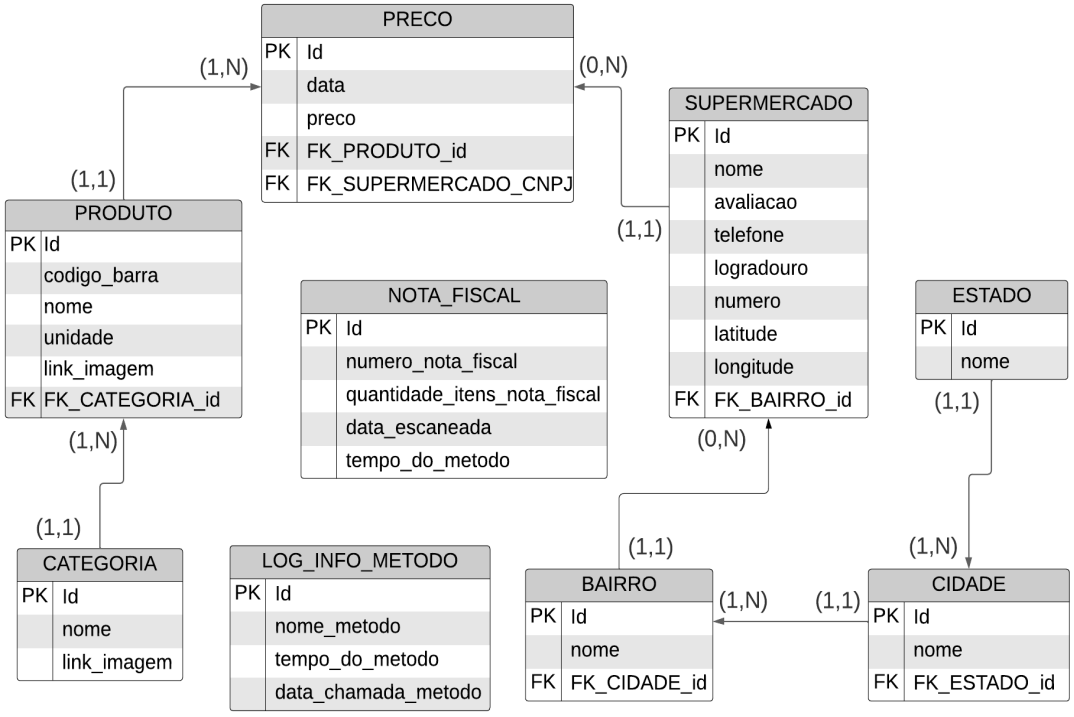
Requisito não funcional	**Descrição**
(RNF01) Sistema Gerenciador de banco de dados	A API deve utilizar a tecnologia SGBD (Sistema Gerenciador de banco de dados) SQL Server;
(RNF02) Tecnologia da aplicação utilizada	A API deve utilizar a tecnologia API WEB ASP.NET CORE;
(RNF03) Estilo de arquitetura utilizada	A API deve utilizar o estilo de arquitetura REST;
(RNF04) resposta GET da API	O tempo de resposta recomendado para responder requisições GET é por volta de 30 segundos;
(RNF05) resposta POST da API	O tempo de resposta recomendado para responder requisições POST é por volta de 2 minutos;
(RNF06) Tempo de funcionamento da API	A API deve funcionar 24/7;
(RNF07) Múltiplas requisições	A API deve suportar múltiplas requisições ao mesmo tempo;

Fonte: Elaborado pelo próprio autor (2023).

3.4 DIAGRAMA DE ENTIDADE-RELACIONAMENTO

Para representar e descrever a estrutura do banco de dados utilizado para armazenar as informações referentes aos produtos e supermercados, a **??** apresenta as tabelas e o relacionamento entre estas tabelas. Sendo possível observar na **??** que um PRODUTO que possui uma CATEGORIA também contém um PREÇO, e este PREÇO do PRODUTO está presente em um supermercado, que por sua vez contém um BAIRRO, e este BAIRRO possui uma CIDADE e esta CIDADE possui um ESTADO.

Figura 9 – Exemplo de um diagrama de entidade-relacionamento



Fonte: elaborado pelo próprio autor (2023)

As tabelas NOTA_FISCAL e LOG_INFO_METODO são utilizadas para obter informações a cerca da utilização dos métodos da API e assim gerar relatórios e gráficos para analisar o tempo de duração do método, a média total de tempo de duração do método, quantidade de vezes que foi acionado, em que momento do dia foi acionado e por fim, o maior e o menor tempo de duração do método.

3.5 TECNOLOGIAS E BIBLIOTECAS UTILIZADAS

- API WEB ASP.NET CORE

  A tecnologia API WEB ASP.NET CORE foi escolhida para servir de aplicação e conter a lógica do projeto. Como mencionado nas seções anteriores, a aplicação está estruturada de acordo com a Figura 8. A aplicação fornece uma lista de métodos, onde

o *Smartphone* realiza a requisição a algum método listado no módulo de Métodos da API REST, seguindo para Repositórios de Gravação e Consultas ao banco de dados e depois seguir para o *Entity Framework*, terminando no banco de dados com uma gravação ou leitura de dados. Por fim faz o processo inverso devolvendo uma resposta para o *Smartphone*. Este fluxo pode ser observado na Figura 8.

- SELENIUM

  O *framework* escolhido para realizar a navegação por uma página web foi o *Selenium*. Ele navega pelas páginas da nota fiscal eletrônica da Secretaria da Fazenda do Estado do Espírito Santo. Ao abrir a página principal da nota fiscal eletrônica, a aplicação através do *Selenium* clica no botão de visualizar em abas e, ao ser redirecionado para uma outra página com mais detalhes sobre a nota fiscal, ele clica em outro botão de Produtos e Serviços para a partir desta página obter as informações necessárias para a aplicação.

- ANGLE SHARP

  O *Angle Sharp* é uma biblioteca do ASP.NET CORE que tem por objetivo manipular textos em HTML e facilitar a navegação e a obtenção de dados através de comandos utilizados dentro da própria aplicação. O *Angle Sharp* foi utilizado para a navegação do texto em HTML da página principal da nota fiscal eletrônica quando não foi preciso fazer uso do *Selenium*.

- REST

  A API desenvolvida utiliza o *REST* (Representational State Transfer) como o estilo de arquitetura de software e assim dispõe de restrições e padrões que devem ser seguidos para o consumo da mesma. Para a construção da documentação e cumprimento das normas da arquitetura *REST* foi utilizado o *Swagger* que é um conjunto de ferramentas para desenvolvedores de API.

- ENTITY FRAMEWORK

  Como dito anteriormente, o *Entity Framework* é uma ferramenta da plataforma .NET de mapeamento de dados objeto-relacional e de persistência. O *Entity Framework* realiza as transformações e as gravações dos objetos instanciados dos Models da aplicação para o banco de dados SQL Server e vice-versa. Para o projeto a utilização deste *framework* foi necessário, pois a API faz a manipulação de objetos instanciados por classes do modelo de projeto orientado a objetos. Já o banco de dados, manipula seus dados através de tabelas e relacionamentos. Sendo assim, a forma que a API utiliza os dados

é diferente da forma que o banco de dados utiliza. Portanto, o *Entity Framework* possui um papel essencial para o funcionamento da aplicação como um todo.

