UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Ludiiprice: API para comparação de preços através de notas fiscais de produtos emitidas por estabelecimentos comerciais

Trabalho de Conclusão de Curso

Adam Cordeiro Araújo

São Cristóvão – Sergipe

2022

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Adam Cordeiro Araújo

Ludiiprice: API para comparação de preços através de notas fiscais de produtos emitidas por estabelecimentos comerciais

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Leonardo Nogueira Matos
Coorientador(a): Thiago Dias Bispo

São Cristóvão – Sergipe

2022

Resumo

No cotidiano, a prática de comparação de preços de produtos em supermercados auxiliam os seus consumidores a terem a melhor economia possível durante uma compra. Entretanto, nem sempre, os métodos empregados nesta prática são eficientes, fazendo com que muitas pessoas se deparem com preços desatualizados ou produtos que não sejam relevantes, seja em panfletos, aplicativos de celulares ou outros meios. Desta forma, esta monografia consiste em modelar e desenvolver a API de comparação de preços que alimenta o LudiiPrice, um aplicativo para dispositivos Android, criado para auxiliar na economia das compras de produtos de atacado e varejo, cuja base de através da varredura do código QR de cupons fiscais. Para conhecer as soluções já existentes no mercado e auxiliar no levantamento dos requisitos da API, um questionário de hábitos de consumo da população e uma pesquisa de mercado foram realizadas. Entre as funcionalidades desenvolvidas, estão o cadastro das notas fiscais, pesquisa de preços de produtos, comparação de gastos nas compras, sugestão de produtos mais comprados por localidade, montagem de lista de compras e históricos de variação de preços de produtos e de compras realizadas pelo usuário. Ao fim deste trabalho foi possível obter uma ferramente que servirá não apenas aos usuários domésticos como também aos donos de pequenos estabelecimentos que realizam compras em grandes centros distribuidores.

Palavras-chave: API. Comparação de preços. Nota fiscal eletrônica.

Abstract

In everyday life, the practice of comparing prices of products in supermarkets helps their consumers to have the best possible savings during a purchase. However, the methods used in this practice are not always efficient, causing many people to come across outdated prices or products that are not relevant, whether in pamphlets, cell phone applications or other means. In this way, this monograph consists of modeling and developing the price comparison API that powers LudiiPrice, an application for Android devices, created to assist in the economy of wholesale and retail product purchases, whose base by scanning the QR code of tax coupons. To learn about the solutions that already exist on the market and assist in the survey of API requirements, a questionnaire on the population's consumption habits and a market survey were carried out. Among the functions developed are the registration of invoices, product price research, comparison of purchase expenses, suggestion of most purchased products by location, assembly of shopping list and history of product price variation and purchases made by the user. At the end of this work, it was possible to obtain a tool that will serve not only domestic users but also the owners of small establishments who make purchases in large distribution centers.

Keywords: API. Price comparison. Electronic invoice.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CNPJ	Cadastro Nacional da Pessoa Jurídica
CPF	Cadastro de Pessoa Física
EAN	*European Article Number*
GTIN	*Global Trade Item Number*
HTML	*HyperText Markup Language*
ICMS	Imposto sobre Circulação de Mercadorias e Serviços
JSON	*JavaScript Object Notation*
MVP	*Minimum Viable Product*
NF-e	Nota Fiscal Eletrônica
NFC-e	Nota Fiscal de Consumidor Eletrônica
NPM	*Node Package Manager*
QR Code	*Quick Response Code*
REST	*Representational State Transfer*
SEFAZ-SE	Secretaria da Fazenda do Estado de Sergipe
SQL	*Structured Query Language*
SOAP	*Simple Object Access Protocol*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifiers*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

Sumário

1 Introdução

A comparação de preços na hora de realizar a compra de algum produto é uma realidade bastante comum na vida de várias famílias consumidoras brasileiras. Uma pesquisa produzida pela Federação do Comércio do Estado do Rio de Janeiro (Fecomércio-RJ), em parceria com o Instituto Ipsos, indica que os preços são um fator decisivo na compra de 82,2% dos brasileiros e o fator qualidade, de 77,1%. Em seguida, constam a marca (17,9%) e o conforto (12,1%) (GRANDA, 2015).

O usuário pode ser motivado a comparar preços por diversos motivos, seja necessidade de economia, relação custo/benefício, entre outros. O consumo também é proporcional ao poder de compra do consumidor. O preço tem significativa relevância no poder de compra, tendo em vista que ele pode redimensionar a renda consoante a sua variação (BERRIOS; SANTOS, 2016). Conforme Gregory (2001), os gastos dos consumidores são proporcionais a sua renda. Aqueles que detém maior renda, consomem e adquirem bens em maior quantidade. Fatores como a inflação, por exemplo, atuam como limitadores do poder de compra da população. Dessa forma, várias famílias precisam recorrer a ações de planejamento e organização para conseguir manter seu consumo no orçamento.

Diariamente, supermercados, mercados ou grades atacados são visitados por consumidores em busca de fazer as compras de seus mantimentos. Nos últimos anos, a tecnologia tem sido uma grande aliada na economia dos consumidores. Diversas são as soluções web ou móveis que permitem a comparação dos preços dos mais diversos produtos e em diversos estabelecimentos (Pinngo[1], Menor Preço Brasil[2], EconomizaClub[3] e Revela Preço[4]).

Este trabalho integra um projeto maior denominado LudiiPrice[5] cujo objetivo é permitir

[1] <http://www.pinngo.com.br/>
[2] <https://play.google.com/store/apps/details?id=br.gov.rs.procergs.mpbr&hl=pt_BR&gl=US>
[3] <http://economiza.club/>
[4] <https://www.revelapreco.com/>
[5] Este título foi sugerido pelo grupo de pesquisa Ludiico, e junta o prefixo 'Ludii' referente ao nome do grupo e a

a comparação de preços através de notas fiscais digitalizadas colaborativamente pelos usuários. Seu desenvolvimento atualmente está dividido em três frentes e cada uma é executada por um aluno de TCC (Trabalho de Conclusão de Curso) do Departamento de Computação da UFS (Universidade Federal de Sergipe). São elas:

- Criação do *Minimum Viable Product* (MVP) de um serviço, aqui denominado de Application Programming Interface (API), responsável pela lógica de armazenamento de informações, controle de preços e sugestões de melhores estabelecimentos para compras. Este corresponde ao objeto de estudo abordado neste TCC;
- Desenvolvimento do aplicativo responsável por captar as notas fiscais e consumir a API disponibilizando seus serviços de maneira amigável;
- Criação de uma base de dados independente de notas fiscais por meio de mineração de produtos e preços na Internet, o que elimina a necessidade do usuário fazer a leitura do código QR de cupons fiscais.

O principal objetivo deste trabalho é a construção da API, ou seja, do *back-end* responsável por fornecer todas as funcionalidades necessárias à interface gráfica do LudiiPrice. Apesar das funções colaborativas que envolvem o projeto maior, a contribuição deste trabalho é voltado para a exploração das informações da nota fiscal enviadas pela aplicação mobile através do processo de *web scrapping*, gerenciamento dos preços dos produtos, sugestão de estabelecimentos com menores preços baseados em geolocalização e fornecimento das informações necessárias para compor gráficos de acompanhamento dos preços dos produtos ao longo do tempo e despesas do usuário.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver uma Interface de Programação de Aplicação (API) que permita o registro de notas fiscais eletrônicas emitidas em estabelecimentos comerciais do Estado de Sergipe, visando identificar as melhores ofertas de preços praticados, garantindo que o usuário obtenha uma maior economia em suas compras.

1.1.2 Objetivos específicos

- Analisar a estrutura de uma nota fiscal eletrônica.
- Especificar os requisitos funcionais e não funcionais do LudiiPrice.

palavra *Price*, de origem inglesa que significa preço.

- Desenvolver uma base de dados que reflita a estrutura de uma nota fiscal eletrônica e os objetivos do LudiiPrice.
- Modelar a API segundo os requisitos listados.
- Desenvolver a API.
- Publicar a API na nuvem para torná-la acessível ao aplicativo *mobile* (MARTINS, 2022) que compõe este projeto pela Internet.

1.2 Metodologia

Para alcançar os objetivos estabelecidos na seção anterior deste trabalho, foram postuladas algumas atividades que permitirão maior compreensão do domínio do problema e ajudarão a modelar a solução a ser desenvolvida. Estas etapas têm seu conteúdo detalhado nos tópicos a seguir.

1.2.1 Análise da estrutura de uma nota fiscal eletrônica

Como o LudiiPrice é alimentado pelas informações das notas fiscais eletrônicas emitidas em estabelecimentos do estado de Sergipe e disponibilizadas pelos usuários, o primeiro passo foi conhecer onde e como obter as notas fiscais, bem como identificar suas estruturas. O processo começou pela coleta das mais diversas notas fiscais, com base em alguns critérios, tais como:

- Notas fiscais emitidas por estabelecimentos do estado de Sergipe;
- Notas fiscais de uma ou mais filiais de um estabelecimento;
- Notas fiscais de diferentes estabelecimentos com um ou mais produtos em comum.

O Código de Resposta Rápida (Código QR) disponibilizado em cada nota foi digitalizada e o link ao qual o usuário é redirecionado foi armazenado. A partir disso, cada nota foi estudada, procurando identificar informações relevantes para o escopo deste projeto, tais como dados gerais do estabelecimento (endereço e CNPJ, por exemplo) e dados dos produtos e preços. A Secretaria da Fazenda do Estado de Sergipe (SEFAZ - SE) disponibiliza em uma seção do seu site eletrônico, uma versão detalhada das notas fiscais, onde são apresentados dados mais completos e em geral relacionados à tributação e comercialização dos produtos. Esta versão da nota foi utilizada para fins de coleta de informações para este projeto, por possuir informações relevantes como o código de Numeração Européia de Artigos (EAN), que permite identificar unicamente um produto em território nacional.

Importante frisar que cada estado lida com a disponibilização da nota fiscal eletrônica de uma forma particular mas em geral mantém um certo padrão de interface e seguem normas

estabelecidas a nível nacional. Ao inspecionar o código fonte de uma Nota Fiscal Comercial Eletrônica (NFC-e) do estado de Sergipe e de São Paulo, por exemplo, é possível perceber diferenças em elementos de divisão da Linguagem de Marcação de HiperTexto (HTML), formato de dados obtido quando consultamos a link da NFC-e. Isso impede, no momento, de expandir a solução apresentada nesta monografia para todos os estados, pois demandaria um esforço maior para lidar com todas as peculiaridades pertinentes às notas fiscais de cada SEFAZ.

1.2.2 Revisão bibliográfica das ferramentas técnicas necessárias para o desenvolvimento da API

Em função do conhecimento prévio por parte do discente deste trabalho, fácil curva de aprendizado e grande disponibilidade de materiais gratuitos na internet, optou-se por empregar o framework Node.js para a construção da API. Uma revisão prévia de determinados conceitos foi realizada, a partir da documentação das principais bibliotecas utilizadas neste projeto.

1.2.3 Modelagem e levantamento de requisitos para a API

Visando identificar o público-alvo deste projeto, o primeiro passo foi aplicar um questionário online sobre hábitos de compras da população. Baseado nele e em reunião com os membros envolvidos no projeto, foi estabelecida a relação de funcionalidades desejáveis para o estabelecimento do Produto Viável Mínimo (MVP). Esta é uma versão mais simples do produto, com o funcionalidades básicas e o emprego da menor quantidade de recursos possível. O processo de modelagem leva em consideração a produção de artefatos importantes para a elaboração de uma estrutura de projetos de software conforme a Linguagem de Modelagem Unificada (UML). Foram elencados os principais artefatos de software a serem produzidos: levantamento de requisitos, casos de uso, estórias do usuário e diagrama de sequência.

1.2.4 Desenvolvimento do MVP deste projeto

De acordo com as funcionalidades levantadas na fase anterior, o MVP foi desenvolvido utilizando a linguagem de programação Typescript em conjunto com o framework Node.js e o banco de dados PostgreSQL. Os testes de validação foram realizados através da ferramenta Insomnia e o MVP foi publicado na Internet para acesso pelo cliente mobile através da plataforma Heroku. Na Tabela 1 constam as versões das tecnologias utilizadas neste MVP.

Tabela 1 – Versões das tecnologias utilizadas no MVP

Tecnologias	**Versão**
Typescript	4.2.4
Node.js	16.3.0
Insomnia	2022.1.0

1.3 Estrutura do Documento

Para facilitar a navegação e melhor entendimento, este documento está estruturado em capítulos e seções, que são:

- Capítulo 2 - Fundamentação Teórica

  Introduz conceitos necessários para entender a monografia, conforme trabalhos de autores relevantes na área.

- Capítulo 3 - Trabalhos Relacionados

  Compreende patentes e a pesquisa de mercado relacionada ao objeto deste trabalho.

- Capítulo 4 - Apresentação e descrição do MVP

  Apresenta o questionário de hábitos de compras da população, artefatos UML, funcionalidades desenvolvidas e o estilo arquitetural.

- Capítulo 5 - Considerações finais

  Desfecho da monografia, retomando os objetivos propostos e apresentando sugestões de trabalhos futuros.

2

Fundamentação Teórica

Este capítulo conta com os principais conceitos abordados neste trabalho, como o contexto de utilização da Nota Fiscal Eletrônica e sua estrutura, importantes para entender os dados manipulados, bem como conceitos acerca das tecnologias usadas na construção da API.

2.1 Nota fiscal eletrônica

Podemos conceituar a Nota Fiscal Eletrônica como sendo um documento de existência apenas digital, emitido e armazenado eletronicamente, com o intuito de documentar, para fins fiscais, uma operação de circulação de mercadorias ou uma prestação de serviços, ocorrida entre as partes (SECRETARIA DA RECEITA FEDERAL, 2021). De 5 a 17 de julho de 2004, titulares das administrações tributárias federal, estaduais, do Distrito Federal e dos municípios de capitais se uniram no 1º Encontro Nacional de Administradores Tributários (ENAT), culminando em protocolos de cooperação técnica para a Nota Fiscal Eletrônica (NF-e). Somente no II ENAT, ao final de Agosto de 2005 que foi assinado um protocolo visando o desenvolvimento e a implantação da NF-e.

A Nota Fiscal Eletrônica proporciona inúmeras vantagens para todos os atores envolvidos em uma transação comercial. Entre elas estão:

- Redução nos custos financeiros da emissão da nota;
- Redução no consumo de papel utilizado para imprimir as notas;
- Redução de custos de armazenagem de documentos fiscais;
- Ganho de produtividade, evitando a digitação das notas fiscais no momento de recepção de mercadorias;
- Redução dos erros de digitação manual nas notas fiscais;

- Aumento na confiabilidade da nota fiscal;

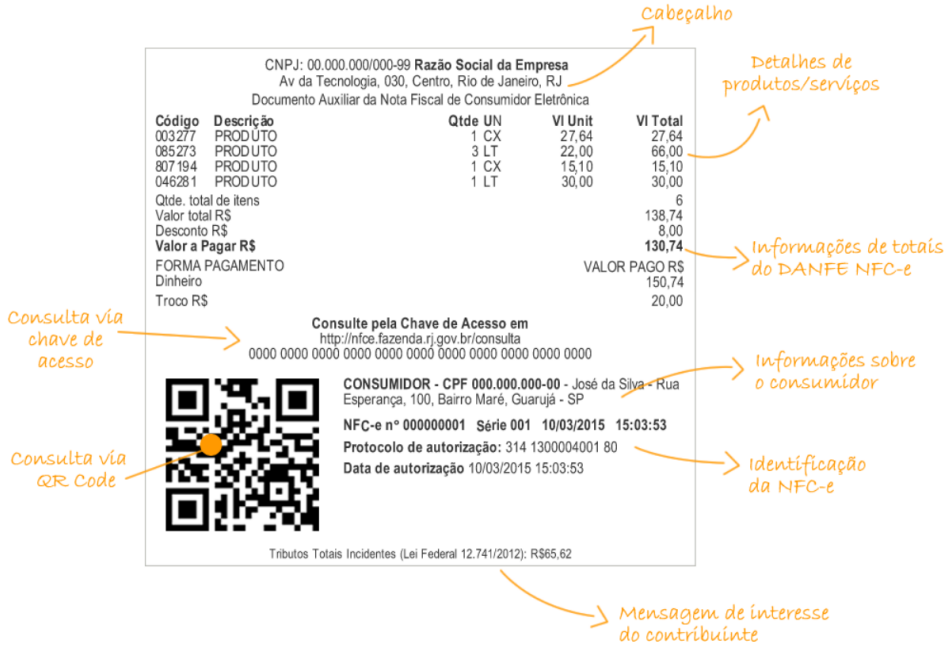
Portanto, a integração e compartilhamento de informações têm o objetivo de racionalizar e modernizar a administração tributária brasileira, reduzindo custos e entraves burocráticos, facilitando o cumprimento das obrigações tributárias e o pagamento de impostos e contribuições, além de fortalecer o controle e a fiscalização por meio de intercâmbio de informações entre as administrações tributárias (SECRETARIA DA RECEITA FEDERAL, 2021).

Para emitir uma nota fiscal, o estabelecimento deve estar credenciado junto a Secretaria da Fazenda (SEFAZ) do estado. Uma vez feito isso, existe uma sequência de etapas até que a nota seja emitida. Inicia-se com a geração de um arquivo XML contendo as informações sobre a empresa e a venda, passa pela assinatura digital, envio ao endereço eletrônico da SEFAZ, validação e autorização, terminando na impressão da nota (TECNOSPEED, 2021). É muito comum ter várias empresas que oferecem serviços especializados de softwares para emissão de notas fiscais.

2.1.1 Estrutura de uma NFC-e

Sempre que algum consumidor efetua uma compra em algum estabelecimento comercial, a exemplo de supermercados e farmácias, ele recebe uma nota fiscal semelhante ao modelo apresentado na Figura 1:

Figura 1 – Modelo de NFC-e



Fonte: Manual de Padrões NFC-e, 2017.

A nota fiscal do consumidor emitida possui uma estrutura previamente definida e padronizada conforme normas específicas, com informações essenciais da transação comercial

efetuada, disposta da seguinte forma:

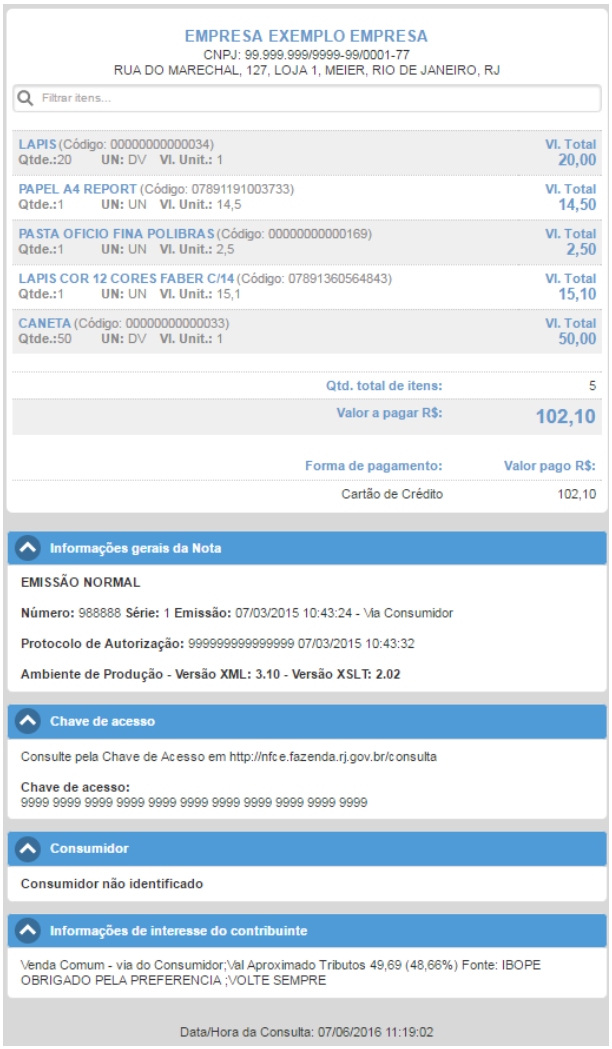
- **Cabeçalho:** contém informações do estabelecimento emissor, como razão social da empresa, cnpj e endereço;
- **Informações de detalhes de produtos/serviços:** apresenta informações dos produtos comercializados nesta transação, incluindo código do produto adotado pelo estabelecimento, descrição do produto, quantidade de unidades do produto adquiridas, unidade de medida do produto, valor unitário do produto e valor total do produto;
- **Informações de totais:** quantidade total de itens, valor total da compra, acréscimos (frete, seguro e outras despesas)/desconto, valor total a pagar (somado aos acréscrimos e subtraído dos descontos), forma de pagamento e valor do troco;
- **Informação de consulta da NFC-e via chave de acesso**: formada por 44 dígitos distribuída em 11 blocos de quatro digitos, a chave de acesso é um identificador único para a NFC-e e através do endereço eletrônico do Portal da Secretaria da Fazenda do estado emissor da nota, é possível realizar uma consulta pública aos dados da NFC-e.
- **Informações de consulta da NFC-e via QR Code**: um QR Code (Código de Resposta Rápida), é um código bidimensional que pode ser digitalizado pela maioria dos smartphones ou tablets através da câmera fotográfica. Esse código, após a decodificação, passa a ser um trecho de texto ou um link redirecionável ao acesso do conteúdo publicado em algum site (PRASS, 2011). Para a NFC-e, o QR Code tem a finalidade de facilitar a consulta aos dados da nota por meio de um aparelho portátil como smartphone, por exemplo. Ao digitalizar, o usuário é redirecionado para a versão resumida da nota fiscal, contendo todos os dados presentes na NFC-e física emitida para o consumidor.
- **Informações do consumidor**: apresenta algumas informações que identificam o consumidor na transação. Pode conter CPF, nome e em alguns casos, endereço. Caso a compra seja inferior a R$ 10.000,00, o consumidor pode optar por não ser identificado na nota, ficando como anônimo na transação
- **Informações de identificação da NFC-e e do protocolo de autorização**: contém número da NFC-e, série, data e hora de emissão da nota, número do protocolo de autorização obtido para NFC-e e a data e hora da autorização;

2.1.2 Consulta pública a NFC-e

Conforme indicado na seção anterior, a NFC-e está disponível para consulta pública, o que garante sua validade e autenticidade. A consulta pode ser realizada de duas formas: inserindo a chave de acesso da nota em endereço eletrônico do Portal Nacional da NF-e ou através da leitura de QR Code impresso na nota utilizando um dispositivo móvel como smartphone. Como resultado

dessa consulta, é apresentado ao consumidor informações da NFC-e de maneira semelhante a presente na versão física, conforme Figura 2.

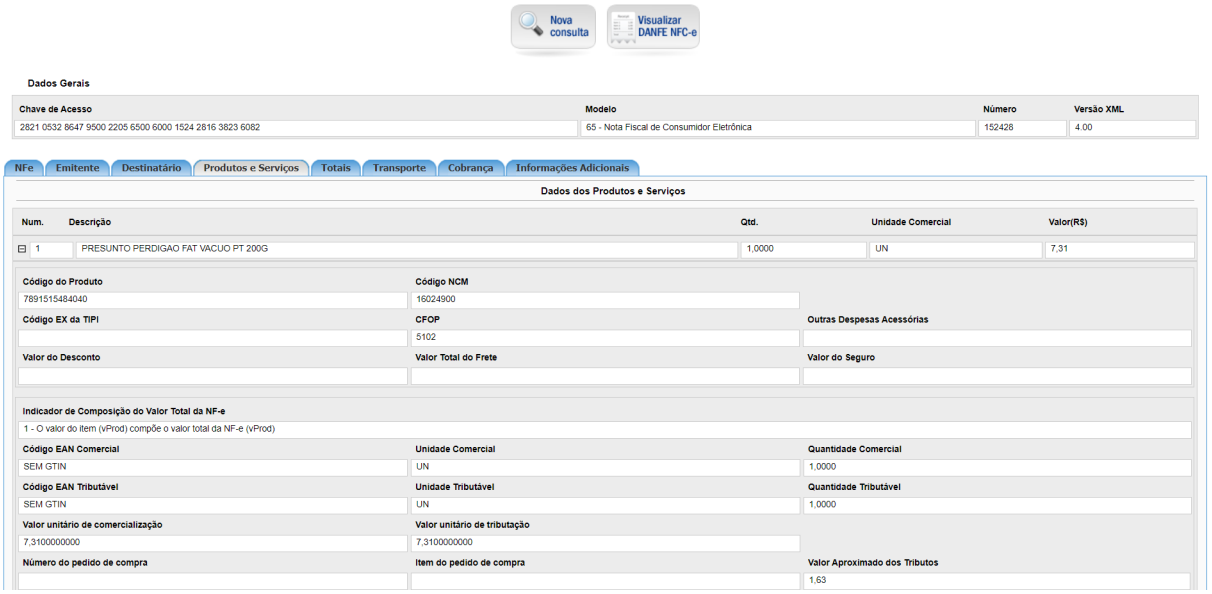
Figura 2 – Modelo de NFC-e digital



Fonte: Manual de Padrões NFC-e, 2017.

A estrutura desta versão da NFC-e é dividida em blocos, onde o primeiro é composto pelas informações do estabelecimento e da compra efetuada, seguido das informações gerais da nota, chave de acesso, informações do consumidor e informações de interesse do contribuinte. No topo da página web contendo a nota fiscal, há a opção de visualizar a NFC-e detalhada, a qual apresenta maior quantidade de informações, muitas delas de cunho tributário, como ICMS, por exemplo, organizados em oito abas, conforme Figura 3.

Figura 3 – Modelo de NFC-e detalhada



Fonte: Portal da NFC-e Eletrônica, 2021.

A aba Produtos e Serviços tem por função mostrar detalhes de cada um dos produtos comprados na transação comercial. Esta aba em específico é bastante relevante para este trabalho, uma vez que nela está presente dois códigos que servirão para identificar unicamente cada produto em território nacional: eanComercial e o eanTributável.

2.1.3 EAN Comercial e EAN Tributável

O código de barras EAN é um número de identificação global de produtos e funciona como um identificador exclusivo para um determinado produto de uma marca específica.. Esse código traz todas as características do produto comercializado, como a descrição do produto, unidade de medida, sabor ou composição, também podendo identificar país de origem e empresa fabricante. Assim, quando um produto sofre alguma alteração, o código também deverá ser alterado para conter as novas características do produto. Essa alteração deverá ser solicitada ao órgão competente (ORIONTEC, 2019). Assim, se uma barra de chocolate ao leite e outra barra de chocolate meio amargo, ambas da marca X e com o mesmo peso, deverão ter um código em específico para cada uma das barras, independente do sabor.

O código EAN é representado através do código de barras GTIN (Número Global do Item Comercial), atribuídos para um produto ou serviço que pode ser precificado, pedido ou faturado em qualquer ponto da cadeia de suprimentos (SECRETARIA DA FAZENDA DO ESTADO DE SÃO PAULO, 2021). Em sua estrutura, um GTIN pode ter diversos tamanhos, variando entre 8, 12, 13 ou 14 digitos, cuja numeração irá variar de acordo com a sua aplicação.

Existem dois tipos de código EAN: o cEAN (código EAN Comercial) e o cEANTrib (código EAN Tributável). O cEAN corresponde ao pacote na forma como está sendo comercializado enquanto que o cEANTrib refere-se a unidade tributável, ou seja, a menor unidade

comercializável identificada por código GTIN (GS1 AISBL, 2021). É comum que o produto descrito na nota possa ser igual à unidade tributável do produto e nesses casos, o cEAN e o cEANTrib serão iguais.

2.2 Conceitos básicos da área de Computação

2.2.1 API

Uma API, conforme define Kurose (2013), é uma interface de programação da aplicação entre a aplicação e a rede, visto que é a interface de programação pela qual as aplicações de rede são criadas. Ou seja, a API age como um mediadora que permite a comunicação entre um cliente e um provedor de dados. Para isso, essa interface define rotinas e protocolos para que esses dados sejam utilizados por outras aplicações.

Uma das arquiteturas mais conhecidas de API é a REST (Transferência Representacional de Estado), introduzida em 2000 pela tese de doutorado de Roy Fielding que especifica um conjunto de princípios de arquitetura. Assim, conforme Fielding (2000), as principais restrições que caracterizam um sistema REST são:

- **Arquitetura cliente-servidor:** separação entre cliente e servidor, garantindo a portabilidade da interface do usuário em várias plataformas e melhorando a escalabilidade;
- **Ausência de monitoramento de estado:** cada solicitação de cliente para servidor deve conter todas as informações necessárias para entender a solicitação, e não pode aproveitar qualquer contexto armazenado no servidor;
- **Cache:** um cache do cliente tem o direito de reutilizar dados de resposta para solicitações posteriores e equivalentes;
- **Interface uniforme:** informações são transferidas de forma padronizada e não específica para as necessidades de um aplicativo. Isso está diretamente associado com quatro restrições: identificação de recursos; manipulação de recursos através de representações; mensagens auto-descritivas; e, hipermídia como o motor do estado de aplicação;
- **Sistema em camadas:** permite que uma arquitetura seja composta de camadas hierárquicas, restringindo o comportamento dos componentes de tal forma que cada componente não possa "ver"além da camada imediata com a qual estão interagindo;
- **Código sob demanda:** permite que a funcionalidade do cliente seja estendida baixando e executando código na forma de applets ou scripts.

Quando um sistema web aplica este princípios, ele é denominado de web service RESTful, onde os dados são recursos que podem ser acessados através de URIs (*Uniform*

Resource Identifiers). Pelo fato de serviços RESTful serem desenvolvidos utilizando a estrutura cliente-servidor, HTTP é o protocolo de comunicação utilizado. A interação entre um cliente e uma API é feita através do envio de diferentes tipos de requisições HTTP. Conforme explica Richardson, Amundsen e Ruby (2013), existem oito tipos de mensagens no padrão HTTP mas somente quatro destas são as mais comumente utilizadas:

- **GET:** obtém a representação de um recurso;
- **POST:** cria um novo recurso com base na representação fornecida;
- **PUT:** substitui o estado de um recurso pelo descrito na representação fornecida;
- **DELETE:** destrói o recurso

Apesar da grande quantidade de restrições para implementar uma API RESTful, ela ainda sim é mais simples que protocolos prescritos como o Protocolo Simples de Acesso a Objetos (SOAP), por oferecer uma implementação flexível, ser simples de entender e permitir uma variedade maior de formatos de dados. Devido ao sistema em camadas, há também garantia de melhor segurança por evitar a comunicação de uma camada com as suas subsequentes.

2.2.2 JSON

Quando um cliente executa uma requisição a uma API RESTful, tanto a solicitação quanto a representação do recurso solicitado é transferida via HTTP utilizando um determinado formato padrão. Um dos formatos mais utilizados atualmente é o JSON, (JavaScript Object Notation), um padrão para representar estruturas de dados em formato de texto. Conforme Pezoa et al. (2016) dado que JSON é uma linguagem que pode ser facilmente compreendido por desenvolvedores e máquinas, tornou-se o formato mais popular para enviar solicitações de API e respostas sobre o protocolo HTTP.

Código 1 - Representação de um determinado tênis em JSON

```
{
        "brand": "Crocs",
        "color": "pink",
        "size": 9,
        "hasLaces": false
}
```

Fonte: (MARS, 2017).

Em JSON, strings são representadas utilizando uma cadeia de caracteres delimitado por aspas duplas, colchetes para arrays e colchetes para descrever objetos. Booleanos são representados com true ou false e números podem ter sua parte fracionária separada com ponto. Um exemplo de uma lista de alunos com suas notas pode ser representado conforme o Código 1 acima.

2.2.3 Node.js

Node.js é uma tecnologia que permite executar código Javascript fora de um navegador web, sendo assim um ambiente de execução *server-side*. Criada em 2009 pelo engenheiro de software Ryan Dahl, sua principal característica é a execução *single-thread*, ou seja, os recursos computacionais são alocados apenas uma vez, sendo esta *thread* (denominada *Event-loop*) a única responsável por executar todo o código (SYED, 2014).

Nas aplicações que executam *multi-thread*, toda nova requisição é entregue a uma nova *thread* responsável por tratá-la. Esse modelo, apesar de eficiente, acaba por demandar uma alta quantidade de recursos, algumas vezes sem necessidade. Esses recursos, por serem limitados, acabam por bloquear novas requisições quando o limite é alcançado, justamente pela falta de recursos.

Na *thread Event-loop*, cada requisição é tratada como um evento. Ao passo que faz isso, a *thread* continua lidando com novas requisições que vão surgindo na pilha de eventos (SYED, 2014). Por meio de chamadas de I/O (entrada e saída) não-bloqueantes, as operações de entrada e saída são executadas de forma assíncrona e a *thread* não precisa aguardar que a operação seja finalizada para dar continuidade a outra. Essa estratégia elimina as filas de processamento, tornando o processo mais eficiente.

Somado as demais vantagens citadas, o Node.js também conta com o NPM (*Node Package Manager*), gerenciador de pacotes do Node.js, que detém um enorme acervo de bibliotecas que podem ser acopladas ao projeto, oferecendo soluções para serem utilizadas nas mais diversas situações. Um dos pacotes mais conhecidos do NPM para Node.js foi utilizado neste projeto:

- **Express:** lançado em novembro de 2010, é um framework para aplicações web e APIs, destacando-se pelo grande leque de recursos.

2.2.4 Web scraping

Web scraping é uma técnica de coleta das informações de páginas web, convertendo-as em um determinado formato para utilização posterior. Esta mineração pode ser feita de forma manual mas no geral, são empregados métodos automatizados que permitirão coletar grande volumes de dados com maior eficiência, economizando tempo e esforço (BROUCKE; BAESENS,

2018). Algumas das aplicações do *web scraping* são: serviços de monitoramento de preços, pesquisa de mercado, empresas de mídia, jornalismo e mercado financeiro.

O processo de mineração das informações inicia com a identificação do(s) site(s) que detém os dados. Uma vez coletadas as URLs desejadas, uma requisição HTTP é realizada a fim de obter os arquivos fonte da página. Para este propósito podem ser utilizadas ferramentas específicas para localizar e extrair dados específicos e em seguida salvar as informações em formatos como JSON, por exemplo.

2.2.5 Unified Modeling Language - UML

A UML nasceu a partir do esforço conjunto entre Grady Booch, James Rumbaugh e Ivar Jacobson na tentativa de unificar metodologias de análise e projetos orientados a objetos no desenvolvimento de softwares (FOWLER, 2003). Essa linguagem é empregada na representação padronizada de um sistema, obtendo uma percepção clara para todas as partes envolvidas no desenvolvimento.

Os diagramas UML auxiliam tanto arquitetos de software quanto desenvolvedores no entendimento e na construção das aplicações. O uso de determinados diagramas dentro da vasta coleção disponibilizada pela UML depende de fatores como o sistema, usuários e detalhes do que se deseja modelar (IBM, 2021b). Alguns destes diagramas foram empregados na modelagem do software foco deste trabalho e são apresentados nas seções a seguir.

2.2.5.1 Diagrama de Casos de Uso

Este diagrama ajuda a entender a forma como os agentes interagem com o sistema, a partir da descrição dos passos necessários para atingir um determinado objetivo (PRESSMAN; MAXIM, 2014). O diagrama é composto por casos de uso, atores e relacionamentos.

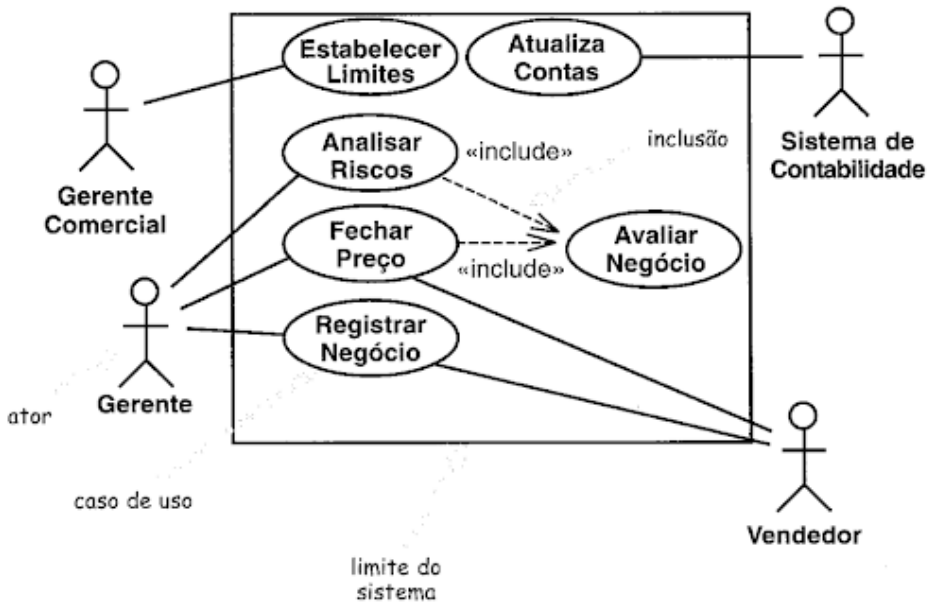
- **Atores:** representam o papel desempenhado por um usuário no sistema. Pode ser uma pessoa, uma organização ou até mesmo outro sistema. No diagrama, são representados com o desenho de uma pessoa utilizando apenas linhas.
- **Casos de uso:** descreve uma funcionalidade desempenhada por um ator no sistema. São representados através da descrição do requisito dentro de uma elipse.
- **Relacionamentos:** representam conexões entre os componentes do diagrama. Os mais comuns são:
  - **Associação**: expressa a conexão existente entre o caso de uso e o ator. É representada por uma linha reta.
  - **Generalização/Especialização:** relação na qual um elemento filho recebe características e comportamentos de um elemento pai mas que ainda apresenta suas

características específicas. Pode ocorrer entre casos de uso ou entre atores. No diagrama, é representado por uma seta que parte do elemento mais específico (filho) e aponta para o elemento mais geral (pai).

– **Inclusão:** indica que a execução de um caso de uso obriga a execução de outro. É representado por uma seta aberta com linha tracejada apontando para o caso de uso incluso. Além disso, anexa-se o termo «include» à seta.

– **Extensão:** indica que a execução de um caso de uso é opcional com relação à outro, a partir da satisfação de determinadas condições. É representado por uma seta aberta com linha tracejada que aponta para o caso de uso do qual se estende. Além disso, anexa-se o termo «extend» à seta.

Figura 4 – Exemplo de diagrama de casos de uso



Fonte: Fowler (2003)

A Figura 4 apresenta um exemplo de diagrama de caso de uso, representando seus principais componentes.

2.2.5.2 Diagrama Entidade Relacionamento

A modelagem do banco de dados é considerada uma fase importante no planejamento de uma aplicação. Um dos modelos de dados mais utilizados nesta fase é o Diagrama Entidade Relacionamento (DER), que descreve a diagramação das informações do domínio de negócio utilizando alto nível de abstração (SILBERSCHATZ; SUNDARSHAN; KORTH, 2016).

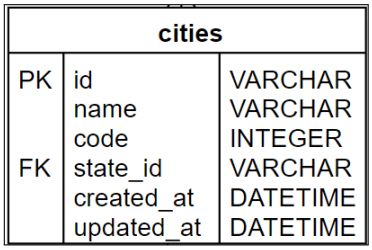
Os principais componentes de um DER são: entidades, relacionamentos e atributos:

- **Entidade:** representa algo no mundo real cuja existência é independente (ELMASRI; NAVATHE; PINHEIRO, 2009).
- **Relacionamentos**: associação entre várias entidades (SILBERSCHATZ; SUNDARSHAN; KORTH, 2016). Por exemplo, uma a entidade nota fiscal pode conter um ou mais produtos.
- **Atributos**: propriedades específicas das entidades (ELMASRI; NAVATHE; PINHEIRO, 2009). Por exemplo, cada entidade usuário contém um atributo nome.