- RESTSHARP

  O *RestSharp* é uma biblioteca cliente HTTP que funciona com todos os tipos de tecnologias .NET. Esta faz requisições para sites e APIs REST. A resposta recebida pode ser negativa ou positiva. Quando positiva, a aplicação utiliza esta resposta para realizar a manipulação do texto desta resposta recebida.

- SQL SERVER

  O banco de dados usado para armazenar e disponibilizar as informações para a API é o SGBD (Sistema Gerenciador de banco de dados) SQL Server. As tabelas e os relacionamentos entre estas tabelas estão descritos na Figura 9.

3.6 DESENVOLVIMENTO DA API DE COMUNICAÇÃO

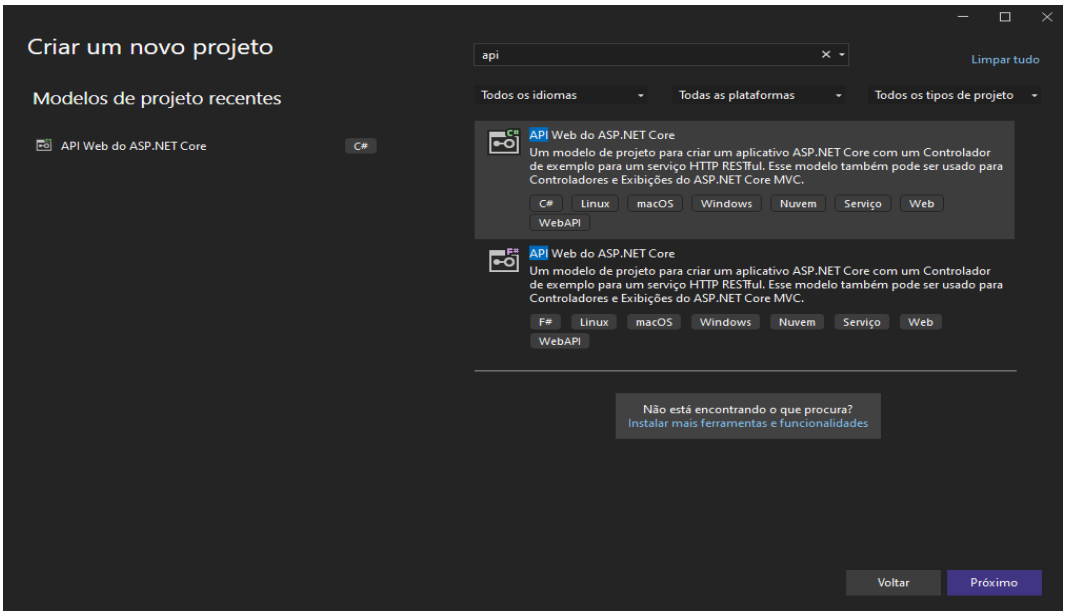
A API desenvolvida para este trabalho possui métodos onde os parâmetros de entrada seguem o padrão JSON (Javascript Object Notation). Estes métodos processam os dados recebidos, realizam as operações necessárias no banco de dados e por fim devolvem os dados de saída para o mundo externo também utilizando o formato JSON.

Ao longo da execução dos métodos da API outras funções também são realizadas, tais como: Requisições a sites ou APIs de terceiros, extração de dados destes sites ou APIs de terceiros, automatização da navegação entre páginas *Web* e acesso ao banco de dados. Estas funções executadas ao longo da execução dos métodos serão descritas com mais detalhes nas próximas seções.

3.6.1 Construção e Estrutura da Api

A API foi criada a partir do modelo de estrutura de projeto API Web do ASP.NET Core que o próprio .Net disponibiliza para iniciar o desenvolvimento da aplicação. É possível observar na Figura 10 este modelo.

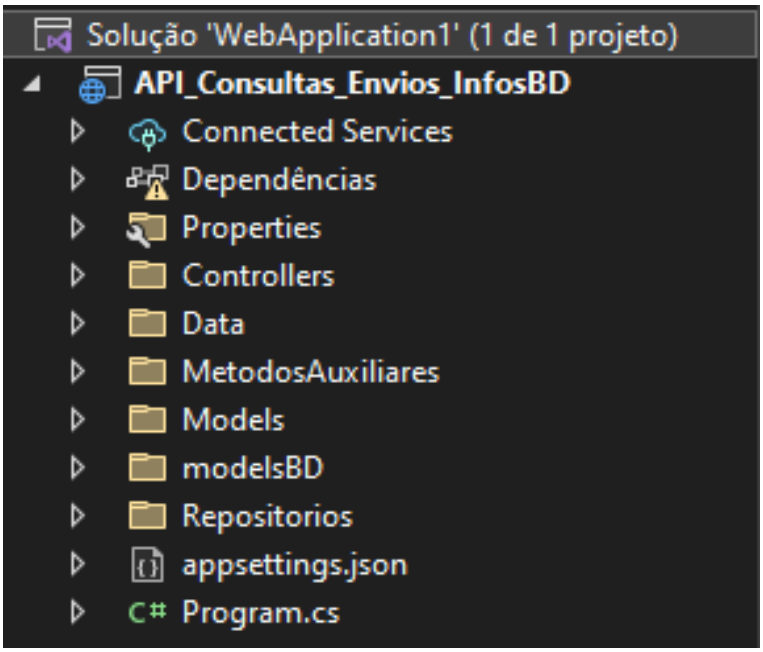
Figura 10 – Criar um novo projeto no Visual Studio



Fonte: Elaborado pelo próprio autor (2023).

Este modelo de estrutura de projeto fornece uma configuração de diretórios para criar uma API, contendo inicialmente apenas o diretório *Controllers* e algumas classes de controle de projeto, como: *Program.cs* e o *appsentings.json*. Além disso, há um *template* de declaração de classes e métodos da API do diretório *Controllers*. Estas classes e diretórios são observados na Figura 11.

Figura 11 – Estrutura do modelo de projeto API Web do ASP.NET Core



Fonte: Elaborado pelo próprio autor (2023).

O diretório Controllers é responsável por disponibilizar as rotas para o uso da API. Já o Data realiza o mapeamento dos objetos da API para os registros do banco de dados

e vice-versa. O diretório de MetodosAuxiliares tem como função realizar requisições a outros sites e APIs. O diretório Models guarda as classes que são utilizadas como objetos diretamente na aplicação. O ModelsBD possui as classes que são utilizadas como objetos na aplicação utilizando os dados obtidos do banco de dados. Por fim, o diretório Repositorios atua como intermediador entre o banco de dados e a aplicação. O arquivo *appsentings.json* possui a *string* para realizar a conexão com o banco de dados e a classe *Program.cs* possui configurações de projeto e de conexão com o banco de dados.

A Figura 12 demonstra brevemente a estrutura do código e da sintaxe da linguagem de programação utilizada. É possível observar também a declaração e a utilização de palavras reservadas para definir um *controller* e um método da API.