2.2.5.2.1 Representação gráfica do DER

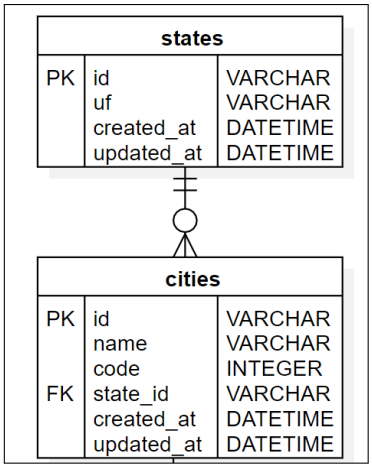
Existem várias abordagens para representar graficamente um DER. Para este projeto, foi escolhida a notação *Crow's foot* (ou no português, pé de galinha), de James Martin. Neste modelo, as entidades são representadas em blocos retangulares, contendo o nome na parte superior e os atributos na parte inferior, conforme Figura 5.

Figura 5 – Exemplo de entidade no DER



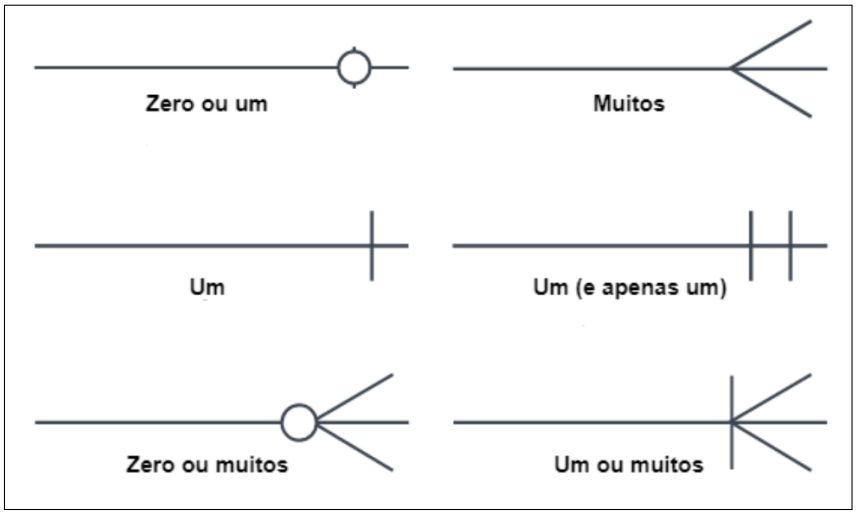
Os relacionamentos são representado por linhas que interligam as entidades, de acordo com a Figura 6.

Figura 6 – Exemplo de relacionamento no DER



As extremidades das linhas mudam conforme o tipo de relacionamento, conforme a Figura 7 descreve.

Figura 7 – Tipos de relacionamentos do DER

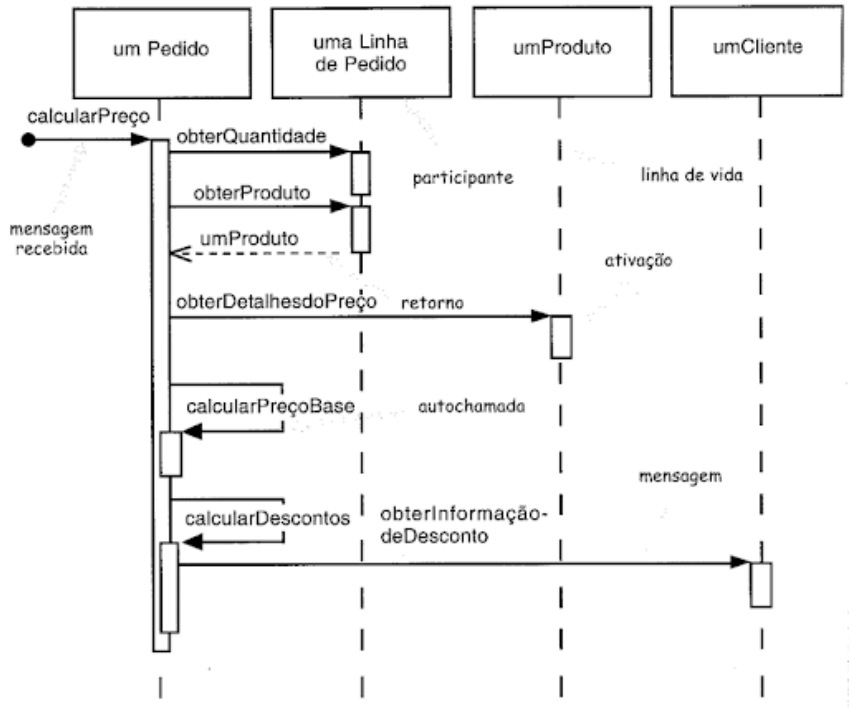


Fonte: Lima (2017)

2.2.5.3 Diagrama de Sequência

Os diagramas de sequência são utilizados para ilustrar as comunicações entre objetos em uma interação (IBM, 2021a). Eles mostram a ordem temporal de envio das mensagens pelos objetos envolvidos na realização de uma tarefa. Esta temporalidade é demonstrada através de linhas tracejadas partindo destes objetos, chamadas de linhas da vida.

Figura 8 – Exemplo de diagrama de sequência



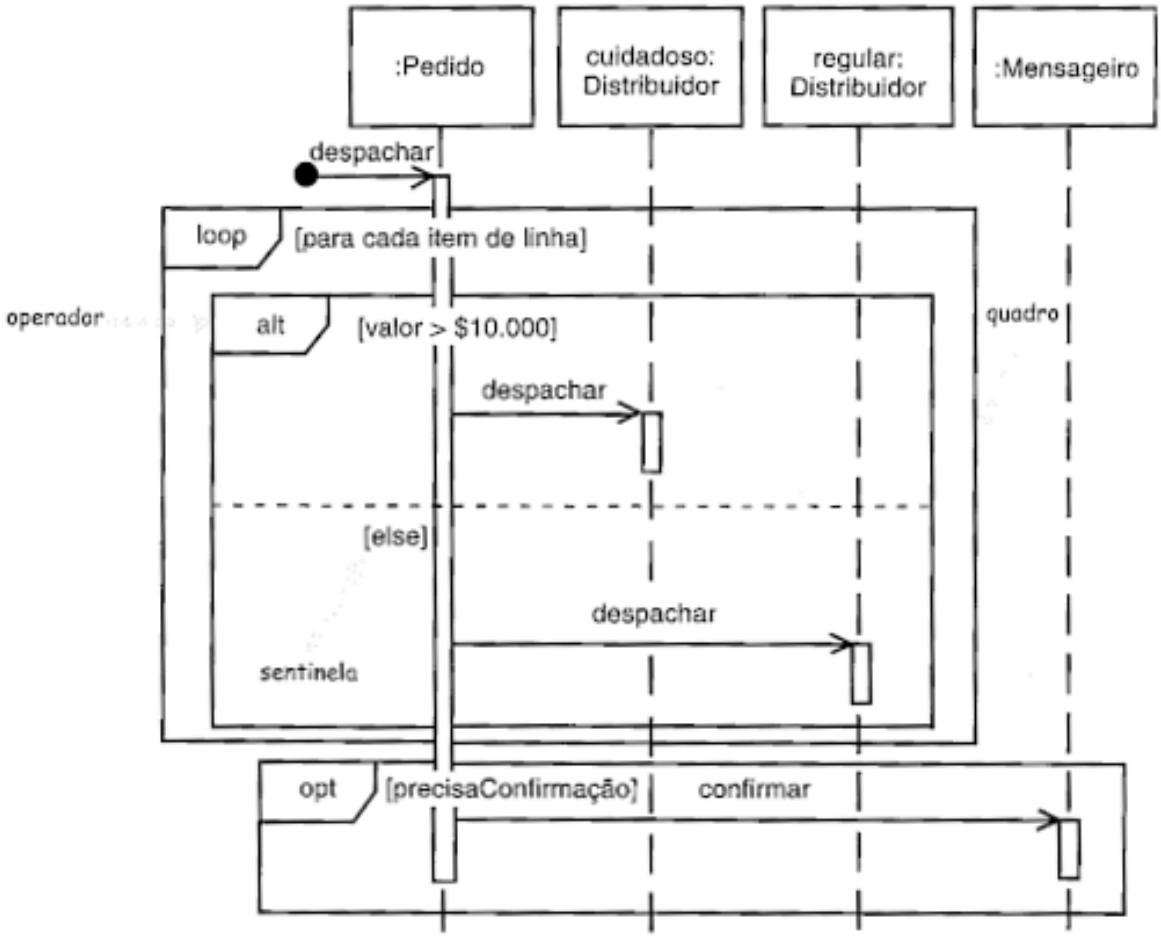
Fonte: Fowler (2003)

A Figura 5 apresenta um exemplo deste diagrama. Cada retângulo ao topo da linha de vida

corresponde a um objeto. Quando um objeto está executando determinada chamada de função, é exibida uma barra branca abaixo da linha de vida, chamada de barra de ativação (FOWLER, 2003). Esse recurso irá indicar quando um determinado objeto participa da interação. A chamada de uma função é representada pela linha preenchida que parte de uma barra de ativação para outra, rotulada com o nome da função e, opcionalmente, com seus parâmetros. O retorno de uma função é indicada por uma linha tracejada também rotulada com o nome do objeto retornado. Por exemplo, na Figura 5, a chamada da função obterProduto() retorna o objeto umProduto.

Caso haja a necessidade de representar estruturas de controle como loops e condicionais, por exemplo, é possível incluir um componente chamado quadro de interação, retângulos que cercam parte do diagrama e são nomeados com o tipo de estrutura de controle que representam (PRESSMAN; MAXIM, 2014).

Figura 9 – Exemplo de quadro de interação em diagrama de sequência



A Figura 6 demonstra um exemplo da aplicação de três estruturas de controle: *loop* (repetição), *alt* (condicional) e *opt* (opcionais).

2.3 Fórmula de Haversine

A fórmula Haversine, importante equação utilizada em navegação, calcula a distância mais curta entre dois pontos em uma esfera usando suas latitudes e longitudes medidas ao longo da superfície. Haversine é uma fórmula muito popular e frequentemente usada no desenvolvimento de uma aplicação do Sistema de Informação Geográfica (GIS) ou na análise de caminhos e campos. O termo Haversine (haverseno) foi criado em 1835 pelo astrônomo e matemático James Inman e é representado pela seguinte função trigonométrica:

$$hav(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

No sistema cartesiano, uma coordenada é medida pelo cálculo da distância do ponto central. Mas, considerando o sistema de coordenadas de latitude-longitude, a coordenada é calculada como o ângulo central do equador (BRUMMELEN, 2012). O ângulo central é composto por duas semi retas que atravessam a circunferência em pontos distintos e cujo vértice é o centro da circunferência. Considere um ângulo central Θ entre dois pontos quaisquer em uma esfera:

$$\Theta = \frac{d}{r}$$

A fórmula acima expressa a relação entre o ângulo central (Θ), a distância entre dois pontos (d) e o raio de uma esfera (r). Para determinarmos a distância entre duas coordenadas no planeta, precisamos saber o valor do ângulo central e o raio da esfera. A esfera, neste caso, é o planeta Terra, cujo raio é de, aproximadamente, 6371 km. A fórmula de haversine permite que o haversine de Θ (ou seja, hav (Θ)) seja calculado diretamente a partir da latitude (representada por φ) e longitude (representada por λ) dos dois pontos:

$$hav(\Theta) = hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)hav(\lambda_2 - \lambda_1)$$

Resolva d aplicando o haverseno inverso ou usando a função arcoseno (seno inverso):

$$d = 2r \arcsin\left(\sqrt{\sin^2(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Esta não é a medida exata porque a fórmula assume que a Terra é uma esfera perfeita quando na verdade é uma esferoide achatada.

3
Trabalhos Relacionados

A finalidade deste capítulo é apresentar aplicações relevantes que dialogam com o objetivo do trabalho proposto neste documento, no que diz respeito à área de pesquisa de preços e economia em compras. A busca por trabalhos relacionados permite entender o que já existe no mercado, e os casos de fracasso e sucesso na tentativa de aprimorar o MVP proposto.

3.1 Metodologia de busca

Para guiar a pesquisa, buscou-se entre o período de Abril de 2021 e Junho de 2021 localizar tais produtos na Google Play Store, loja oficial da Google que dispõe de um vasto acervo de aplicativos para sistemas Android, que inclusive motivou a escolha desta base como domínio de busca para esta seção. Além disso, buscou-se também registros de patentes na área, utilizando grandes bases de patentes.

3.1.1 Critérios de seleção

Com a finalidade de filtrar os resultados obtidos nas bases de pesquisa, foram formulados alguns critérios de triagem, levando em conta a proposta deste trabalho. Os critérios adotados foram:

- Critério 01 - A aplicação deve possuir uma plataforma mobile.
- Critério 02 - A aplicação tem seu funcionamento voltado para o território brasileiro.
- Critério 03 - A aplicação ou patente foi publicada nos últimos 10 anos.
- Critério 04 - Para o Google Play Store, deve considerar apenas os resultados exibidos na primeira página, em razão do grande volume retornado pela busca.

As strings de busca foram aplicadas em três bases de patentes (INPI, WIPO e Espacenet) e constam na Tabela 1. Como o serviço de busca do Google Play Store não possui um sistema de busca avançado, foram introduzidas frases mais objetivas.

Tabela 2 – Bases de dados e strings de busca aplicados

Base de dados	**String de busca**
INPI	“comparação de preço”
WIPO	price comparison AND (app OR mobile) AND (invoice OR tax invoice OR receipt)
EspaceNet	"price comparison"AND mobile AND (grocery OR groceries) AND receipt
Google Play Store	“comparação de preço” “price comparison”

3.1.2 Critérios de análise

Feita a busca nas bases de dados, os resultados das bases de patentes tiveram seus resumos e títulos lidos, enquanto que os resultados da Google Play Store tiveram as descrições lidas. Nessa fase de análise dos resultados, levou-se em conta os critérios de análise a seguir:

- Critério 01 - A aplicação tem como objetivo fornecer primariamente ao cliente o menor preço de um produto comparado a vários estabelecimentos;
- Critério 02 - A aplicação tem os preços dos produtos alimentados através de notas fiscais das compras de usuários, diretamente ou indiretamente;
- Critério 03 - A aplicação registra o histórico de preços dos produtos ao longo do tempo;
- Critério 04 - A aplicação possui preços de produtos atualizados nos últimos três meses;
- Critério 05 - A aplicação permite ao usuário selecionar uma localização ou utilizar geolocalização para situar localmente os preços;

A investigação sob a ótica dos critérios de análise reduziram a quantidade de resultados finais, conforme apresentado na Tabela 2. Com isso, os resultados das bases de patentes não foram relevantes, restando apenas aqueles originados da Google Play Store.

Tabela 3 – Quantidade de resultados da busca antes e depois da filtragem

Base de dados	**Quantidade de resultados na busca**	**Quantidade de resultados após filtragem**
INPI	0	0
WIPO	3	0
EspaceNet	1	0
Google Play Store	49	4

3.2 Aplicativos relacionados

Cada aplicativo selecionado após filtragem foi instalado e testado para verificar diretamente cada uma das funcionalidades, adicionando sempre o mesmo conjunto de notas fiscais e procurando pelos mesmos produtos, afim de garantir a confiabilidade deste teste. O resumo das principais funcionalidades encontradas está presente na Tabela 4 e nas subseções seguintes, as funcionalidades são detalhadas para cada aplicativo selecionado.

Tabela 4 – Principais funcionalidades encontradas nos produtos

	EconomizaClub	**Pinngo**	**Revela Preço**	**Melhor Preço Brasil**
Cadastro dos preços por meio da nota fiscal	X	X	X	X
Utiliza Código QR como entrada de dados	X	X	X	(código de barras)
Registra histórico de preços	X	-	-	-
Permite comparar preços entre estabelecimentos	X	X	X	-
Permite montar lista de compras	X	X		-

3.2.1 Menor Preço Brasil

Desenvolvido pela Receita Estadual do Rio Grande do Sul, o aplicativo permite consultar o preço de produtos em diferentes estabelecimentos. Através das consultas às Notas Fiscais Eletrônicas (NF-e) e às Notas Fiscais de Consumidor Eletrônicas (NFC-e), as informações são atualizadas em tempo real toda vez que um estabelecimento realiza uma venda a varejo, sem necessidade do usuário entrar com estas informações. Entre as outras principais funcionalidades, permite verificar a localização dos estabelecimentos com menor preço, ver o histórico de preços para cada produto e consultar um produto ao escanear o código de barras. O aplicativo está disponível em boa parte dos estados do Brasil. Para a maioria dos produtos pesquisados, o preço dos produtos era recente.

3.2.2 Pinngo

O aplicativo tem os preços de seus produtos alimentados por notas fiscais inseridas através do QR Code presente nas notas, sendo assim, colaborativo. Além da comparação de preços em estabelecimentos próximos da localidade do usuário ou em localidade escolhida por ele, o aplicativo também permite ao usuário montar sua lista de compras, ver informações de notas fiscais anteriores e pesquisar produtos por código de barras ou nome.

3.2.3 Economiza Club

Restrito ao estado do Rio Grande do Sul, o aplicativo segue a linha colaborativa de outros aplicativos, através da alimentação de preços por scan da nota fiscal usando QR Code. Além da comparação de preços em estabelecimentos próximos do usuário, o aplicativo também permite montar listas de compras e consultar histórico de preços. O preço dos produtos estava defasado e a aplicação parecia não receber atualizações a bastante tempo.

3.2.4 Revela Preço

Além da função colaborativa de inserção das notas fiscais via QR Code, o aplicativo também permite criar sua lista de compras e, diferente dos outros aplicativos, também permite compartilhar com familiares e amigos. Importante frisar que aqui, toda nota fiscal inserida é colocada em análise por um tempo para só depois ser de fato computada. Um outro ponto interessante é um incentivo, onde para algumas cidades, a cada R$ 30,00 em compras o usuário ganha 1 ponto e ao juntar certas quantidades de pontos, pode trocar por produtos em estabelecimentos parceiros. Para a maioria dos produtos pesquisados, o preço dos produtos era atual.

3.3 Considerações do capítulo

A maioria dos aplicativos analisados apresentam características semelhantes no que tange o meio de cadastro dos preços dos produtos e em permitir comparar preços entre estabelecimentos. O EconomizaClub e o Pinngo apresentaram a maior quantidade de funcionalidades principais utilizadas como parâmetros nesta análise.

Com relação a atualização dos preços, metade dos aplicativos apresentava preços recentes enquanto o Economiza Club e Pinngo estavam defasados, o que revela um risco comum aos serviços que dependem exclusivamente dos usuários colaborarem com a digitalização das notas fiscais para renovar os preços dos produtos. Neste sentido, o Menor Preço Brasil se destaca por atualizar as informações sempre que uma NFC-e é emitida (através de parceria direta com as Secretarias da Fazenda dos estados).

Com exceção do EconomizaClub, todos os aplicativos carecem de informações gráficas acerca da evolução dos preços ou da comparação dos preços em estabelecimentos próximos aos usuários. O LudiiPrice, entretanto, além de oferecer as funcionalidades básicas presentes nos demais serviços, se preocupa em contornar esse *deficit*.

4

Apresentação e descrição do MVP

Esta seção tem o propósito de apresentar o processo de construção da API do LudiiPrice, desde o levantamento de requisitos e produção de artefatos até a descrição das funcionalidades desenvolvidas.