Figura 12 – Estrutura e sintaxe do código da linguagem de programação

```
12      [ApiController]
13      [Route("[controller]")]
        1 referência
14      public class ConsultasController : ControllerBase
15      {
16          private readonly PrecosDBContext _dbContext;
17
            0 referências
18          public ConsultasController(PrecosDBContext precosDBContext)
19          {
20              _dbContext = precosDBContext;
21          }
22
23          [HttpGet("SupermercadosProximos")]
            0 referências
24          public async Task<ActionResult> RetornarSupermercadosProximos(double latitude, double longitude, int? raioDistancia)
25          {
26              var temporizadorMetodo = new Stopwatch();
27              temporizadorMetodo.Start();
28
29              SupermercadoRepositorio supBD = new SupermercadoRepositorio(_dbContext);
30
31              List<SupermercadoCompleto> listaSupermercados = await supBD.ProcurarSupermercadosProximosCompleto(latitude, longitude, raioDistancia);
32
33              if (listaSupermercados != null)
34              {
35                  return Ok(listaSupermercados);
36              }
37
38              responseSucess responseMet = new responseSucess();
39              responseMet.status = false;
40              responseMet.message = "Nenhum supermercado encontrado próximo a sua região";
41
42              return Ok(responseMet);
43          }
44
45          [HttpGet("ProdutosCategoriaSupermercados")]
```

Fonte: Elaborado pelo próprio autor (2023).

Como mencionado nos capítulos anteriores, para a API foi utilizado a linguagem de programação orientada a objeto *C#*. Na linha 12 a palavra reservada *ApiController* indica que esta classe se trata de um conjunto de *endpoints* da API. Já as linhas 16 a 21 indicam que será utilizada a classe PrecosDBContext, que é responsável por realizar o mapeamento objeto-relacional. Na linha 23, a palavra reservada *HttpGet* indica que nas linhas de 24 a 43 se encontram os comandos necessários para a execução do método RetornarSupermercadosProximos, além de indicar também que se trata de um método *GET* da API.

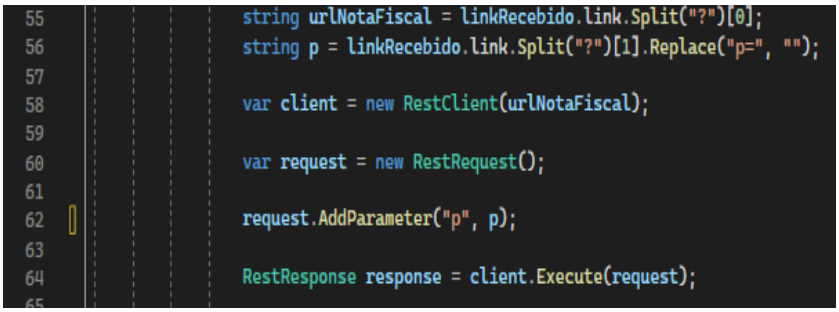
3.7 OBTENÇÃO DOS DADOS DA NOTA FISCAL

A etapa de extração de dados da nota fiscal eletrônica ocorreu através da requisição à página *Web* de nota fiscal eletrônica do consumidor. O endereço da nota fiscal é codificado

na forma de um QRcode, presente na parte inferior de toda nota fiscal impressa. A nota fiscal se encontra armazenada nos sistemas da Secretaria da Fazenda do Estado do Espírito Santo. Para a extração destes dados foram utilizados duas tecnologias: a biblioteca nativa do .NET chamada *RestSharp* e o *WebCrawler Selenium*.

A Figura 13 demonstra um exemplo de como é feita a requisição HTTP para a página *Web* de nota fiscal eletrônica utilizando o *RestSharp*.

Figura 13 – Código em C# que apresenta a forma de realizar uma requisição HTTP com *RestSharp*

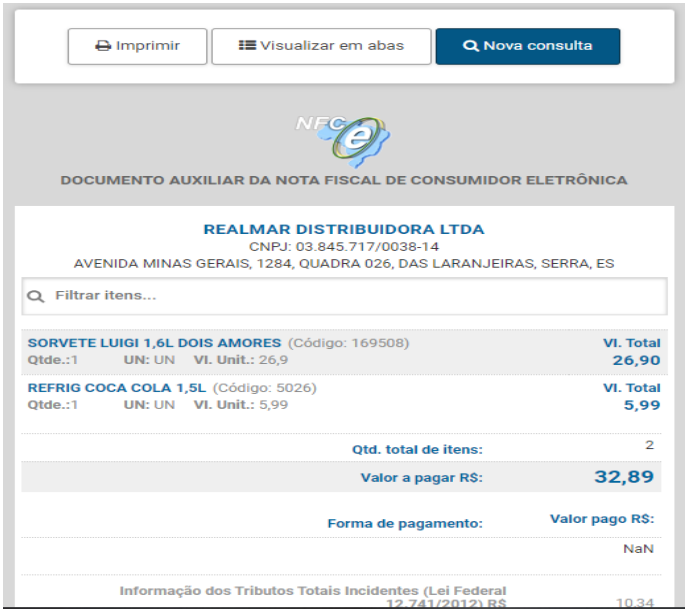


Fonte: Elaborado pelo próprio autor (2023).

Nas linhas 55 e 56 ocorre a separação da URL da nota fiscal eletrônica entre o parâmetro e o *host* da URL. Já nas linhas 58 e 60 é instanciada a classe *RestSharp* com o *host* da URL e na linha 62 é adicionado o parâmetro na classe instanciada. Por fim, na linha 64 é realizada a requisição para o site da nota fiscal eletrônica.

A Figura 14 apresenta a página principal da nota fiscal eletrônica. Neste momento são extraídos os dados de número da nota fiscal, a data de emissão da nota fiscal, o CNPJ (Cadastro nacional de pessoa jurídica) do supermercado e a quantidade de cada produto comprado.