4.1 Análise de hábitos de compras da população

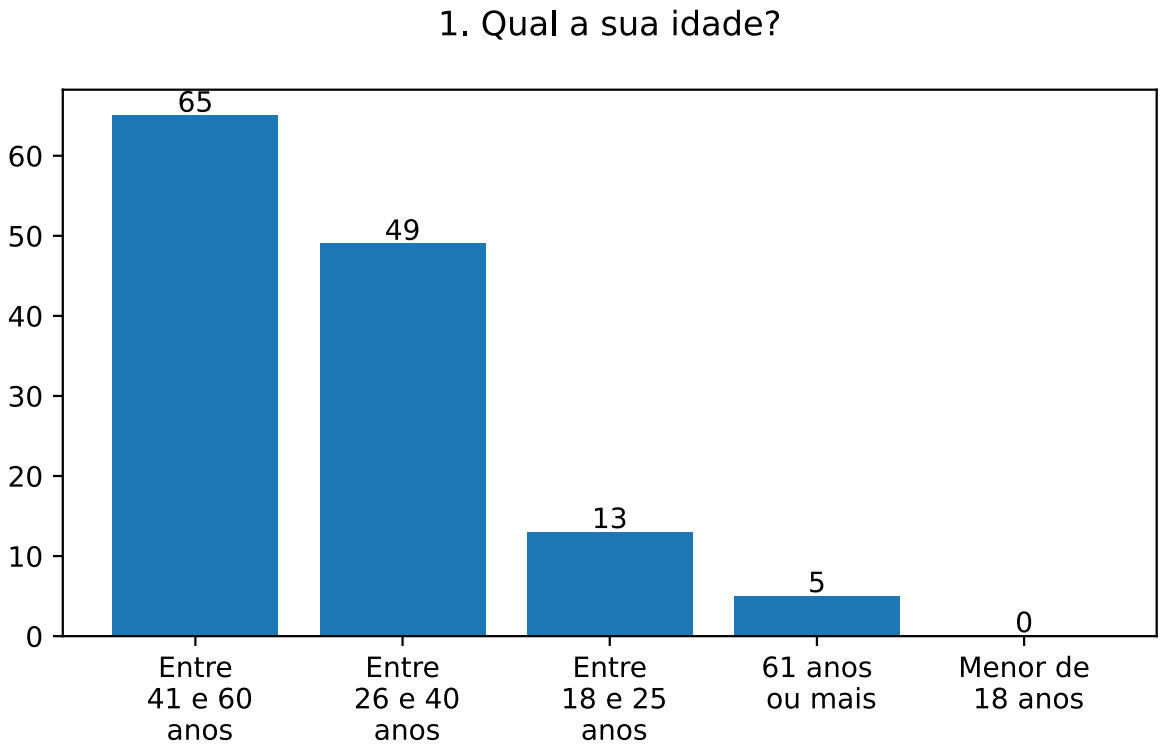
Com o objetivo de conhecer o comportamento dos consumidores de modo a auxiliar no levantamento dos requisitos desta aplicação, foi realizada uma pesquisa de hábitos de compras da população. Esta pesquisa foi realizada em parceria com a Coordenação de Inovação e Transferência de Tecnologia da Universidade Federal de Sergipe (CINTTEC-UFS), que teve o papel fundamental de divulgação do questionário disponilizado através da plataforma Formulários Google.

O questionário possuia oito questões de múltipla escolha, divididas entre objetivas e abertas, onde o participante poderia dar mais detalhes de forma livre. A pesquisa foi realizada entre os os dias 26 de Agosto de 2021 e 7 de Novembro de 2021, e obteve 132 respostas.

Esta seção se concentrará em realizar observações pontuais sobre os resultados do questionário. A versão completa da pesquisa pode ser encontrada em <https://sites.google.com/view/ludiiprice/questionário>.

Conforme Figura 10, quase metade dos entrevistados é composta por pessoas na faixa etária entre 41 e 60 anos (49,2%), seguida por 37,1% correspondente aos participantes entre 26 e 40 anos.

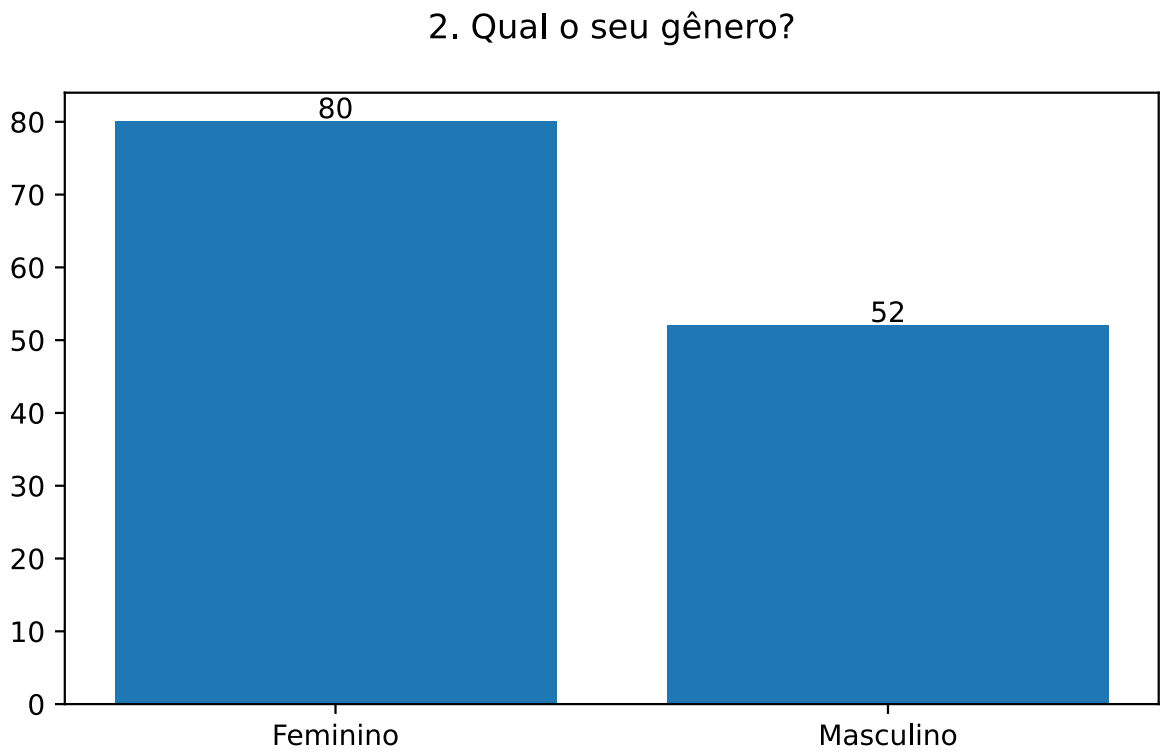
Figura 10 – Divisão dos entrevistados por faixa de idade



Fonte: Ludiico (2022b)

Na Figura 11, pessoas o gênero feminino representam 60,6% dos participantes, seguidas por 39,4% do gênero masculino.

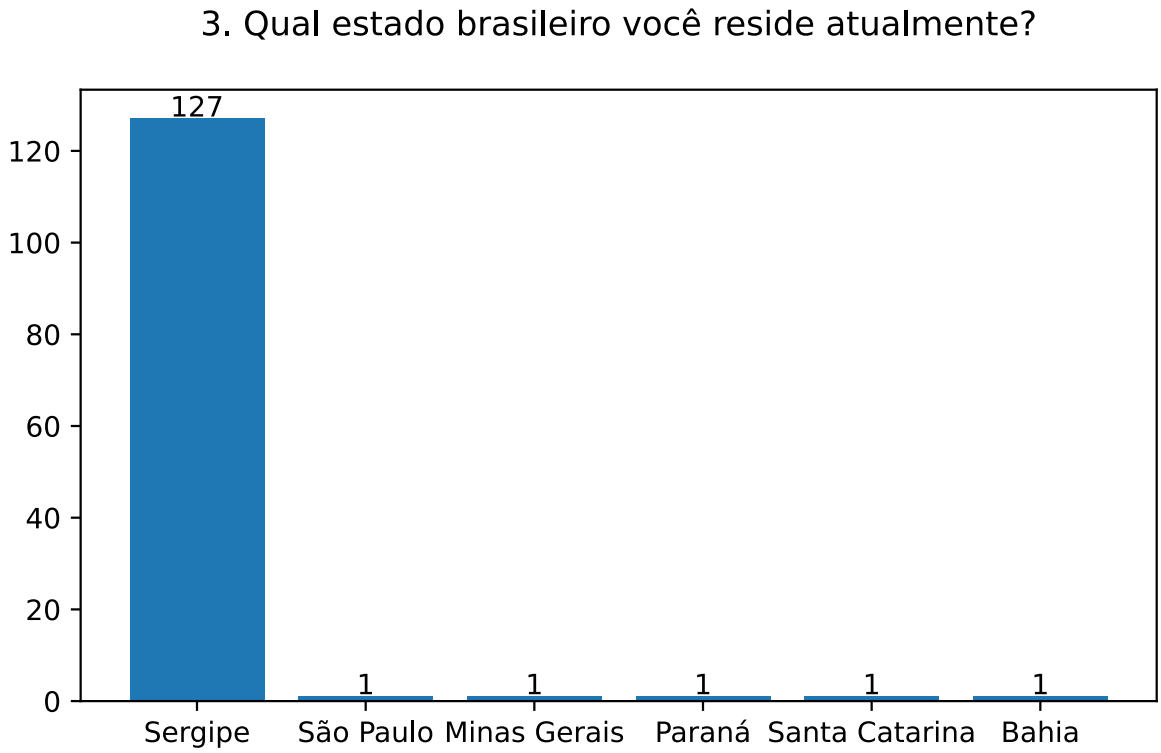
Figura 11 – Divisão dos entrevistados por gênero



Fonte: Ludiico (2022b)

Cerca de 96,2% dos participantes eram residentes do estado de Sergipe, de acordo com a Figura 12.

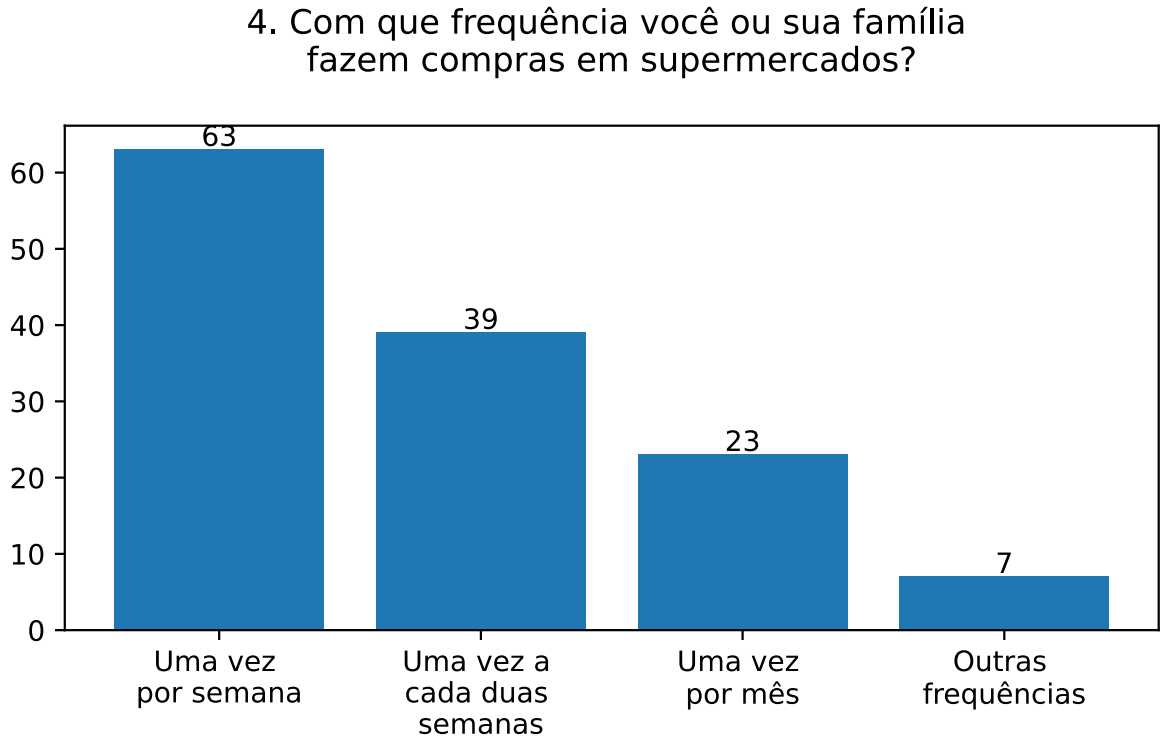
Figura 12 – Divisão dos entrevistados por UF



Fonte: Ludiico (2022b)

Na Figura 13, os participantes foram perguntados com relação a frequência de realização de compras em supermercados. A maioria (47,7%) realizam compras mensais, seguidos por 29,5% que realizam compras quinzenais e 17,4% que realizam compras duas vezes na semana.

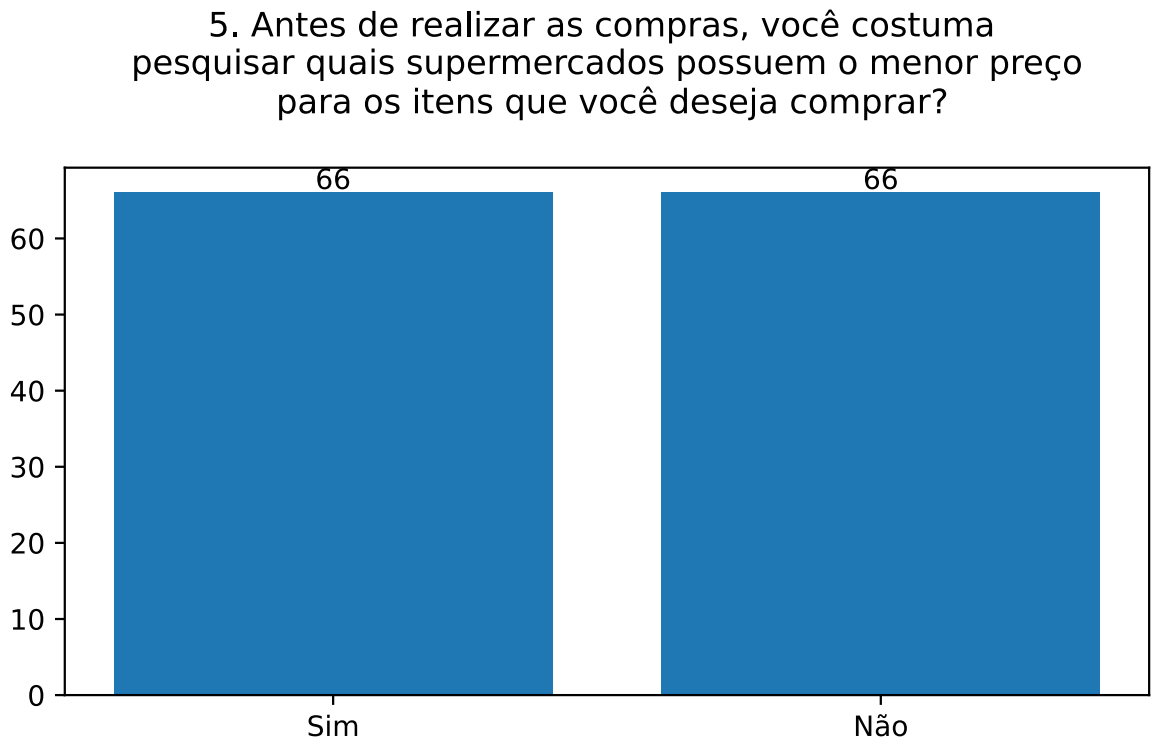
Figura 13 – Frequência com que os entrevistados fazem compras no supermercado



Fonte: Ludiico (2022b)

Entre os entrevistados, não há unanimidade na prática de comparação de preços, visto que metade realiza esta prática, enquanto a outra metade não, como pode ser visto na Figura 14.

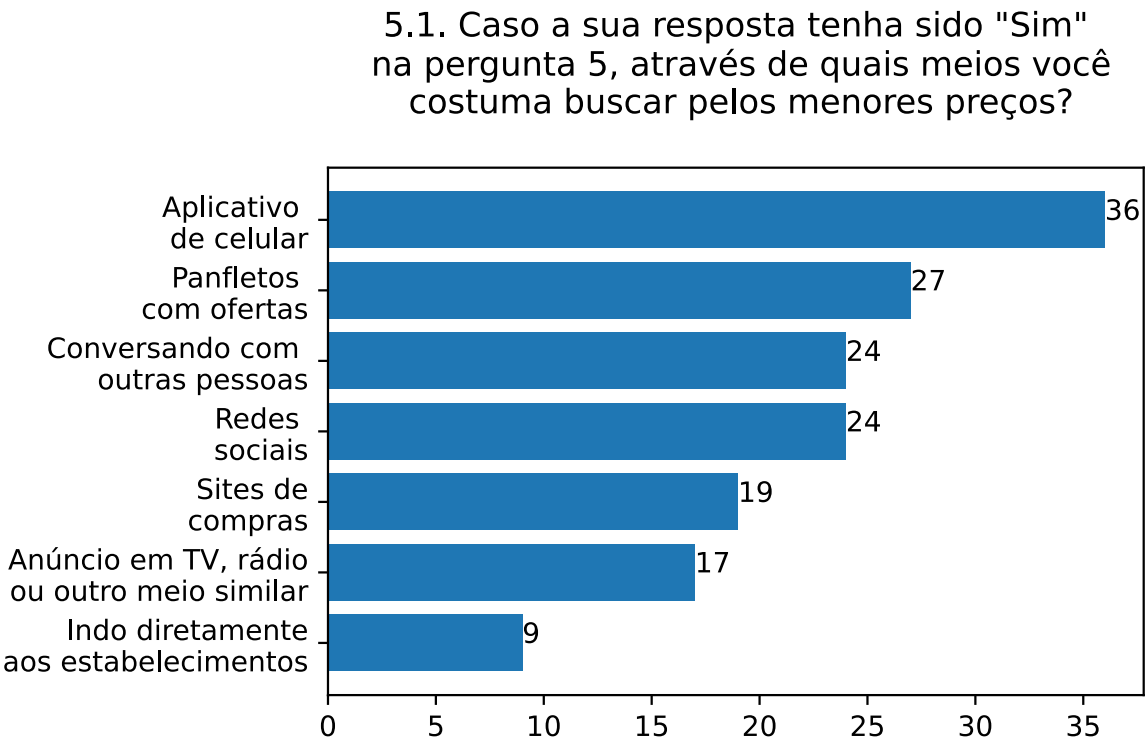
Figura 14 – Divisão dos entrevistados com relação ao hábito de buscar o menor preço



Fonte: Ludiico (2022b)

Aos participantes que responderam "Sim"na pergunta anterior, foi questionado sobre os meios que utilizam para realizar a pesquisa de preços. Conforme a Figura 15, 52,2% asseguraram utilizar aplicativos de celular, seguido por 39,1% que o fazem através de panfletos de ofertas.

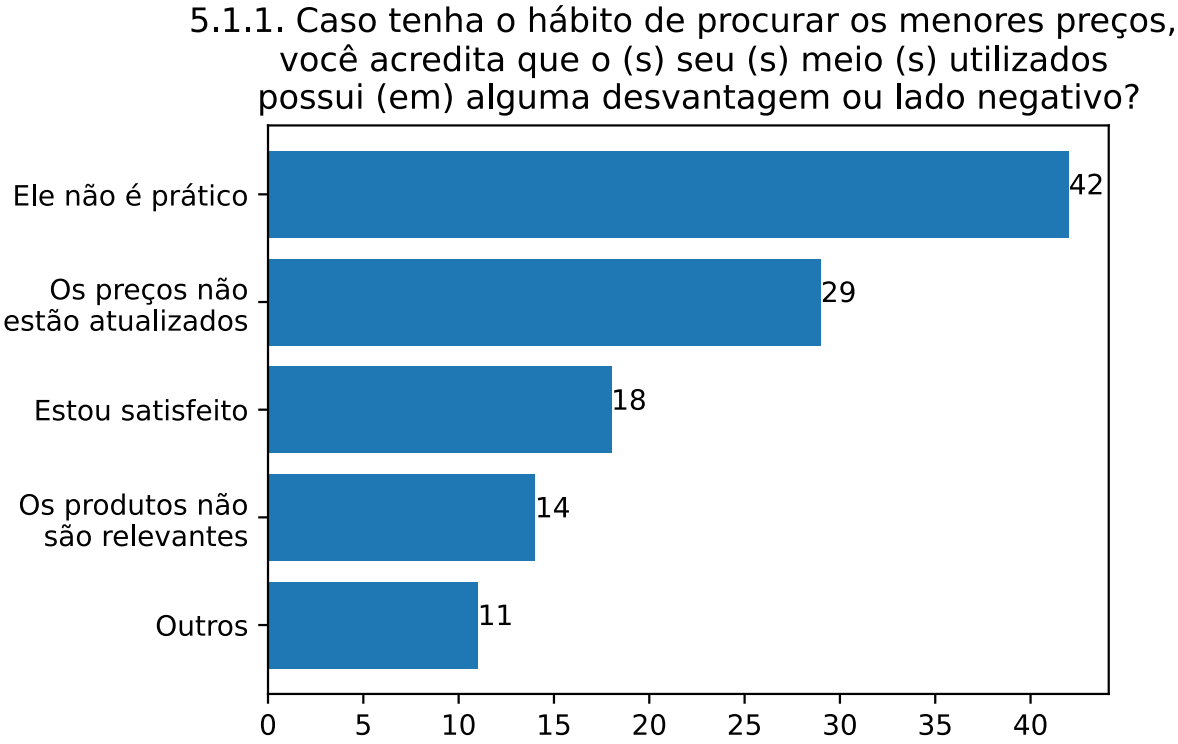
Figura 15 – Meios mais utilizados pelos entrevistados para buscar preços



Fonte: Ludiico (2022b)

Ainda para aqueles que responderam "Sim"na pergunta de número 5, foram questionados sobre as desvantagens dos meios que utilizam para fazer a pesquisa dos preços, conforme Figura 16. A maioria (36,5%) respondeu não haver praticidade, seguidos de 25,2% que consideram que os preços não estão atualizados.

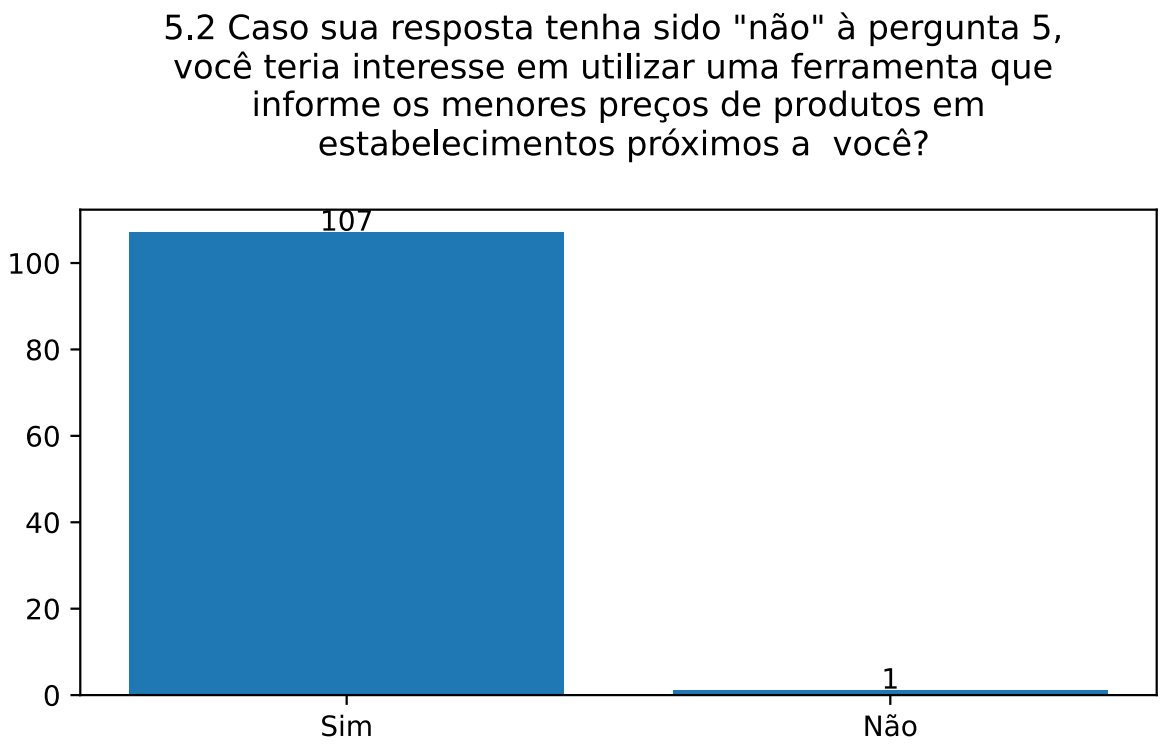
Figura 16 – Desvantagens dos métodos utilizados para comparação de preços



Fonte: Ludiico (2022b)

Por fim, conforme a Figura 17, a maioria (99,1%) respondeu afirmativamente quando perguntados sobre o interesse em uma ferramenta que informe os menores preços de produtos em estabelecimentos próximos

Figura 17 – Interesse dos entrevistados em ferramentas de comparação de preços



Fonte: Ludiico (2022b)

4.2 Histórias dos usuários

Após o questionário apresentado na Seção 4.1 ajudar a identificar o público-alvo deste MVP, é necessário mapear o que essas pessoas fazem ou necessitam fazer, determinando que utilidade o software pode ter para elas. Uma das formas de fazer esta tarefa é através das histórias dos usuários, uma breve declaração de intenção que descreve algo que o sistema precisa fazer para o usuário (LEFFINGWELL, 2011). A descrição deve ser generalizada, empregando uma linguagem informal.

As histórias foram mapeadas com base no resultado do questionário e em discussões envolvendo os discentes que atuam na construção do LudiiPrice, o orientador e o co-orientador. Levou-se em conta também pontos fortes e fracos das aplicações já existentes no mercado.

A Tabela 5 das histórias dos usuários é divida em três colunas: "Como", representando o sujeito que quer realizar uma determinada ação, "Quero", descrevendo a intenção desta *persona* e "Para", que indica a motivação por trás da ação, o benefício desejado ou o problema que precisa ser solucionado.

Tabela 5 – Histórias do usuário

Como	**Quero**	**Para**
Consumidor.	Pesquisar por um produto e saber o melhor preço.	Economizar em minhas compras.
Consumidor.	Ver as minhas compras anteriores.	Saber o quanto gastei nas últimas compras que fiz.
Consumidor	Montar uma lista de compras.	Saber o preço dos produtos da lista em diferentes estabelecimentos.
Consumidor	Ver o histório de preços de um produto	Saber se o seu preço aumentou ou diminuiu com o passar do tempo.

4.3 Levantamento de requisitos

Uma vez definidos as histórias dos usuários, é o momento de definir quais os requisitos do sistema. Diferente do primeiro, os requisitos são mais específicos, o que permite capturar também detalhes técnicos de implementação de uma funcionalidade ou particularidades de como o negócio funciona, por exemplo.

Os subtópicos a seguir apresentam os requisitos funcionais, não funcionais e as regras de domínio, por tabelas, dispondo a identificação do requisito e a sua descrição.

4.3.1 Requisitos funcionais

Os requisitos funcionais representam as funcionalidades da aplicação, aquilo que ela se propõe a solucionar, serviços que oferece ou funções que o software deve desempenhar. Os requisitos presentes na Tabela 6 foram agrupados em funcionalidades, dispostas da seguinte forma:

- **Funcionalidades de autenticação:** RF1 até o RF5.
- **Funcionalidades de produtos:** RF6 até o RF10.
- **Funcionalidades de lista de compras:** RF11 até o RF17.
- **Funcionalidades de notas fiscais:** RF18 até o RF21
- **Funcionalidades do menu de configurações:** RF22 até o RF23

A estrutura de identificação do código do requisito é representado no formato "X0Y", onde Y estabelece dependência em relação ao requisito X.

Tabela 6 – Requisitos funcionais do sistema

Identificação	**Descrição**	**Ator**
RF1	O usuário não autenticado cadastra uma conta.	Usuário não autenticado
RF2	O usuário não autenticado entra com uma conta.	Usuário não autenticado
RF3	O usuário não autenticado redefine a senha de sua conta.	Usuário não autenticado
RF4	O usuário altera a senha de sua conta.	Usuário
RF5	O usuário sai da sua conta.	Usuário
RF6	O sistema deve exibir uma lista contendo os produtos mais comprados próximos ao usuário.	Sistema
RF7	O usuário busca por um produto avulso.	Usuário
RF8	O usuário visualiza os detalhes de um produto.	Usuário
RF9	O sistema deve exibir o histórico de preços de um produto.	Sistema
RF10	O sistema deve calcular a distância de um estabelecimento a partir da localização do dispositivo do usuário.	Sistema
RF11	O usuário consulta as suas listas de compras.	Usuário
RF12	O usuário cria uma lista de compras.	Usuário
RF13	O usuário exclui uma de suas listas de compras.	Sistema
RF14	O usuário adiciona produtos à sua lista de compras.	Sistema
RF15	O usuário remove produtos da sua lista de compras.	Sistema
RF16	O usuário visualiza detalhes da sua lista de compras.	Usuário
RF17	O sistema deve exibir um gráfico comparativo de preços de produtos pertencentes à lista de compras do usuário.	Usuário
RF18	O usuário lê o código QR de uma nota fiscal.	Usuário
RF1801	O sistema deve armazenar as informações de uma nota fiscal cujo código QR foi lido.	Sistema
RF19	O usuário exclui uma nota fiscal eletrônica de sua conta.	Usuário
RF20	O usuário visualiza detalhes de uma nota fiscal eletrônica.	Usuário
RF21	O sistema deve exibir um gráfico comparativo de preços dos produtos pertencentes à nota fiscal eletrônica	Sistema

Continua na próxima página

Tabela 6 – continuação da página anterior

Identificação	**Descrição**	**Ator**
RF22	O usuário define os seus filtros de busca	Usuário
RF23	O usuário concede permissões do dispositivo necessárias para o funcionamento do aplicativo.	Usuário

4.3.2 Requisitos Não Funcionais

Os requisitos não funcionais representam restrições acerca das funcionalidades do sistema, expressando condições ou especificidades que o software deve atender, como os da Tabela 7.

Tabela 7 – Requisitos não funcionais do sistema

Identificação	**Descrição**
RNF1	O sistema deve executar em dispositivos móveis com sistema operacional Android.
RNF2	O sistema utilizará o SGBD *PostgreSQL*.
RNF3	O sistema será implementado no ambiente do *Heroku*.
RNF4	O sistema deve implementar controle de acesso para todas as funcionalidades.
RNF5	O sistema deve aguardar um tempo máximo de 9 segundos após realizar uma requisição.
RNF6	A interface *mobile* do sistema utilizará o *framework Flutter*.
RNF7	A interface *mobile* do sistema utilizará a linguagem de programação Dart.
RNF8	O servidor do sistema utilizará o *framework* Node.js.
RNF9	O servidor do sistema utilizará a linguagem de programação Typescript.

4.3.3 Regras de Domínio

As regras de domínio na Tabela 8 referem-se a características específicas do negócio com a qual o serviço lida ou do problema que se propõe a resolver.

Tabela 8 – Regras de domínio do sistema

Identificação	**Descrição**
RD1	O código QR de uma nota fiscal só será aceito caso esta possua uma data de emissão de no máximo 90 dias.
RD2	Uma nota fiscal só pode ser armazenada uma única vez na base de dados da aplicação.
RD3	Uma nota fiscal lida será associada como uma compra do usuário que a leu.
RD4	O sistema deve armazenar notas fiscais do estado de Sergipe.

Continua na próxima página

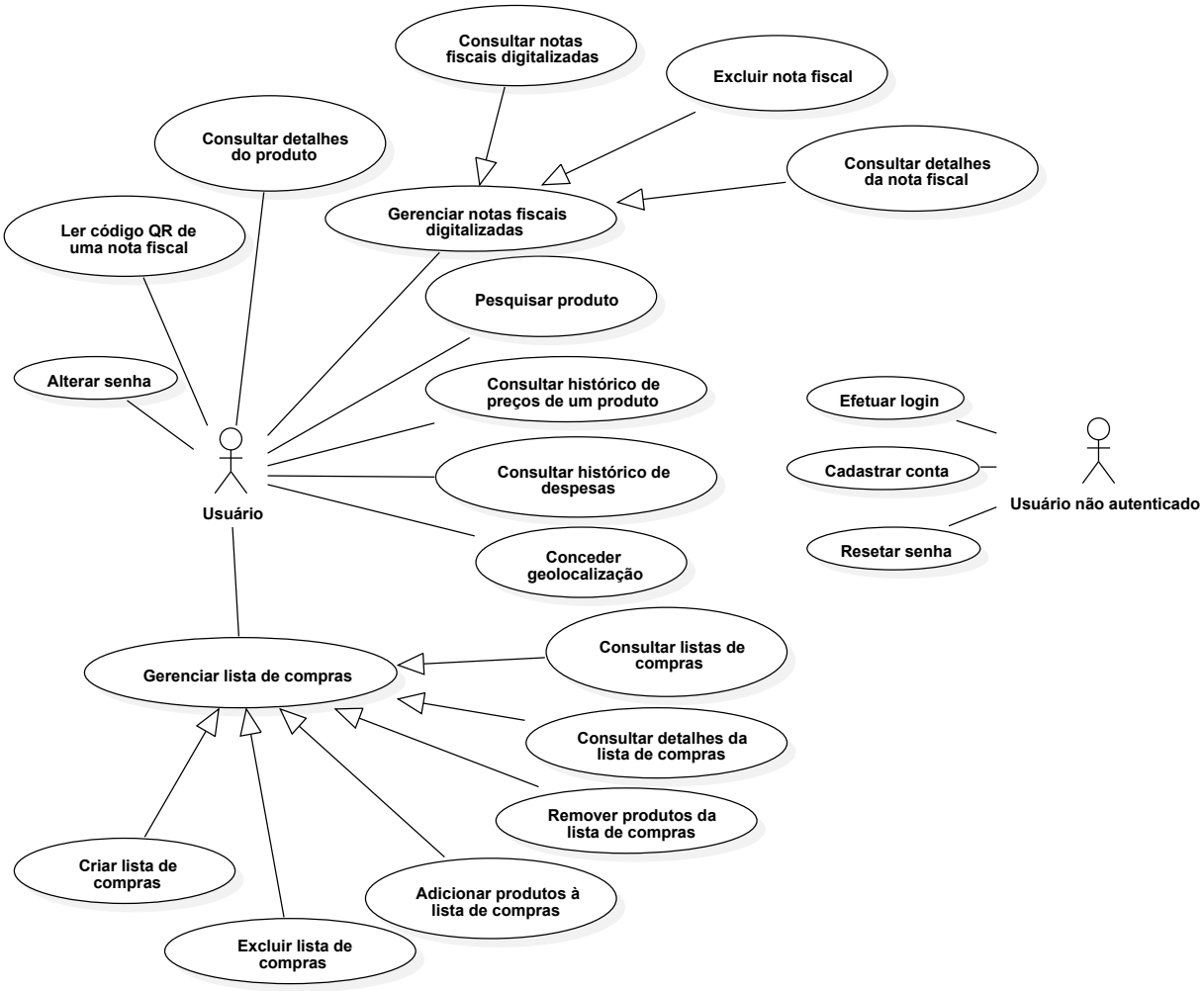
Tabela 8 – continuação da página anterior

Identificação	**Descrição**
RD5	O filtro de distância “sem limites” deve ter o valor igual ao diâmetro do planeta Terra, ou seja, 12,742Km.
RD6	A exibição das informações de um produto sempre será baseada na geolocalização do dispositivo do usuário.

4.4 Casos de uso

Utilizando os requisitos levantados na Seção 4.3, foi possível construir os casos de uso da Figura 18, que identifica principais funcionalidades e os relacionamentos estabelecidos entre elas.

Figura 18 – Diagrama de casos de uso do LudiiPrice



Fonte: Ludiico (2022a)

Além dos casos de uso apresentados na Figura 5, também há dois atores, os quais detém

os seguintes papéis:

- **Usuário autenticado:** principal usuário da aplicação, acessa as principais funcionalidades da API.
- **Usuário não autenticado:** ator que ainda não possui uma conta de usuário ou que não possui acesso a ela no momento.

4.5 Diagrama Entidade Relacionamento

A partir dos artefatos produzidos nas seções anteriores, é possível traçar as principais entidades e os relacionamentos existentes entre elas, além de especificar as características das propriedades de cada dado da entidade, conforme Figura 19.

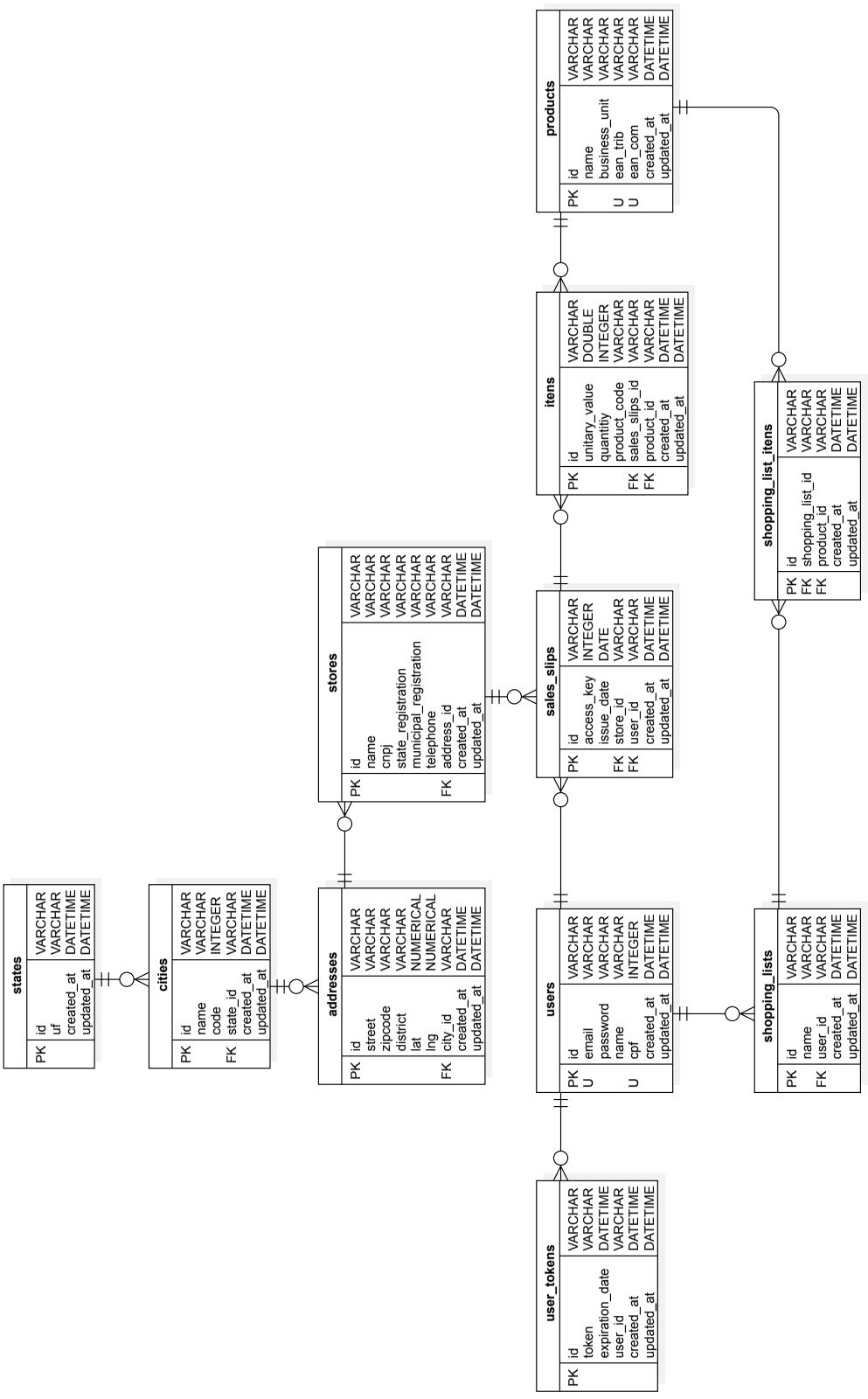


Figura 19 – Diagrama entidade relacionamento da API do LudiiPrice

4.6 Diagrama de Sequência

Em função da complexidade da etapa de inserção das notas fiscais, foi necessário descrever o processo através do diagrama de sequência da Figura 20. Nesta interação, diversos objetos são invocados para realizar a tarefa, tornando difícil entender o processo apenas com uma mera descrição.