Figura 14 – Página *Web* da nota fiscal de consumidor eletrônica

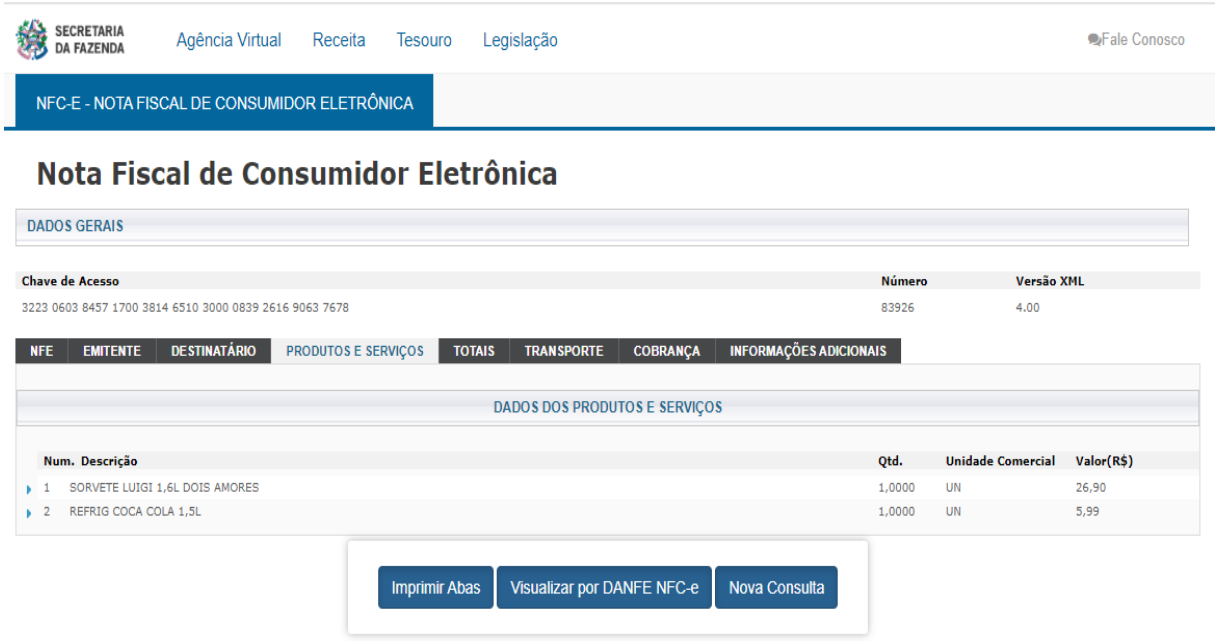


Fonte: Elaborado pelo próprio autor (2023).

O *RestSharp* foi utilizado com o objetivo de extrair dados da página principal da nota fiscal eletrônica (Figura 14). Esta biblioteca foi escolhida para obter os dados mais rapidamente quando não se é preciso navegar para outras páginas. Existem outros dados necessários para extração que se encontram em outra página e que o *RestSharp* não consegue acessar, pois não possui um endereço URL direto.

Como pode ser observado na Figura 14, esta não possui todos os dados necessários. Estes dados se encontram em outra parte da página, a qual mostra os dados detalhados da nota fiscal. Neste momento o *Selenium* é utilizado para clicar no botão “Visualizar em abas” localizado no centro superior da Figura 14 e abrir uma página secundária apresentada na Figura 15.

Figura 15 – Página *Web* da nota fiscal de consumidor eletrônica com dados gerais



Fonte: Elaborado pelo próprio autor (2023).

Desta página secundária são extraídos os dados de nome do produto, unidade de medida do produto, código NCM (Nomenclatura Comum Mercosul) do produto, código do produto em um supermercado específico, o preço do produto e o código de barras do produto.

A Figura 16 demonstra um exemplo de uso do *Selenium* para a navegação entre páginas e extração de dados.

Figura 16 – Código em C# que apresenta a forma de realizar uma requisição HTTP usando o *Selenium*

```
124         ChromeOptions opt = new ChromeOptions();
125         opt.AddArguments("headless");
126
127         var webdriver = new ChromeDriver(opt);
128
129         webdriver.Navigate().GoToUrl(linkRecebido.link);
130
131         IWebElement BtnSeguir = null;
132         BtnSeguir = webdriver.FindElement(By.XPath("//*[@id='body_btnVisualizarAbas']"));
133         BtnSeguir.Click();
134
135         IWebDriver aq = webdriver.SwitchTo().Window(webdriver.WindowHandles.Last());
136
137         BtnSeguir = webdriver.FindElement(By.XPath("//*[@id='tab_3']"));
138         BtnSeguir.Click();
139
140         aq = webdriver.SwitchTo().Window(webdriver.WindowHandles.Last());
141
142         IReadOnlyList<IWebElement> listaProdutosVisivel = webdriver.FindElements(By.XPath("//table[@class='toggle box']"));
143         IReadOnlyList<IWebElement> listaProdutos = webdriver.FindElements(By.XPath("//table[@class='toggable box']"));
```

Fonte: Elaborado pelo próprio autor (2023).

Nas linhas 124 a 127 da Figura 16 a classe do *Selenium* é instanciada. Na linha 129 é realizado a requisição *HTTP GET* para a página principal do site da nota fiscal eletrônica. Seguindo na linha 132, o botão de Visualizar em abas é capturado e na linha 133 o botão é clicado a fim de obter a página secundária contendo mais informações a cerca da nota fiscal. Esta página com as informações é acessada na linha 135 e nas linhas 137 e 138 é realizado novamente o clique em um botão na página secundária para obter as informações que realmente são necessárias para extrair. Por fim, na linha 140 acontece a navegação para

outra parte da página secundária e nas linhas 142 e 143 ocorrem a extração de informações necessárias para utilização.

Como mencionado nas seções anteriores, existem alguns métodos da API que realizam a requisição a métodos de APIs e sites de terceiros. Um método da API da *distancematrix.ai* retorna a distância entre dois pontos dispostos no mapa. Outro método da API da docs.minhareceita.org retorna informações a cerca de um determinado supermercado, como o nome fantasia e o endereço do supermercado. O método da API do developer.tomtom.com retorna as latitude e longitude de um determinado supermercado.

O site da Secretaria da Fazenda do Estado do Espírito Santo é o principal site consultado para extração de informações relacionadas aos produtos da nota fiscal eletrônica. Entretanto, ele não possui todas as informações necessárias. Por exemplo, a nota fiscal não possui a imagem do produto. Além disso, os nomes de produtos presentes na nota fiscal podem estar abreviados. Para resolver esses problemas, foi consultado o site Bluesoft, que contém informações sobre produtos do varejo. Ele é consultado quando os nomes dos produtos na nota fiscal eletrônica estão de forma simplificada, dificultando a compreensão e a utilização do nome do produto. Os dados extraídos a partir deste site são: O nome completo do produto e também o *link* da imagem do produto. Quando o site da bluesoft não possui a imagem correta do produto, o site de busca da Google é utilizado para obter *links* de imagem de produtos. Pode ocorrer a busca por um *link* de imagem do produto, tanto no site da bluesoft quanto no site da Google, que possui o nome do produto abreviado. Existe também a possibilidade de um produto não possuir um código de barras e conter nomes distintos em diferentes supermercados.

O código NCM (Nomenclatura Comum Mercosul) é um código de oito dígitos estabelecido pela legislação federal e é usado para identificar os produtos e também facilitar o comércio internacional (NCM, 2019). O Portal Único Siscomex é um site que disponibiliza uma lista de capítulos e subcapítulos contendo o agrupamento dos produtos com características semelhantes (SUMáRIO, 2023). Para o projeto, este Portal Único Siscomex foi consultado a fim de obter a compreensão acerca da categorização dos produtos. A aplicação contém dentro de sua estrutura interna um mapeamento dos capítulos e subcapítulos deste Portal Único Siscomex e classifica o produto de acordo com este mapeamento. Quando a aplicação está realizando esta classificação com um código NCM de um produto da nota fiscal, ela verifica que este código pertence a uma determinada categoria, fazendo assim a categorização dos produtos e agrupando-os de acordo com suas características semelhantes a fim de facilitar a navegação do usuário.

A API desenvolvida para este projeto recebe apenas *links* de notas fiscais eletrônicas do estado do Espírito Santo. Para a leitura de notas fiscais eletrônicas de outros estados é preciso refazer ou criar o método de recebimento de *links* de notas fiscais.

4 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos após o processo de desenvolvimento da aplicação. De modo geral, os resultados serão divididos em duas seções, uma para apresentar a documentação dos métodos da API e outra para realizar a análise acerca do comportamento dos métodos da API através de gráficos e estatísticas.

4.1 DOCUMENTAÇÃO DOS MÉTODOS DA API

A API desenvolvida para o trabalho contém dois métodos *HTTP POST* e nove métodos *HTTP GET*. A API utiliza o modelo de arquitetura de software *REST* e as respostas de todos os métodos utilizam o formato JSON. Os parâmetros de entrada e as informações retornadas nas respostas dos métodos serão descritos nesta seção.

4.1.1 Métodos de Envios (Http Post)

4.1.1.1 LinkNotaFiscal

Este método é responsável por receber como parâmetro o link de uma nota fiscal, capturar as informações contidas nesta nota fiscal eletrônica e gravar somente as informações necessárias no banco de dados. A resposta para a aplicação que chamou o método é um *JSON* contendo as informações do supermercado, como nome e endereço, além de conter também as informações dos produtos da nota fiscal, como nome, preço, código de barras, quantidade do produto e unidade de medida.

Rota:

POST - envios/LinkNotaFiscal

Exemplo de Entrada:

```
{
  "linkRecebido": "Link da nota fiscal"
}
```

Exemplo de Resposta:

```
{
  "CNPJ": 1234567,
  "nome": "Supermercado Gelol",
  "avaliacao": 4.2,
  "telefone": 999999999,
  "logradouro": "Rua das Estrelas",
  "numero": 12,
  "bairro": "bairro Jardim Encantado",
  "estado": "Alegria do Sul",
```

```
10    "cidade": "Serenópolis",
11    "dataEmissao": "dd/MM/yyyy",
12    "listaProdutos": [{
13      nomeProduto: "Biscoito lefer",
14      codigo_barra: "1234456",
15      quantidade: "1",
16      unidade: "UN",
17      valorUnidade: 3.00
18    }]
19  }
```

4.1.1.2 ProdutosEscolhidosCarrinho

Este método recebe como parâmetros de entrada a latitude e longitude do usuário, o raio de distância escolhido pelo usuário em metros e uma lista dos produtos contendo o nome ou o código do produto. Sua função é encontrar os supermercados próximos ao usuário, listar os produtos escolhidos pelo usuário em cada um dos supermercados encontrados, para no final da execução do método, retornar como resposta um *JSON* com uma lista contendo as informações dos supermercados próximos encontrados, como nome e endereço, juntamente com as informações dos produtos da lista que foram passados para o método, como nome, preço, descrição, código de barra, unidade de medida e link da imagem do produto.

Rota:

POST - envios/ProdutosEscolhidosCarrinho

Exemplo de Entrada:

```
1  {
2    "latitudeUsuario": 20.13243,
3    "longitudeUsuario": 20.16243,
4    "raioDistanciaMetros": 1000,
5    "listaNomeProduto": [{
6      "Nome": "Nome do produto",
7      "Codigo": "Código de barras do produto"
8    }]
9  }
```

Exemplo de Resposta:

```
1  [{
2    "CNPJ": 1234567,
3    "NomeSupermercado": "Supermercado Gelol",
4    "avaliacao": 4.2,
5    "telefone": 999999999,
```

```
6    "logradouro": "Rua das Estrelas",
7    "numero": 12,
8    "nomeBairro": "bairro Jardim Encantado",
9    "nomeEstado": "Alegria do Sul",
10   "nomeCidade": "Serenópolis",
11   "Produtos": [{
12     "Nome": "Biscoito lefer",
13     "Preco": 3.21,
14     "Descricao": "Produto catalogado.",
15     "codigo_barra": "1234456",
16     "unidade_medida": "UN",
17     "link_imagem": "link da imagem"
18   }]
19 }]
```

4.1.2 Métodos de Consultas (Http Get)

4.1.2.1 SupermercadosProximos

Este método recebe como parâmetros a latitude e longitude do usuário e o raio de distância escolhido pelo usuário em metros. O método irá encontrar os supermercados próximos ao usuário e retornar como resposta um *JSON* com uma lista contendo o nome e o endereço de todos os supermercados próximos a localização do usuário.

Rota:

GET - consultas/SupermercadosProximos

Exemplo de Entrada:

```
1 latitude=20.13243&longitude=20.16243&raioDistanciaMetros=1000
```

Exemplo de Resposta:

```
1  [{
2    "CNPJ": 1234567,
3    "nome": "Supermercado Gelol",
4    "avaliacao": 4.2,
5    "telefone": 999999999,
6    "logradouro": "Rua das Estrelas",
7    "numero": 12,
8    "nomeBairro": "bairro Jardim Encantado",
9    "nomeEstado": "Alegria do Sul",
10   "nomeCidade": "Serenópolis",
11   "latitude": 20.67342,
12   "longitude": 20.69023
13 }]
```

4.1.2.2 ProdutosCategoriaSupermercados

Este método recebe como parâmetro o CNPJ do supermercado ou o nome do supermercado e a categoria de um produto. A função do método é encontrar todos os produtos da categoria, passada como parâmetro, no supermercado também passado como parâmetro e retornar como resposta um *JSON* com uma lista com as informações dos produtos referentes a categoria no supermercado. As informações de um produto retornadas são: nome, código de barras, unidade de medida, preço e o link de imagem.

Rota:

GET - consultas/ProdutosCategoriaSupermercados

Exemplo de Entrada:

```
1  categoria=carnes&CNPJSupermercado=1234567
```

Exemplo de Resposta:

```
1 [{
2   "nome": "Biscoito lefer",
3   "codigo_barra": "1234456",
4   "unidade_medida": "UN",
5   "preco": 3.25,
6   "link_imagem": "link da imagem"
7 }]
```

4.1.2.3 ProdutosCategoria

Este método recebe como parâmetro uma categoria de produtos. Sua função é encontrar todos os produtos referentes a categoria passada como parâmetro e por fim retornar como resposta um *JSON* com uma lista com as informações dos produtos referentes a categoria passada para o método.

Rota:

GET - consultas/ProdutosCategoria

Exemplo de Entrada:

```
1  categoria=carnes
```

Exemplo de Resposta:

```
1 [{
2   "nome": "Pá bovina",
3   "codigo_barra": "1211496",
```

```
4     "unidade_medida": "UN",
5     "preco": 3.25,
6     "link_imagem": "link da imagem"
7   }]
```

4.1.2.4 HistoricoPrecoGeral

Este método recebe como parâmetro o nome do produto ou o código de barras do produto e como parâmetros opcionais a data inicial e a data final para a consulta do histórico de preços. A função do método é encontrar o produto informado como parâmetro e seu histórico de preços. O método retorna como resposta um *JSON* com as informações do produto, como: nome, código de barras, link da imagem e unidade de medida, juntamente com uma lista com os preços e as datas obtidas no intervalo de tempo especificado.