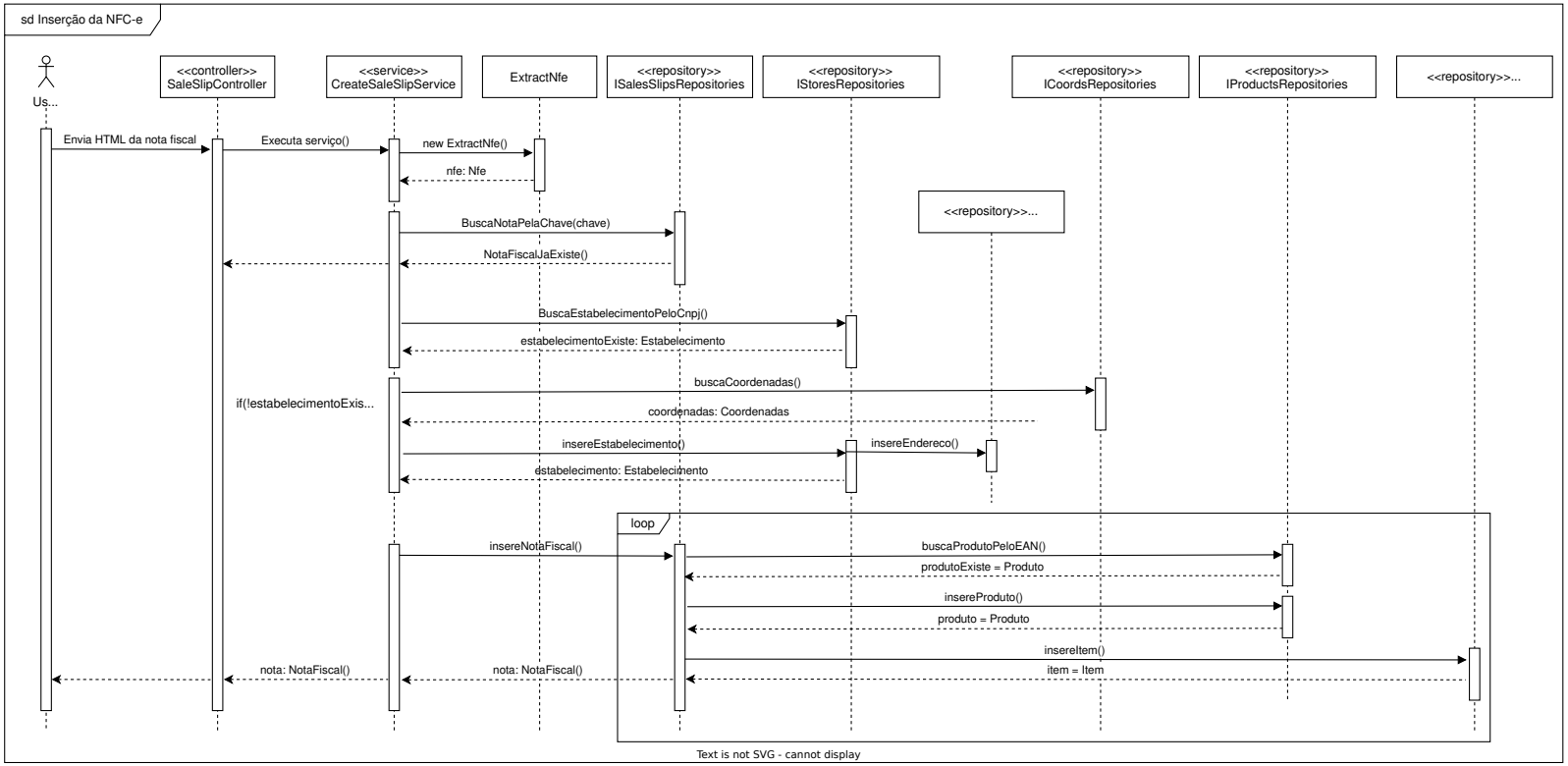


Figura 20 – Diagrama de sequência para inserção das notas fiscais

4.7 Funcionalidades da API

Esta seção trata das funcionalidades desenvolvidas nesta API, descrevendo as características, rotas e o conteúdo das requisições HTTP.

4.7.1 Cadastro e autenticação

Para conseguir acessar qualquer serviço ofertado, o usuário precisa estar logado. Isso significa que, previamente, é necessário possuir um cadastro registrado na base de dados do sistema. Caso não possua, o usuário pode realizar o cadastro acessando o serviço através de uma requisição HTTP POST da Tabela 9, informando em formato JSON, os atributos de email, senha e nome completo. A partir daí, o serviço no servidor avalia os dados recebidos, verificando por exemplo se o email informado é válido ou se este mesmo email já está cadastrado no sistema.

Tabela 9 – Dados da requisição para cadastro de usuário

URI	**/users**
Método	POST
Corpo da requisição	name, email e password

Em caso de sucesso, um provedor é acionado para enviar um email de confirmação do cadastro para o usuário. Este provedor trata-se de um módulo Node.js que permite enviar emails utilizando o protocolo SMTP para transporte de mensagens definindo apenas host, porta e detalhes de autenticação. No caso deste MVP, preferiu-se utilizar uma conta *Gmail*, serviço da *Google*, para servir como o principal *host* de email da API.

Cada email é acompanhado de um *token* individual com prazo de expiração de três horas a partir da emissão. Este *token* é acoplado como parâmetro de uma URL também presente no corpo do email, que ao ser acessada, irá redirecionar para uma página web, responsável por enviar uma nova requisição HTTP POST da Tabela 10 para o servidor que confirmará o registro do usuário.

Tabela 10 – Dados da requisição para confirmação do cadastro de usuário

URI	**/account/confirmation**
Método	POST
Query da requisição	token

Uma vez registrado, o usuário poderá realizar *login* informando email e senha em uma requisição HTTP POST da Tabela 11. Em caso de sucesso, o usuário receberá um objeto JSON contendo os dados do usuário, seguidos de um objeto composto por um token JWT e seu respectivo tempo de expiração. O *token* deve ser armazenado pelo cliente e inserido no campo *Authorization* do cabeçalho das futuras requisições à rotas que necessitam de autenticação para serem acessadas. O *token* permitirá identificar quem está solicitando uma determinada requisição

além de constatar se o solicitante possui permissão para acessar um determinado recurso do servidor, impedindo que usuários não cadastrados possam acessar os serviços da API.

Tabela 11 – Dados da requisição para autenticação do usuário

URI	**/sessions**
Método	POST
Query da requisição	email, password

4.7.2 *Reset* e redefinição de senhas

O *reset* de senha refere-se à situação em que o usuário esqueceu a sua senha e portanto, precisa cadastrar uma nova. Para isto, o cliente precisa enviar uma requisição HTTP POST da Tabela 12 ao servidor, informando o email do usuário que deseja fazer a operação. O servidor, ao receber a requisição, enviará uma mensagem para o email informado, com uma URL que direciona para uma página web onde o usuário poderá digitar a nova senha seguida da confirmação da mesma. Uma vez que o processo ocorre com sucesso, a nova senha entra em vigor.

Tabela 12 – Dados da requisição para reset de senha do usuário

URI	**/password/forgot**
Método	POST
Query da requisição	email

A redefinição de senha ocorre quando o usuário encontra-se logado em algum dispositivo que consome a API (no caso do LudiiPrice, trata-se de um aplicativo móvel) e deseja alterar a sua senha. Neste caso de uso, o usuário está ciente de sua senha antiga. As informações da senha antiga, a senha nova e a confirmação da senha nova devem ser enviadas em uma requisição HTTP PATCH da Tabela 13.

Tabela 13 – Dados da requisição para redefinição de senha do usuário

URI	**/password/change**
Método	PATCH
Query da requisição	old_password, new_password, new_password_confirmation

4.7.3 Cadastro de notas fiscais

O cadastro de uma nota fiscal na base de dados do LudiiPrice permitirá que os dados armazenados sejam convertidos em informações que serão consultadas por diversos serviços do servidor. Para isto ser feito, os dados da nota fiscal devem ser consultados, extraídos, processados e armazenados na base de dados.

O primeiro passo desse processo consiste no cliente prover um meio de ler o código QR da NFC-e, obtendo a URL da nota detalhada disponível na página web da SEFAZ-SE que contém

os detalhes da compra, como informações do estabelecimento emissor, produtos comprados e seus respectivos valores, por exemplo. Isso geralmente pode ser feito utilizando alguma biblioteca própria do *framework* de implementação do cliente.

Uma vez que o cliente acessa a página web contendo a NFC-e detalhada, é necessário enviar o conteúdo HTML da página para a API. Essa requisição de acesso à SEFAZ-SE é realizada pelo próprio cliente para evitar que hajam múltiplas requisições partindo da mesma máquina, ou seja, do servidor deste MVP, o que poderia ser considerado pela SEFAZ-SE como tentativas de sobrecarregar seus servidores tornando seu serviço indisponível, comum em ataques de força bruta e *DDoS*. Portanto, essa alternativa acaba evitando que o servidor da API tenha seu acesso bloqueado temporariamente ou permanentemente, prejudicando este MVP.

Como a *string* HTML obtida da página web da NFC-e detalhada na maioria dos casos é muito grande, possuindo múltiplos espaços e caracteres, optou-se por converter essa *string* em base64 antes de realizar a requisição HTTP POST da Tabela 14 para a API, realizando o cadastro da nota fiscal desejada. Além disso, é necessário também passar como parâmetro a URL da nota fiscal para verificar de que unidade federativa aquela nota foi emitida, bloqueando a inserção de notas que não foram emitidas no estado de Sergipe.

Tabela 14 – Dados da requisição para cadastro da NFC-e

URI	**/salesslip**
Método	POST
Necessita de token de autenticação?	Sim
Parâmetros da requisição	url, html_file

Visto que as informações da requisição são recebidas pelo servidor, a *string* em base64 é decodificada, transformando-se novamente em *string* HTML. O processo de *web scraping* é aplicado ao corpo da página web, localizando as informações do estabelecimento emissor (nome, CNPJ e endereço, por exemplo) e detalhes da compra (produtos, quantidade e preços). Nessa etapa, é utilizada a biblioteca para *Node.js* chamada *Cheerio*, que procurará por determinados elementos HTML a partir de seletores predefinidos. Para defini-los, foi necessário estudar previamente o código HTML de alguns exemplos de notas fiscais da SEFAZ-SE em busca de determinar o padrão presente. Os seletores em geral determinam os caminhos a serem percorridos até chegar em um elemento HTML que contém a informação desejada.

Para este MVP, considerou-se implementar o recebimento de notas fiscais apenas do estado de Sergipe pois apesar de possuir a mesma estrutura visual, as páginas web das Secretarias da Fazenda de diferentes estados diferem na implementação do código, a nível da estrutura do corpo HTML. Isso exige um esforço maior no mapeamento das propriedades e dos seletores HTML. Além disso, expandir a disponibilidade do serviço para outros estados exigiria ter em mãos uma boa quantidade de notas fiscais advindas destes estados para que sejam feitos os testes para verificar se o serviço foi implementado corretamente com mínimos problemas, observando se os dados necessários para o MVP estão sempre disponíveis ou as diversas variações de textos

nas páginas web. Neste sentido, reduzir o escopo deste MVP permite entregar funcionalidades mais coesas e com o mínimo de problemas possíveis.

Após o *web scraping*, as informações coletadas passam por um processamento de modo a reduzir problemas como múltiplos espaços em branco, por exemplo e também para serem montados objetos correspondentes às entidades do banco de dados. O processo de inserção das informações ocorre conforme o diagrama de sequência da Figura 20, na seção 4.6. Se o estabelecimento não estava registrado previamente, ele é cadastrado. O mesmo acontece com os produtos da nota. Em seguida os dados dos itens da compra (valor e quantidade, por exemplo) são adicionados.

Caso haja um problema durante o registro, toda a transação do banco é desfeita e a API retorna uma mensagem de erro correspondente. Erros como código QR inválido ou erro ao resgatar as informações da página web da SEFAZ-SE são os mais comuns nesta rota. Em caso de sucesso, os dados da nota são retornados em formato de objeto JSON.

4.7.4 Monitoramento da localização do usuário

Como parte dos requisitos deste MVP, existem funcionalidades que visam a utilização de geolocalização, de modo a personalizar o retorno das informações baseadas na posição atual do usuário. Tais funcionalidades são referenciadas nos requisitos RF6, RF7, RF8, RF9, RF17 e RF21.

Sempre que uma requisição é feita a um *endpoint* relacionado a alguma destas funcionalidades, o cliente é obrigado a enviar três parâmetros através da *query*:

- **Localização do usuário:** valores numéricos das coordenadas de latitude e longitude do usuário no local onde a requisição foi realizada;
- **Raio de busca dos estabelecimentos:** numeral decimal representando o raio em quilômetros na qual os estabelecimentos serão buscados;
- **Quantidade de estabelecimentos:** número inteiro representando a quantidade de estabelecimentos considerados para a busca.

A partir desta informação, o servidor verifica quais estabelecimentos estão no raio de busca definido, calculando a distância entre a localização do usuário e a localização dos estabelecimentos a partir das coordenadas armazenadas na base de dados. Essa distância é calculada a partir da fórmula de Haversine aplicada diretamente na *query* ao banco de dados. O método responsável por esta operação retorna uma lista com a distância e o *id* de cada estabelecimento, informações que serão utilizadas posteriormente para montar *queries* ao banco de dados para cada caso de uso específico.

4.7.5 Consulta de notas fiscais do usuário

Todas as notas inseridas podem ser consultadas posteriormente pelo cliente através de uma requisição HTTP GET da Tabela 15. A rota retorna um *array* de objetos com as informações principais das notas (sem os detalhes).

Tabela 15 – Dados da requisição para consulta das notas fiscais de um usuário

URI	**/salesslips/user/:user_id**
Método	GET
Parâmetros da requisição	user_id

Cada nota pode ser consultada individualmente por uma requisição HTTP GET da Tabela 16 utilizando o id da nota como parâmetro na URL. Esta rota retorna um objeto JSON com os detalhes da nota, como informações do estabelecimento, produtos e preços.

Tabela 16 – Dados da requisição para consulta dos detalhes de uma nota fiscal

URI	**/salesslips/:sale_slip_id**
Método	GET
Necessita do token de autenticação?	Sim
Parâmetros da requisição	user_id

O cliente pode optar, por exemplo, por desassociar uma nota fiscal daquele determinado usuário. Isso pode ocorrer em casos onde o usuário não foi o cliente daquela compra, apenas inseriu a nota fiscal para auxiliar o LudiiPrice a ter mais dados. Aquela nota então desaparece da listagem de notas do usuário, mas não é excluída da base de dados do servidor. Para isto, o cliente precisa enviar uma requisição HTTP PUT da Tabela 17 com o id da nota fiscal desejada.

Tabela 17 – Dados da requisição para excluir associação da nota fiscal com um usuário

URI	**/salesslips/:sale_slip_id/desassociate**
Método	PUT
Necessita do token de autenticação?	Sim
Parâmetros da requisição	sale_slip_id

4.7.6 Gerenciamento da lista de compras

Esta funcionalidade permite que o usuário possa criar uma lista de compras com produtos cadastrados na base de dados do servidor. Com isto, é possível descobrir quais estabelecimentos detém o menor preço para aqueles itens, ajudando o consumidor a descobrir onde vale a pena comprar seus itens.

A criação da lista de compras pode ser feita realizando uma requisição HTTP POST da Tabela 18 passando o atributo *name* como nome de identificação da lista de compras a ser criada.

Tabela 18 – Dados da requisição para criação de uma lista de compras

URI	**/shoppinglists**
Método	POST
Necessita do token de autenticação?	Sim
Parâmetros da requisição	name

Criada a lista, o cliente pode adicionar produtos à lista de compras acionando outra rota através de uma requisição HTTP POST da Tabela 19, enviando um *array* de *ids* de produtos através do campo *products*.

Tabela 19 – Dados da requisição para adicionar produtos em uma lista de compras

URI	**/shoppinglists/:shopping_list_id/products/add**
Método	POST
Necessita do token de autenticação?	Sim
Parâmetros da requisição	products

Da mesma maneira, também é possível remover produtos da lista de compras enviando um array de *ids* de produtos através de uma requisição HTTP DELETE da Tabela 20.

Tabela 20 – Dados da requisição para remover produtos de uma lista de compras

URI	**/shoppinglists/:shopping_list_id/products/remove**
Método	DELETE
Necessita do token de autenticação?	Sim
Parâmetros da requisição	products

A exclusão de lista de compras pode ser feita através de uma requisição HTTP DELETE da Tabela 21 passando o *id* da lista de compras.

Tabela 21 – Dados da requisição para remover uma lista de compras

URI	**/shoppinglists/:shopping_list_id**
Método	DELETE
Necessita do token de autenticação?	Sim

O conjunto de listas de compras relacionadas a um usuário pode ser consultado através de uma requisição HTTP GET a Tabela 22.

Tabela 22 – Dados da requisição para consultar as listas de compras de um usuário

URI	**/shoppinglists**
Método	GET
Necessita do token de autenticação?	Sim

Existe também uma rota para consultar os detalhes da lista de compras (nome e produtos) através de outra requisição HTTP GET da Tabela 23.

Tabela 23 – Dados da requisição para consultar detalhes de uma lista de compras

URI	**/shoppinglists/:shopping_list_id**
Método	GET
Necessita de token de autenticação?	Sim
Parâmetros da requisição	shopping_list_id

4.7.7 Funcionalidades relacionadas a produtos

4.7.7.1 Produtos mais populares

A consulta aos produtos mais populares é feita com base na geolocalização enviada pelo cliente. Com isto, o serviço retorna os produtos mais presentes nas notas fiscais emitidas por estabelecimentos no raio das coordenadas fornecidas. Para acessar esse serviço, o cliente realiza uma requisição HTTP GET da Tabela 24 ao servidor.

Tabela 24 – Dados da requisição para consultar produtos mais comprados

URI	**/products/most-purchased**
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	stores_limit, distance_limit, products_limit, lat, lng

4.7.7.2 Buscar produto

A busca de produtos é outra funcionalidade que também utiliza a geolocalização. Nesta requisição HTTP GET da Tabela 25, o cliente envia o termo de busca e o serviço retorna uma lista de produtos compatíveis com o termo. Para cada item da lista, são exibidos o maior e menor preço (quando existir) dentre os estabelecimentos que se encontram no raio de alcance da localização enviada na requisição.