Rota:

GET - consultas/HistoricoPrecoGeral

Exemplo de Entrada:

```
1   codigo_barra=1234456&dataInicial=2023-07-29&dataFinal=2023-09-01
```

Exemplo de Resposta:

```
1   {
2     "nomeProduto": "Biscoito lefer",
3     "codigo_barra": "1234456",
4     "unidade_medida": "UN",
5     "link_imagem": "link da imagem",
6     "listPrecoGeral": [{
7       "data": "2023-06-18T11:09:38",
8       "preco": 3.25
9     }]
10  }
```

4.1.2.5 CategoriasProdutos

Este método não recebe nenhuma informação como parâmetro. Este método apenas obtém todas as categorias de produtos cadastradas no banco de dados e retorna como resposta um *JSON* com uma lista de categorias, onde cada categoria possui nome e link de imagem da categoria.

Rota:

GET - consultas/CategoriasProdutos

Exemplo de Entrada:

Exemplo de Resposta:

```
[{
  "nome": "carnes",
  "link_imagem": "link da imagem"
}]
```

4.1.2.6 SupermercadosProduto

Este método recebe como parâmetros o nome do produto ou o código de barra do produto. A função do método é encontrar todos os supermercados em que existe o produto informado como parâmetro. O método irá retornar como resposta um *JSON* com uma lista de supermercados contendo o produto, juntamente com as informações do produto no supermercado. Para o supermercado estas informações são nome e o endereço. Já para o produto são nome, código de barras, unidade de medida, preço e o link da imagem.

Rota:

GET - consultas/SupermercadosProduto

Exemplo de Entrada:

```
  codigo_barra=1234456
```

Exemplo de Resposta:

```
[{
  "CNPJ": 1234567,
  "nome": "Supermercado Gelol",
  "telefone": 999999999,
  "logradouro": "Rua das Estrelas",
  "numero": 12,
  "bairro": "bairro Jardim Encantado",
  "estado": "Alegria do Sul",
  "cidade": "Serenópolis",
  "produto": {
    "nome": "Biscoito lefer",
    "codigo_barra": "1234456",
    "unidade_medida": "UN",
    "preco": 3.25,
    "link_imagem": "link da imagem"
  }
}]
```

4.1.2.7 HistoricoPrecoSupermercado

Este método recebe como parâmetros o nome do produto ou o código de barras do produto, o CNPJ do supermercado e como parâmetros opcionais a data inicial e data final para a consulta do histórico de preços. A função do método é encontrar todos os preços referentes a um produto em um supermercado, sendo este produto e supermercado informados como parâmetros ao método. Ao final, retornar como resposta um *JSON* com as informações do produto, como nome, código de barras, link da imagem e unidade de medida, juntamente com uma lista com os preços e as datas obtidas no intervalo de tempo especificado.

Rota:

GET - consultas/HistoricoPrecoSupermercado

Exemplo de Entrada:

```
codigo_barra=1234456&CNPJSupermercado=1234567&dataInicial=2023-07-29&
    dataFinal=2023-09-01
```

Exemplo de Resposta:

```
{
  "nomeProduto": "Biscoito lefer",
  "codigo_barra": "1234456",
  "unidade_medida": "UN",
  "link_imagem": "link da imagem",
  "listPrecoGeral": [{
    "data": "2023-06-18T11:09:38",
    "preco": 3.25
  }]
}
```

4.1.2.8 RelatorioInfoMetodo

Este método recebe como parâmetro o nome do método e como parâmetros opcionais a data inicial e a data final. A função do método é obter todos os registros referentes ao método que foi informado como parâmetro, realizar a média do tempo de execução de todas as vezes em que o método foi acionado e por fim retornar como resposta um *JSON* contendo a quantidade de vezes que o método foi acionado, a média do tempo de execução do método, o maior tempo de execução do método, o menor tempo de execução do método e uma lista com as horas do dia com as respectivas quantidade de vezes que método foi acionado.

Rota:

GET - consultas/RelatorioInfoMetodo

Exemplo de Entrada:

```
nomeMetodo=HistoricoPrecoSupermercado&dataInicial=2023-07-29&dataFinal
    =2023-09-01
```

Exemplo de Resposta:

```
{
  "quantidadeVezesChamada": 323,
  "mediaTempoMetodo": 2.1,
  "MaiorMenortempo": [{
    "id": 2,
    "nomeMetodo": "HistoricoPrecoSupermercado",
    "dataChamadaMetodo": "2023-06-18T11:09:38",
    "tempoDoMetodo": 1.65
  }],
  "listaChamadasNoDia": [{
    "horaDoDia": 1,
    "quantidadeVezesChamadaMetodo": 93
  }]
}
```

4.1.2.9 RelatorioNotaFiscal

Este método é responsável por receber como parâmetros opcionais a data inicial e a data final. A função do método é obter todos os registros de nota fiscal eletrônica, realizar a média do tempo de execução de todas as vezes em que o método foi acionado e por fim retornar como resposta um *JSON* contendo a quantidade de vezes que o método foi acionado, a média do tempo de execução do método, o maior tempo de execução do método com a respectiva quantidade de produtos na nota fiscal, o menor tempo de execução do método também com a respectiva quantidade de produtos na nota fiscal e uma lista com as horas do dia com as respectivas quantidade de vezes que método foi acionado.

Rota:

GET - consultas/RelatorioNotaFiscal

Exemplo de Entrada:

```
dataInicial=2023-07-29&dataFinal=2023-09-01
```

Exemplo de Resposta:

```
{
  "quantidadeVezesChamada": 323,
```

```
  "mediaTempoMetodo": 2.1,
  "MaiorMenortempo": [{
    "id": 2,
    "numeroNotaFiscal": "252642-117",
    "quantidadeItensNotaFiscal": 22,
    "dataEscaneada": "2023-06-18T11:09:38",
    "tempoDoMetodo": 1.95
  }],
  "listaChamadasNoDia": [{
    "horaDoDia": 1,
    "quantidadeVezesChamadaMetodo": 93
  }]
}
```


4.2 ANÁLISE DO COMPORTAMENTO DOS MÉTODOS DA API

Como mencionado no capítulo 3, As tabelas NOTA_FISCAL e LOG_INFO_METODO são utilizadas para a geração de gráficos e análises a cerca da utilização dos métodos da API.

O período em que a API foi utilizada juntamente com a aplicação cliente para a realização de testes e alimentação do banco de dados foi de 1 de outubro a 31 de outubro. Ao final deste período, o banco de dados obteve as seguintes informações registradas em suas tabelas: 64 notas fiscais eletrônicas, 505 produtos, 611 preços referentes aos produtos, 18 supermercados e 11 usuários que realizaram o *download* do aplicativo *mobile*.

4.2.1 Método de LinkNotaFiscal

Para analisar o método de envio de link de nota fiscal, foram obtidas as seguintes informações das tabelas do banco de dados: Quantidade de vezes que o método foi acionado, média do tempo de execução do método, o maior tempo de execução do método, o menor tempo de execução do método e as horas do dia em que o método foi acionado por mais vezes.

- O método foi acionado 64 vezes e, consequentemente, foram registradas 64 notas fiscais no banco de dados. Para obter esta informação, foi realizada a soma do total de registros obtidos do banco de dados.

- A média do tempo de execução do método foi de 45 segundos. Para realizar esta média, a soma total do tempo de execução do método em segundos foi dividido pela quantidade de registros de notas fiscais.

- O maior tempo para analisar e ler uma nota fiscal foi de 1 minuto e 16 segundos. Esta nota fiscal com Id igual a 17 continha 2 produtos e foi escaneada em 06/10/2023.