Tabela 25 – Dados da requisição para buscar produtos

URI	**/products/search**
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	product_name, stores_limit, distance_limit, products_limit, lat, lng

4.7.7.3 Detalhes de um produto

A exibição de detalhes de um produto além de apresentar informações básicas como por exemplo o nome, EAN tributável e EAN comercial, também apresenta os preços daquele produto para estabelecimentos próximos da coordenada fornecida pelo cliente no momento da requisição HTTP POST da Tabela 26 para a rota que fornece este serviço, ordenados a partir do menor preço.

Tabela 26 – Dados da requisição para consultar detalhes de um produto

URI	/products/details/:product_id
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	stores_limit, distance_limit, products_limit, lat, lng
Parâmetros da requisição	product_id

4.7.7.4 Gráfico de variação de preços de um produto

O acompanhamento da variação de preços de um produto é uma funcionalidade importante para comparar o quão distante o preço atual encontra-se de acordo com aqueles praticados anteriormente. Isso pode ser feito através de requisição HTTP GET da Tabela 27.

Tabela 27 – Dados da requisição para consultar gráfico de variação de preços de um produto

URI	/products/details/:product_id
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	stores_limit, distance_limit, products_limit, lat, lng
Parâmetros da requisição	product_id

4.7.8 Histórico de despesas

Refere-se ao acompanhamento de valores totais das compras presentes nas notas fiscais associadas ao usuário. Toda nota que é inserida pelo usuário e permanece associada a ele, tem seu valor computado em função do tempo, que pode ser customizado conforme a necessidade do cliente da requisição HTTP GET da Tabela 29 para obter este serviço. Através do parâmetro *query*, é possível definir um período de tempo mensal ou anual para acompanhamento das despesas usando o parâmetro *timestamp*.

Tabela 28 – Dados da requisição para consultar histórico de despesas do usuário

URI	/histories/expense
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	limit, timestamp

4.7.9 Gráficos de comparação de preços

A API possui duas funcionalidades de comparação dos preços a partir das listas de compras e das notas fiscais digitalizadas pelos usuários.

4.7.9.1 Gráfico de preços de uma lista de compras

Dado que a lista de compras foi criada, é possível realizar uma requisição HTTP GET da Tabela 29 que disponibilizará um objeto JSON contendo um comparativo do preço total dos itens da lista em estabelecimentos próximos da coordenada fornecida pelo cliente da requisição. Esse serviço verifica a disponibilidade de um produto no estabelecimento e caso haja, têm o preço mais recente somado ao total da lista de compras para aquele local. Portanto, podem haver casos onde o preço total da compra ser mais baixo não exatamente significa que todos os produtos foram considerados. Para contornar isto, o objeto JSON também retorna a lista de produtos considerados na hora de computar o valor total da compra.

Tabela 29 – Dados da requisição para consultar gráfico de preços de uma lista de compras

URI	**/shoppinglists/:shopping_list_id/prices-around**
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	stores_limit, distance_limit, products_limit, lat, lng
Parâmetros da requisição	shopping_list_id

4.7.9.2 Gráfico de preços de uma nota fiscal

Através de uma requisição HTTP GET a esta rota, é possível obter um objeto JSON com o total da compra de uma nota fiscal no estabelecimento e na época em que foi emitido. Além disso, também retorna objetos com os preços dos produtos daquela mesma nota em outros estabelecimentos próximos da coordenada fornecida pelo cliente da requisição. Esse serviço verifica a disponibilidade de um produto no estabelecimento e caso haja, têm o preço mais recente somado ao total da lista de compras para aquele local. Caso não haja, é utilizado o preço do produto na época e no estabelecimento que a nota foi emitida.

Tabela 30 – Dados da requisição para consultar gráfico de preços de uma nota fiscal

URI	**/histories/personal/saleslip/:sale_slip_id**
Método	GET
Necessita de token de autenticação?	Sim
Query da requisição	stores_limit, distance_limit, products_limit, lat, lng
Parâmetros da requisição	sale_slip_id

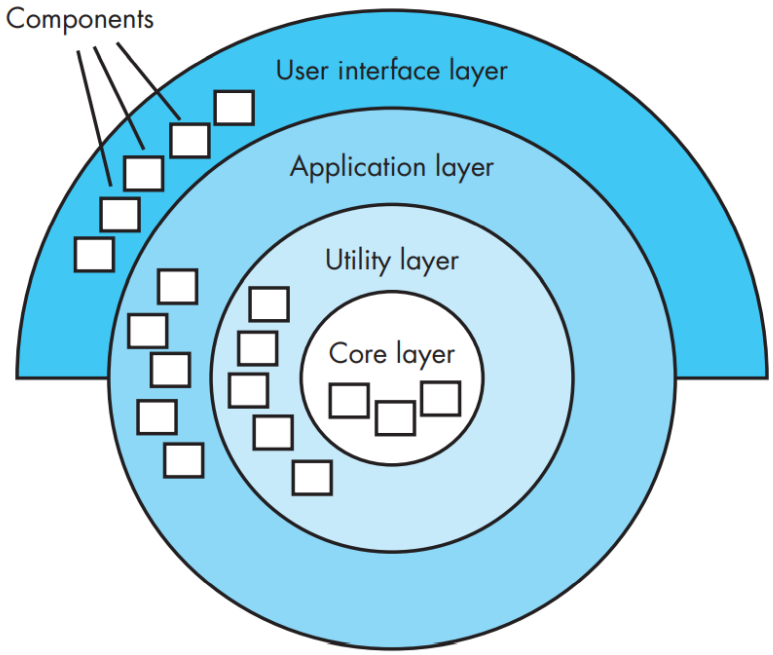
4.8 Estilo arquitetural da API

A construção de sistemas usualmente segue um determinado padrão ou organização em sua estrutura. Conforme Pressman e Maxim (2014), um estilo arquitetônico de software é uma transformação imposta ao projeto de um sistema inteiro na intenção de estabelecer uma estrutura

para todos os componentes do sistema. De acordo com Bass, Clements e Kazman (2012), cada estilo vai englobar determinados elementos como componentes, seus meios de comunicação e suas restrições de integração.

Dentre os diversos estilos arquiteturais, a arquitetura em camadas foi escolhida para estruturar este MVP. Ela divide o software em unidades chamadas camadas, que são um agrupamento de módulos que oferece um conjunto coeso de serviços e uma relação unidirecional de permissão de uso entre as camadas (BASS; CLEMENTS; KAZMAN, 2012). Na camada externa, os componentes atendem às operações da interface do usuário. Na camada interna, os componentes realizam a interface com o sistema operacional. As camadas intermediárias fornecem serviços de utilidade e funções de software de aplicação (PRESSMAN; MAXIM, 2014). A Figura 21 exemplifica a divisão entre as camadas.

Figura 21 – Representação do estilo arquitetural em camadas



Fonte: (PRESSMAN; MAXIM, 2014)

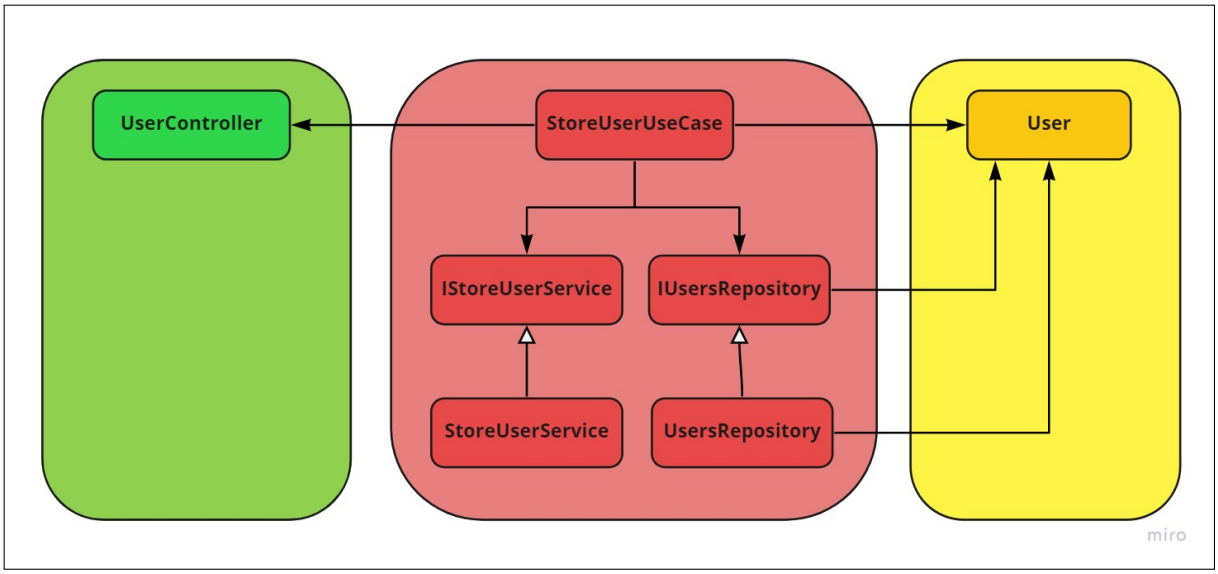
Algumas vantagens inerentes a este tipo de estilo arquitetural justificaram a escolha para o desenvolvimento desta API:

- Possibilidade de expandir o nível de abstração
- Divisão de problemas complexos em etapas iterativas
- Como cada camada está "isolada"e possui seu próprio papel, é possível identificar mais rapidamente a fonte de um problema

- Facilita a manutenção, uma vez que é possível alterar toda uma camada sem comprometer as camadas adjacentes

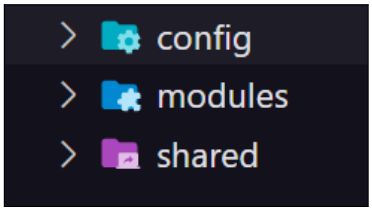
Ao aplicar o conceito da arquitetura em camadas para a API do LudiiPrice, existem três principais camadas, da mais interna à mais externa, respectivamente: domínio, aplicação e infraestrutura, conforme a Figura 22, que representa o fluxo da arquitetura para a entidade Usuário na API. A camada de apresentação é inexistente na API, uma vez que não lida com interfaces, que são de responsabilidade do cliente *mobile*. As pastas do código-fonte foram organizadas neste sentido, de modo a perceber esta configuração a partir de sua estrutura.

Figura 22 – Representação da arquitetura da API



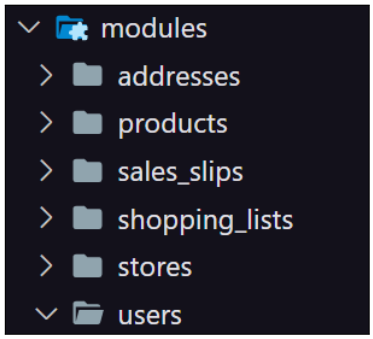
Para cada entidade existente na aplicação, a disposição das pastas no código-fonte engloba todos os arquivos que implementam as suas camadas, todas reunidas na pasta *modules*. A pasta raiz engloba três pastas: *config*, *modules* e *shared*.

Figura 23 – Estrutura principal de pastas do código-fonte da API



Os módulos abrangem as áreas de conhecimento da aplicação, diretamente relacionados com o domínio da aplicação. A implementação das camadas é dada a partir de classes contidas nestes módulos.

Figura 24 – Estrutura das pastas de módulos



Arquivos de uso geral compartilhados com mais de um módulo da aplicação, como por exemplo, o arquivo do servidor, o arquivo principal de rotas, conexão com banco de dados, entre outros encontram-se na pasta *shared*.

Já os arquivos de configurações de bibliotecas externas como autenticação ou serviço de email e classes de erros ficam na pasta *config*.

4.8.0.1 Camada de entidades

Considerado o núcleo da aplicação, a camada de domínio contém lógica e dados essenciais para negócios na forma de entidades. É representada por um arquivo de entidade.

4.8.0.2 Camada de aplicação

Os serviços representam a implementação das funcionalidades presentes no diagrama de casos de uso. Cada caso é representado por um arquivo de serviço, responsável por executar a lógica do caso de uso e retornar as informações para a camada mais externa, de infra, contendo o *controller*.

4.8.0.3 Camada de infraestrutura

A camada de infraestrutura proposta nesta API engloba tanto a camada de *Adapters* quando de *Infraestructure*. Dessa forma, representa a comunicação com elementos externos, como banco de dados e *controllers*, que representam a comunicação com os serviços.

O *controller* é responsável por receber as requisições HTTP, tratar as entradas de dados com validações e repassar os dados para a camada de serviço.

Já a conexão com o banco de dados é representada através dos repositórios, compostos por classes que controlam o acesso aos dados, gerenciando a persistência destas informações. Essa camada não lida apenas com os objetos armazenados nos bancos de dados mas também com dados advindos de APIs externas, como no caso deste MVP, o serviço de consulta às coordenadas dos estabelecimentos comerciais emissores das NFC-e.

Os repositórios contém a implementação de suas interfaces, que desempenham um papel importante, pois definem contratos com métodos que devem ser respeitados durante a implementação. Dessa forma, é possível, por exemplo, alterar o banco de dados da aplicação mas garantindo que a nova implementação mantenha métodos com os mesmos parâmetros e retornos.

4.9 *Deploy* da API

Visando o acesso da interface mobile a esta API, foi necessário colocar o sistema em um ambiente de produção. Para isto, foi escolhido o Heroku[1], plataforma na nuvem com suporte a várias linguagens de programação, permitindo construir, disponibilizar e monitorar aplicações.

O Heroku é uma solução *Platform as a Service* (PaaS), modelo que fornece ao contratante uma plataforma em nuvem com todas as ferramentas, serviços e *templates* para o desenvolvimento, gerenciamento e manutenção de aplicações sem que haja a preocupação com os custos ou a administração de uma infraestrutura deste porte (JUNIOR et al., 2020). O provedor de um PaaS fornece, gerencia e suporta pacotes de software, servidores, rede, sistemas operacionais, banco de dados e ferramentas de desenvolvimento, por exemplo.

A PaaS melhora a produtividade, minimiza o custo do software e permite que as empresas disponibilizem seus produtos mais rapidamente hospedando todo o ambiente de desenvolvimento de aplicativos online (YASRAB, 2018). Além disso, este modelo favorece a escalabilidade, o fornecimento do serviço pode variar de acordo com a demanda ou picos de sazonalidade.

O Heroku executa e dimensiona as aplicações em contêineres isolados e virtualizados chamados de *dynos*, projetados para executar código com base em um comando especificado pelo usuário. Os aplicativos executados no Heroku geralmente têm um domínio exclusivo usado para enviar solicitações HTTP para o *dyno* correto. Quando são criadas, as aplicações são associadas a repositórios remotos, geralmente acessados através do *Git*. O código fonte, uma descrição das dependências do projeto e um arquivo *Procfile* para especificar os comandos que são executados pelo aplicativo na inicialização são necessários neste processo (SOLóRZANO; CHARãO, 2017). Entre as vantagens do Heroku, estão:

- *Deploy* automatizado através do Git
- Praticidade e rapidez, sendo possível disponibilizar uma aplicação em poucos minutos
- Grande ecossistema de ferramentas
- Oferece um poderoso *dashboard* e *Command-Line Interface* (CLI)
- Amigável para iniciantes

[1] <https://www.heroku.com/>

A plataforma fornece uma série de planos gratuitos e pagos, que dependem da necessidade do contratante. Em razão desta versão da API do LudiiPrice se tratar de um MVP e ausência de recursos financeiros que apoiem a iniciativa neste momento, o Heroku foi escolhido pelos benefícios de sua versão gratuita serem suficientes para cobrir as necessidades de disponibilização da API nestas condições. Dentre as limitações, o plano gratuito oferece apenas um *dyno* e após trinta minutos de inatividade, o sistema entra em repouso, necessitando de alguns minutos para entrar em atividade assim que a aplicação é contatada após esse período.

Como esta API utiliza armazenamento através do *PostgreSQL*, o Heroku oferece um *add-on* gratuito chamado Heroku Postgres. Ele é responsável pela criação e disponibilização do banco de dados, oferecendo as credenciais para se conectar com a aplicação, através de variáveis de sistema. Na versão gratuita, são permitidas apenas 20 conexões simultâneas, 10000 linhas de dados nas tabelas e armazenamento máximo de 1GB.

Para disponibilizar o MVP através deste serviço, o código-fonte da aplicação foi versionado através do *Github*. É necessário também se registrar através do site do Heroku. Para conectar com a plataforma, foi instalado o CLI, que permite criar e gerenciar as aplicações diretamente do terminal. A partir dele, foi criado um contêiner para a aplicação. Para realizar o *deploy* da API, bastou executar um comando *git push*, que envia o conteúdo do repositório local para um repositório remoto. A partir daí, o Heroku começará a baixar as dependências necessárias para a execução do projeto e caso o processo ocorra com sucesso, a plataforma disponibiliza um link para acesso da aplicação.

4.10 Validação dos *endpoints* da API

A validação permite testar o comportamento dos serviços da API através da inserção de dados em uma requisição, obtendo informações e um código de status como resposta. Testar a requisição é o processo de analisar se o objeto JSON enviado atende ao que foi ajustado, da mesma forma que também verifica se as informações e o código retornados correspondem ao esperado. Os testes deste MVP foram realizados utilizando o *software Insomnia*, aplicação que permite realizar requisições HTTP através de uma *interface*.

Durante o desenvolvimento de um endpoint da API, as requisições eram feitas utilizando o programa em questão. Quando um erro ou inconsistência era detectada (status ou mensagens de erro erradas, parâmetros errados no corpo da requisição, corpo da resposta com dados incorretos, por exemplo), a funcionalidade era revisada e o problema era corrigido. Os resultados finais obtidos nesta validação constam no Anexo A deste documento.

5
Considerações finais

Monitorar preços é fundamental na hora de fazer as compras de produtos do dia-a-dia nos supermercados e faz a diferença na economia das famílias, em especial aquelas menos abastardas. Ferramentas tecnológicas como aplicativos possuem um enorme potencial na hora de ajudar os consumidores a garantirem o melhor preço em estabelecimentos comerciais. Nesse sentido, este trabalho faz parte de um projeto denominado LudiiPrice, visando comparar preços através de notas emitidas por estabelecimentos do estado de Sergipe e digitalizadas colaborativamente pelos usuários. O projeto foi dividido em três frentes, sendo delegado a este trabalho a criação de um MVP de API Rest responsável por fornecer funcionalidades que serão consumidas por um cliente mobile que foi desenvolvido por outro aluno de TCC do Departamento de Computação da UFS.

Para isto, procurou-se fazer o levantamento de diversas NFC-e, de modo a identificar a estrutura e as informações dispostas nestas notas. Um levantamento de aplicativos semelhantes existentes no mercado auxiliou a entender como os possíveis concorrentes lidaram com a necessidade de seus clientes. Em seguida, foi aplicado um questionário online sobre hábitos de compras da população, na tentativa de identificar o perfil do público-alvo do LudiiPrice. Com esta etapa, foi possível definir artefatos de engenharia de software (histórias dos usuários, casos de uso, diagramas de sequência e entidade-relacionamento, por exemplo) visando modelar o domínio do problema.

O desenvolvimento da API foi feito a partir do framework Node.Js, utilizando Typescript, um superconjunto da linguagem Javascript. Para a base de dados, optou-se por utilizar o banco relacional PostgreSQL. As funcionalidades desenvolvidas neste MVP envolveram a digitalização das notas fiscais, comparação de preço de produtos entre estabelecimentos comerciais, busca de itens, acompanhamento da evolução de preços de produtos e montagem de lista de compras. De modo a oferecer uma experiência customizada para o usuário, algumas funcionalidades da API consideraram a geolocalização do usuário em comparação as coordenadas dos estabelecimentos cadastrados. O cálculo da distância foi feita a partir da Fórmula de Haversine. A comunicação da

API com o cliente é feita por requisições HTTP em rotas específicas, que retornam objetos JSON.