- O menor tempo para analisar e ler uma nota fiscal foi 27 segundos. Esta nota fiscal com Id igual a 41 continha também 2 produtos e foi escaneada em 20/10/2023.

- As horas do dia em que o método foi mais acionado foram às 9h com 11 acionamentos e às 18h com 15 acionamentos.

Levando em consideração que as notas que apresentaram maior e menor tempo de execução contém a mesma quantidade de produtos, é possível supor que o número de produtos presentes na nota fiscal não é o elemento influenciador principal em relação ao desempenho do método. Estas duas notas fiscais foram lidas em dias diferentes e, por isso, é possível supor novamente que o tempo de leitura da nota fiscal é mais influenciado pelo desempenho do servidor da Secretaria da Fazenda do Espírito Santo. Considerando o menor tempo de 27 segundos para analisar e ler uma nota fiscal, pode-se perceber que o método possui um tempo de resposta elevado, o qual pode não ser aceitável. Do ponto de vista do usuário, pode gerar um certo nível de insatisfação ao utilizar o sistema, pelo fato desses estarem acostumados com tempos de respostas muito menores.

As horas do dia em que método foi mais acionado foram às 9h e às 18h. Sendo assim, pode-se supor que os usuários geralmente realizam compras ou pela manhã, antes do início do preparo do almoço, ou no final da tarde, horário que coincide com o final do expediente de trabalho da maioria das pessoas.

4.2.2 Método de SupermercadosProximos

Para analisar o método de obter todos os supermercados dentro do limite de distância fornecido, foram obtidas as seguintes informações das tabelas do banco de dados: Quantidade de vezes que o método foi acionado, média do tempo de execução do método, o maior tempo de execução do método, o menor tempo de execução do método e as horas do dia em que o método foi acionado por mais vezes.

- O método foi acionado 105 vezes. Para obter esta informação, foi realizada a soma do total de registros obtidos do banco de dados.

- A média do tempo de execução do método foi de 6.5 segundos. Para realizar esta média, a soma total do tempo de execução do método em segundos foi dividido pela quantidade de registros de notas fiscais.

- O maior tempo de execução do método para obtenção das informações foi de 15 segundos. Este método foi acionado em 16/10/2023.

- O menor tempo de execução do método para obtenção das informações foi de 0.5 segundos. Este método foi acionado em 28/09/2023.

- As horas do dia em que o método foi mais acionado foram às 17h com 13 acionamentos e às 18h com 14 acionamentos.

Levando em consideração o maior e o menor tempo de execução do método, é possível supor que o fator que mais influencia no desempenho do método é o tempo de resposta da API da *distancematrix.ai*, consumida para obter a distância entre dois pontos dispostos no mapa. Enquanto a resposta desta API não for obtida, o método fica esperando até obter o resultado e só então finaliza e retorna as informações para o cliente.

As horas do dia em que método foi mais acionado foram às 17h e às 18h. Sendo assim, pode-se supor novamente que o método foi mais acionado nestas horas pelo fato de serem as horas em que os consumidores estão saindo do trabalho e aproveitam para realizar compras.

4.2.3 Método de ProdutosCategoriaSupermercados

Para analisar o método de obter todos os produtos referentes a uma categoria fornecida, foram obtidas as seguintes informações das tabelas do banco de dados: Quantidade de vezes que o método foi acionado, média do tempo de execução do método, o maior tempo de execução do método, o menor tempo de execução do método e as horas do dia em que o método foi acionado por mais vezes.

- O método foi acionado 57 vezes. Para obter esta informação, foi realizada a soma do total de registros obtidos do banco de dados.

- A média do tempo de execução do método foi de 0.12 segundos. Para realizar esta média, a soma total do tempo de execução do método em segundos foi dividido pela quantidade de vezes que o método foi acionado.

- O maior tempo de execução do método para obtenção das informações foi de 0.4 segundos. Este método foi acionado em 20/10/2023.

- O menor tempo de execução do método para obtenção das informações foi de 0.008 segundos. Este método foi acionado em 09/11/2023.

- As horas do dia em que o método foi mais acionado foram às 18h com 17 acionamentos e às 19h com 14 acionamentos.

Levando em consideração o maior e o menor tempo de execução do método, é possível supor que o fator que mais influencia no desempenho do método é apenas o tempo de resposta do banco de dados.

As horas do dia em que método foi mais acionado foram às 18h e às 19h. Sendo assim, pode-se supor novamente que o método foi mais acionado nestas horas pelo fato de serem as horas em que os consumidores estão saindo do trabalho e aproveitam para realizar compras.

4.2.4 Método de HistoricoPrecoSupermercado

Para analisar o método que obtém o histórico de preços de um produto em um supermercado, foram obtidas as seguintes informações das tabelas do banco de dados: Quantidade de vezes que o método foi acionado, média do tempo de execução do método, o maior tempo de execução do método, o menor tempo de execução do método e as horas do dia em que o método foi acionado por mais vezes.

- O método foi acionado 12 vezes. Para obter esta informação, foi realizada a soma do total de registros obtidos do banco de dados.

- A média do tempo de execução do método foi de 0,015 segundos. Para realizar esta média, a soma total do tempo de execução do método em segundos foi dividido pela quantidade de vezes em que o método foi acionado.

- O maior tempo de execução do método para obtenção das informações foi de 0.05 segundos. Este método foi acionado em 27/10/2023.

- O menor tempo de execução do método para obtenção das informações foi de 0.006 segundos. Este método foi acionado em 31/10/2023.

- As horas do dia em que o método foi mais acionado foram às 9h com 10 acionamentos e às 18h com 2 acionamentos.

Levando em consideração o maior e o menor tempo de execução do método, novamente é possível supor que o fator que mais influencia no desempenho do método é apenas o tempo de resposta do banco de dados.

As horas do dia em que método foi mais acionado foram às 9h e às 18h. Sendo assim, pode-se supor que o método foi mais acionado nestas horas pelo fato de serem as horas em que os consumidores estão prestes a ingressarem ou saírem do trabalho.

4.2.5 Método de SupermercadosProduto

Para analisar o método que obtém todos os supermercados em que se encontram um produto, foram obtidas as seguintes informações das tabelas do banco de dados: Quantidade de vezes que o método foi acionado, média do tempo de execução do método, o maior tempo de execução do método, o menor tempo de execução do método e as horas do dia em que o método foi acionado por mais vezes.

- O método foi acionado 88 vezes. Para obter esta informação, foi realizada a soma do total de registros obtidos do banco de dados.

- A média do tempo de execução do método foi de 0.03 segundos. Para realizar esta média, a soma total do tempo de execução do método em segundos foi dividido pela quantidade de vezes que o método foi acionado.

- O maior tempo de execução do método para obtenção das informações foi de 0.21 segundos. Este método foi acionado em 07/11/2023.

- O menor tempo de execução do método para obtenção das informações foi de 0.005 segundos. Este método foi acionado em 07/10/2023.

- As horas do dia em que o método foi mais acionado foram às 15h com 13 acionamentos e às 16h com 14 acionamentos.

Levando em consideração o maior e o menor tempo de execução do método, novamente é possível supor que o fator que mais influencia no desempenho do método é apenas o tempo de resposta do banco de dados.

As horas do dia em que método foi mais acionado foram às 15h e às 16h. Pode-se supor que o método foi mais acionado nestas horas pelo fato de serem as horas em que os consumidores pesquisam antes de realmente se deslocarem até ao supermercado, obtendo assim a informação da disponibilidade de um produto nos supermercados.

5 CONSIDERAÇÕES FINAIS

Nesta seção será apresentada a conclusão a respeito do desenvolvimento do sistema, as principais dificuldades encontradas durante o processo e as melhorias futuras.

5.1 CONCLUSÃO

Neste trabalho foi construído um banco de dados que contém informações referentes aos produtos, histórico de preços dos produtos e supermercados, juntamente com uma API pública que disponibiliza uma lista de métodos de consultas e envios. Sendo assim, todos os objetivos foram alcançados com sucesso.

Foram encontradas dificuldades em relação a categorização correta dos produtos advindos da nota fiscal eletrônica; gravar e obter um produto que não contém um código de barras; e obter a imagem correta em relação a algum produto quando este possui um nome abreviado na nota fiscal eletrônica.

Considerando todos os requisitos levantados para a realização do projeto, todos os requisitos considerados importantes para o funcionamento da API foram implementados.

Foi possível comprovar a importância da utilização de técnicas de análise de requisitos, construção de diagramas de entidade-relacionamento e padrões de projeto aprendidas durante o curso no processo de desenvolvimento. O levantamento de requisitos e o diagrama de entidade-relacionamento ajudaram na definição do que seria suficiente para construir a aplicação, de modo que fosse desenvolvido uma aplicação funcional.

Também é possível concluir que as tecnologias utilizadas para o desenvolvimento da aplicação foram boas escolhas visto que, todas as tecnologias utilizadas atenderam as necessidades requeridas para a realização do trabalho.

5.2 TRABALHOS FUTUROS

Embora os objetivos propostos no trabalho tenham sido alcançados, algumas melhorias são possíveis visando trabalhos futuros:

- Realizar a classificação da categoria de todos os produtos de forma mais precisa.
- Realizar o desenvolvimento de uma classe que consiga definir o nome real e completo de um produto que possui o nome abreviado.
- Realizar o desenvolvimento de uma classe que consiga identificar produtos iguais sem código de barras e com nomes diferentes e assim não gravar no banco de dados um

produto repetido.

- Realizar a seleção de um link de imagem do produto mais precisamente quando o produto possui o nome abreviado.

REFERÊNCIAS

ENGENHARIA de software. 2023. Disponível em: <https://pt.wikipedia.org/wiki/Engenharia_de_software>. Acesso em: 03 jul. 2023.

GS1. *Saiba como funciona o registro do código de barras*. 2018. Disponível em: <https://blog.gs1br.org/saiba-como-funciona-o-registro-do-codigo-de-barras/>. Acesso em: 13 dez. 2023.

HAT, Red. *API REST*. 2023. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 07 jul. 2023.

IORE, Andye. *Pesquisa do Procon aponta diferença de até 294% nos preços dos produtos da cesta básica*. 2023. Disponível em: <http://www.maringa.pr.gov.br/site/noticias/2023/02/13/pesquisa-do-procon-aponta-diferenca-de-ate-294-nos-precos-dos-produtos-da-cesta-basica/41092>. Acesso em: 17 abr. 2023.

JR, Mazer. *O que é o Protocolo HTTP*. 2020. Disponível em: <https://mazer.dev/pt-br/http/introducao-protocolo-http/>. Acesso em: 06 jul. 2023.

LEãO, Thiago. *Código de barras: o que é, como funciona e principais tipos*. 2022. Disponível em: <https://www.nomus.com.br/blog-industrial/codigo-de-barras/>. Acesso em: 09 mai. 2023.

LUCIDCHART. *O que é um diagrama entidade relacionamento?* 2023. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento>. Acesso em: 03 jul. 2023.

MACORATTI, José Carlos. *Curso Entity Framework - Cenários de Persistência - XVI*. 2023. Disponível em: <https://mazer.dev/pt-br/http/introducao-protocolo-http/>. Acesso em: 07 jul. 2023.

NCM. 2019. Disponível em: <https://www.gov.br/receitafederal/pt-br/assuntos/aduana-e-comercio-exterior/classificacao-fiscal-de-mercadorias/ncm>. Acesso em: 09 nov. 2023.

PAULI, Liciani Beatriz. *Comércio varejista brasileiro: uma análise dos determinantes macroeconômicos das vendas*. 2019. Disponível em: <https://repositorio.ufsm.br/handle/1/16608#:~:text=A\%20importÃćncia\%20do\%20comÃŀrcio\%20varejista,para\%20o\%20crescimento\%20econÃťmico\%20brasileiro.> Acesso em: 24 abr. 2023.

PINHO, Raquel. *Supermercados: ainda bem que sempre tem um perto de você*. 2021. Disponível em: <https://www.segs.com.br/mais/economia/318511-supermercados-ainda-bem-que-sempre-tem-um-perto-de-voce#:~:text=Mas\%20a\%20import\%C3\%A2ncia\%20do\%20setor,\%2C5\%25\%20do\%20PIB\%20brasileiro.> Acesso em: 09 mai. 2023.

PMC - Pesquisa Mensal de Comércio. 2023. Disponível em: <https://www.ibge.gov.br/estatisticas/economicas/comercio/9227-pesquisa-mensal-de-comercio.html?=&t=resultados>. Acesso em: 24 abr. 2023.

REIS, Cristiano. *Código de barras: como funciona e os diferentes tipos*. 2022. Disponível em: <https://cr.inf.br/blog/codigo-de-barras-como-funciona-e-os-diferentes-tipos/#:~:text=A\%20GS1\%20Brasil\%20\%C3\%A9\%20a,no\%20Brasil\%20e\%20no\%20mundo.&text=A\%20filia\%C3\%A7\%C3\%A3o\%20envolve\%20a\%20assinatura,uma\%20inscri\%C3\%A7\%C3\%A3o\%20e\%20uma\%20anuidade.> Acesso em: 09 mai. 2023.

SAMPAIO, Guilherme. *Introdução a Arquitetura Cliente-Servidor*. 2021. Disponível em: <https://medium.com/@kaisergui258/introduÃğÃčo-a-arquitetura-cliente-servidor-dfae4c0218bd>. Acesso em: 05 jul. 2023.

SUMáRIO. 2023. Disponível em: <https://portalunico.siscomex.gov.br/classif/#/sumario?perfil=publico>. Acesso em: 12 nov. 2023.

TODAS as reclamações para Buscapé. 2023. Disponível em: <https://www.reclameaqui.com.br/empresa/buscape/lista-reclamacoes/>. Acesso em: 06 jul. 2023.

TOVAR, Yamara. *Preços nos supermercados do ES varia até 202%, diz Procon*. 2023. Disponível em: <https://tribunaonline.com.br/economia/preco-nos-supermercados-do-es-varia-ate-202-diz-procon-136766?home=esp\%C3\%ADrito+santo>. Acesso em: 17 abr. 2023.