O MVP produzido foi validado a partir da utilização do software Insomnia, responsável por enviar requisições HTTP à API. Dessa forma, é possível verificar se os dados da requisição e da resposta (conteúdo e status) estão de acordo com o esperado. A API foi disponibilizada na nuvem através da plataforma PaaS Heroku, que oferta um plano gratuito suficiente para as necessidades deste MVP. Através de um link disponibilizado pela plataforma, o cliente *mobile* do LudiiPrice pode acessar o serviço.

5.0.1 Trabalhos futuros

A expansão das funcionalidades deste MVP permitirá aprimorar a qualidade do serviço. Entre as sugestões estão:

- Expandir a disponibilidade da API para digitalização da notas fiscais emitidas por outras unidades federativas do Brasil além de Sergipe.
- Desenvolver um sistema de incentivo à colaboração dos usuários na digitalização das notas fiscais através de pontuação.
- Aplicar técnica de Processamento de Linguagem Natural em produtos que não possuem código EAN para que possam ter seus nomes identificados por extenso.
- Adicionar suporte da API ao cliente *mobile* para funcionalidade de geolocalização por mapas, permitindo mostrar localização dos estabelecimentos ou traçar rotas de navegação àqueles de escolha do usuário.
- Expandir disponibilidade da API para comparação de preços de combustíveis em postos de gasolina.

Referências

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2012. ISBN 0321815734. Citado na página 60.

BERRIOS, L.; SANTOS, J. Impactos da inflaÇÃo no poder de compra do salÁrio mÍnimo: Um breve panorama. *Revista de Administração do Unisal*, v. 6, n. 9, 2016. Disponível em: <http://www.revista.unisal.br/sj/index.php/RevAdministracao/article/view/470>. Citado na página 11.

BROUCKE, S. V.; BAESENS, B. *Practical Web Scraping for Data Science: Best Practices and Examples with Python*. 1st. ed. USA: Apress, 2018. ISBN 1484235819. Citado na página 24.

BRUMMELEN, G. V. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press, 2012. ISBN 9781400844807. Disponível em: <https://books.google.com.br/books?id=mYz7QIt3vQoC>. Citado na página 29.

ELMASRI, R.; NAVATHE, S.; PINHEIRO, M. *Sistemas de banco de dados*. Pearson Addison Wesley, 2009. ISBN 9788588639171. Disponível em: <https://books.google.com.br/books?id=tylQGgAACAAJ>. Citado na página 26.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, 2000. Citado na página 21.

FOWLER, M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3. ed. USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321193687. Citado 4 vezes nas páginas 24, 25, 27 e 28.

GRANDA, A. *Preços determinam decisão de compra dos consumidores, indica pesquisa | Agência Brasil*. 2015. (Accessed on 07/03/2021). Disponível em: <https://agenciabrasil.ebc.com.br/economia/noticia/2015-03/precos-determinam-decisoes-de-compra-dos-consumidores-brasileiros-indica>. Citado na página 11.

GREGORY, M. *Introdução à Economia: princípios de micro e macroeconomia*. Rio de Janeiro: Campus, 2001. Citado na página 11.

GS1 AISBL. *Qual a diferença entre cEAN e cEANTrib?* 2021. (Accessed on 07/02/2021). Disponível em: <https://www.gs1br.org/faq/qual-a-diferen%C3%A7a-entre-cean-e-ceantrib>. Citado na página 21.

IBM. *Criando Diagramas de Sequência - Documentação da IBM*. 2021. <https://www.ibm.com/docs/pt-br/rsar/9.5?topic=diagrams-creating-sequence>. (Accessed on 02/04/2022). Citado na página 27.

IBM. *Modelos e Diagramas UML - Documentação da IBM*. 2021. <https://www.ibm.com/docs/pt-br/rsar/9.5?topic=diagrams-uml-models>. (Accessed on 02/04/2022). Citado na página 24.

JUNIOR, J. M. et al. Promovendo inovação com a atualização de serviços de platform as a service. *Brazilian Journal of Development*, v. 6, p. 18143–18154, 01 2020. Citado na página 63.

KUROSE, K. W. R. J. F. *Redes de computadores e a Internet: uma abordagem top-down*. São Paulo: Pearson Education do Brasil, 2013. ISBN 9788543014432. Citado na página 21.

LEFFINGWELL, D. A. Agile software requirements: Lean requirements practices for teams, programs, and the enterprise. In: . [S.l.: s.n.], 2011. Citado na página 42.

LIMA, L. P. d. M. Jociane Franzoni de. *ANÁLISE DE DESEMPENHO SOBRE CAMADAS DE PERSISTÊNCIA EM BANCO DE DADOS RELACIONAL E NÃO-RELACIONAL*. 2017. Monografia (Bacharel em Computação), UTFPR (Universidade da Região da Campanha), Curitiba, Brasil. Citado na página 27.

LUDIICO. *LudiiPrice - Engenharia de software*. 2022. <https://sites.google.com/view/ludiiprice/engenharia-de-software?authuser=0>. (Accessed on 23/02/2022). Citado na página 46.

LUDIICO. *LudiiPrice - Questionário*. 2022. <https://sites.google.com/view/ludiiprice/questionário>. (Accessed on 23/02/2022). Citado 8 vezes nas páginas 35, 36, 37, 38, 39, 40, 41 e 42.

MARS, T. *JSON at Work*. [S.l.]: O'Reilly Media, Inc., 2017. ISBN 9781449358327. Citado na página 22.

MARTINS, R. F. *Interface mobile para a recomendação de produtos de supermercados baseado nas melhores ofertas próximas do usuário*. Avenida Marechal Rondon Jardim s/n - Rosa Elze, São Cristóvão - SE, 49100-000, 2022. 100 p. Citado na página 13.

ORIONTEC. *EAN tributável: entenda sua importância para o cadastro tributário*. 2019. Disponível em: <https://blog.oriontec.com.br/ean-tributavel/>. Citado na página 20.

PEZOA, F. et al. Foundations of json schema. In: . Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016. (WWW '16), p. 263–273. ISBN 9781450341431. Disponível em: <https://doi.org/10.1145/2872427.2883029>. Citado na página 22.

PRASS, R. *Nota Fiscal eletrônica (NF-e): tudo que você precisa saber*. 2011. Disponível em: <http://g1.globo.com/tecnologia/noticia/2011/05/entenda-o-que-sao-os-qr-codes-codigos-lidos-pelos-celulares.html>. Citado na página 18.

PRESSMAN, R.; MAXIM, D. B. R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 2014. ISBN 9780078022128. Disponível em: <https://books.google.com.br/books?id=i8NmnAEACAAJ>. Citado 4 vezes nas páginas 24, 28, 59 e 60.

RICHARDSON, L.; AMUNDSEN, M.; RUBY, S. *RESTful Web APIs*. [S.l.]: O'Reilly Media, Inc., 2013. ISBN 1449358063. Citado na página 22.

SECRETARIA DA FAZENDA DO ESTADO DE SÃO PAULO. *Perguntas Frequentes*. 2021. (Accessed on 07/02/2021). Disponível em: <https://portal.fazenda.sp.gov.br/servicos/nfe/Paginas/perguntas-frequentes.aspx/respostas_X.asp>. Citado na página 20.

SECRETARIA DA RECEITA FEDERAL. *Portal da Nota Fiscal Eletrônica*. 2021. Disponível em: <https://www.nfe.fazenda.gov.br/portal/>. Citado 2 vezes nas páginas 16 e 17.

SILBERSCHATZ, A.; SUNDARSHAN, S.; KORTH, H. *Sistema de Banco de Dados*. Elsevier, 2016. ISBN 9788535251425. Disponível em: <https://books.google.com.br/books?id=1FBaDwAAQBAJ>. Citado 2 vezes nas páginas 25 e 26.

SOLóRZANO, A. L. V.; CHARãO, A. S. Explorando a plataforma de computação em nuvem heroku para execução de programas paralelos com openmp. In: *Anais da XVII Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*. Porto Alegre, RS, Brasil: SBC, 2017. ISSN 2595-4164. Disponível em: <https://sol.sbc.org.br/index.php/eradrs/article/view/2962>. Citado na página 63.

SYED, B. A. *Beginning Node.Js*. 1st. ed. USA: Apress, 2014. ISBN 1484201884. Citado na página 23.

TECNOSPEED. *Nota Fiscal eletrônica (NF-e): tudo que você precisa saber*. 2021. Disponível em: <https://blog.tecnospeed.com.br/nota-fiscal-eletronica-nf-e-tudo-que-voce-precisa-saber/>. Citado na página 17.

YASRAB, R. *PaaS Cloud: The Business Perspective*. 2018. Citado na página 63.

Apêndices

APÊNDICE A – Capturas de tela das validações dos *endpoints* da API através do *Insomnia*

Figura 25 – Criação de usuário

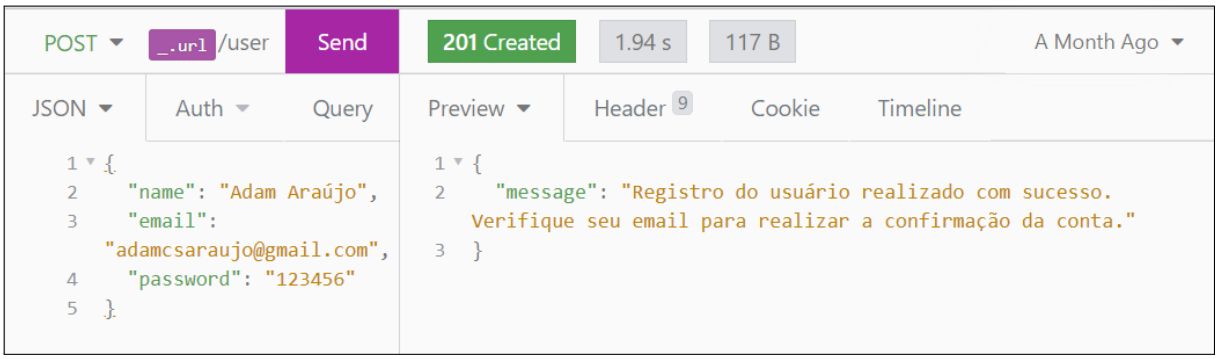


Figura 26 – Autenticação

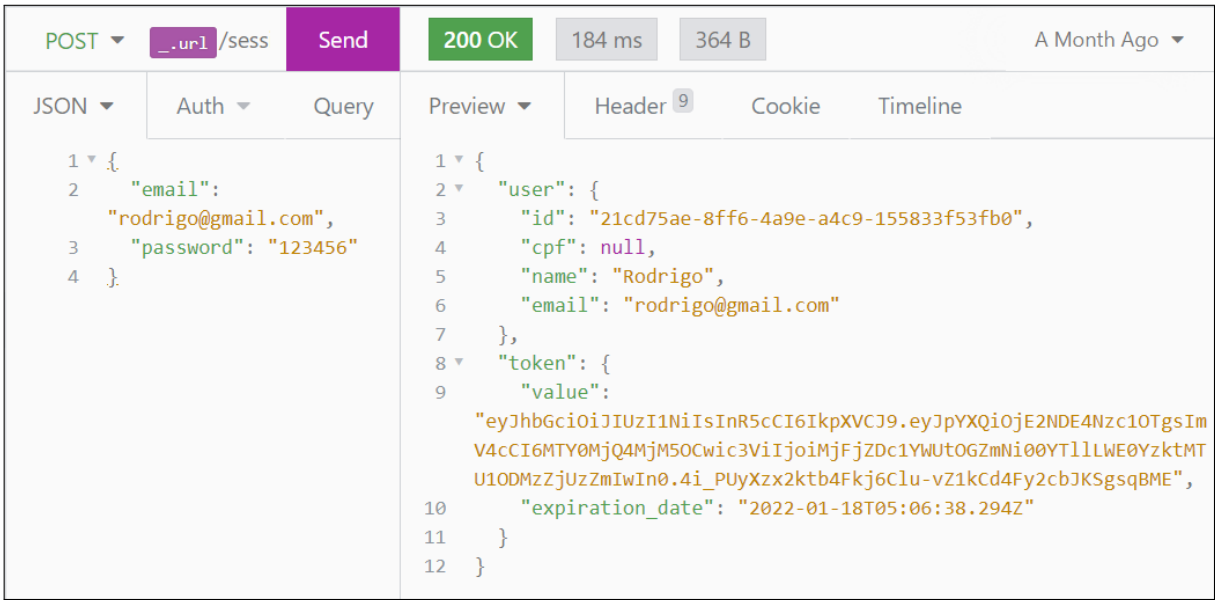


Figura 27 – Adicionar nota fiscal

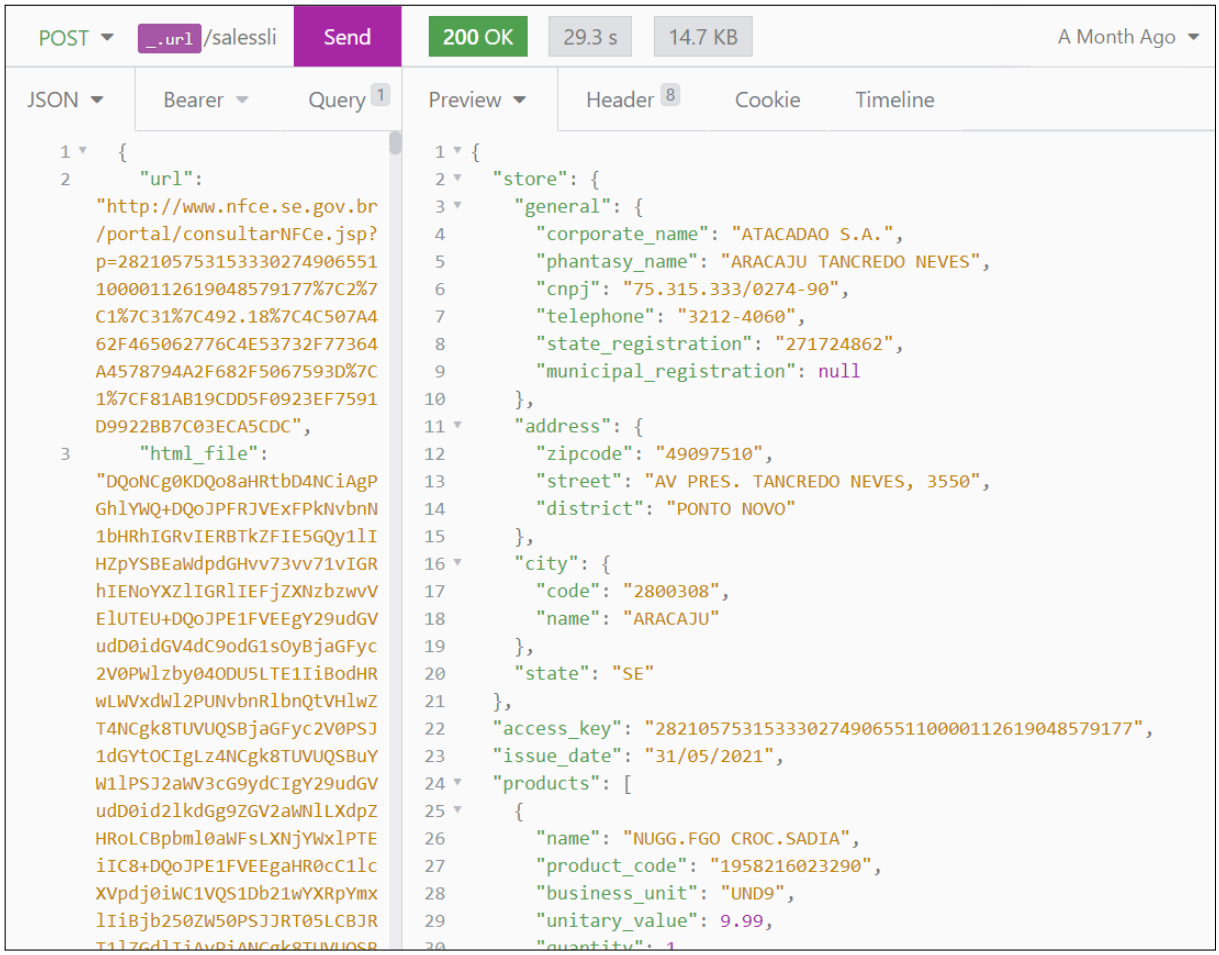


Figura 28 – Notas fiscais de um usuário

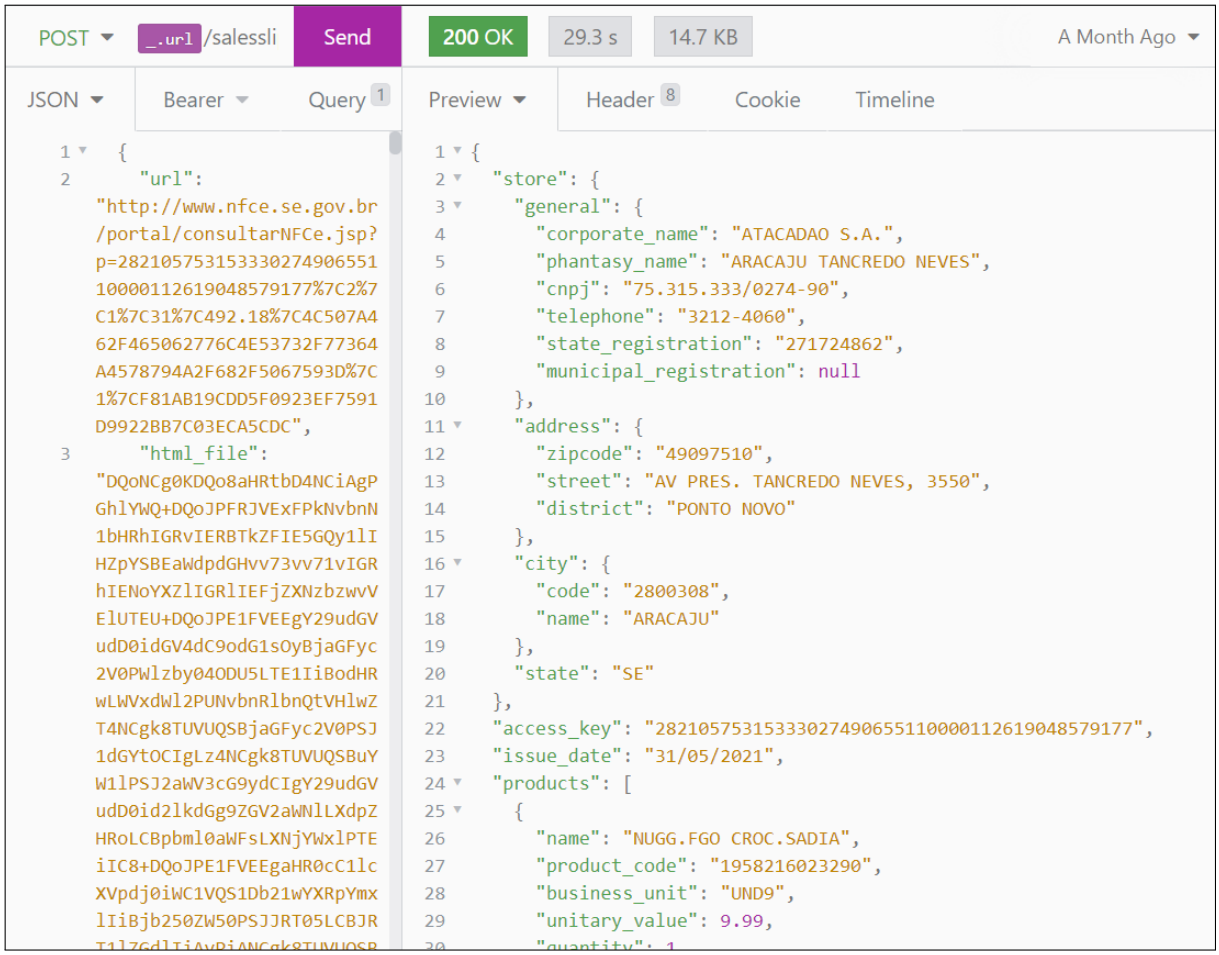


Figura 29 – Detalhes de uma nota fiscal

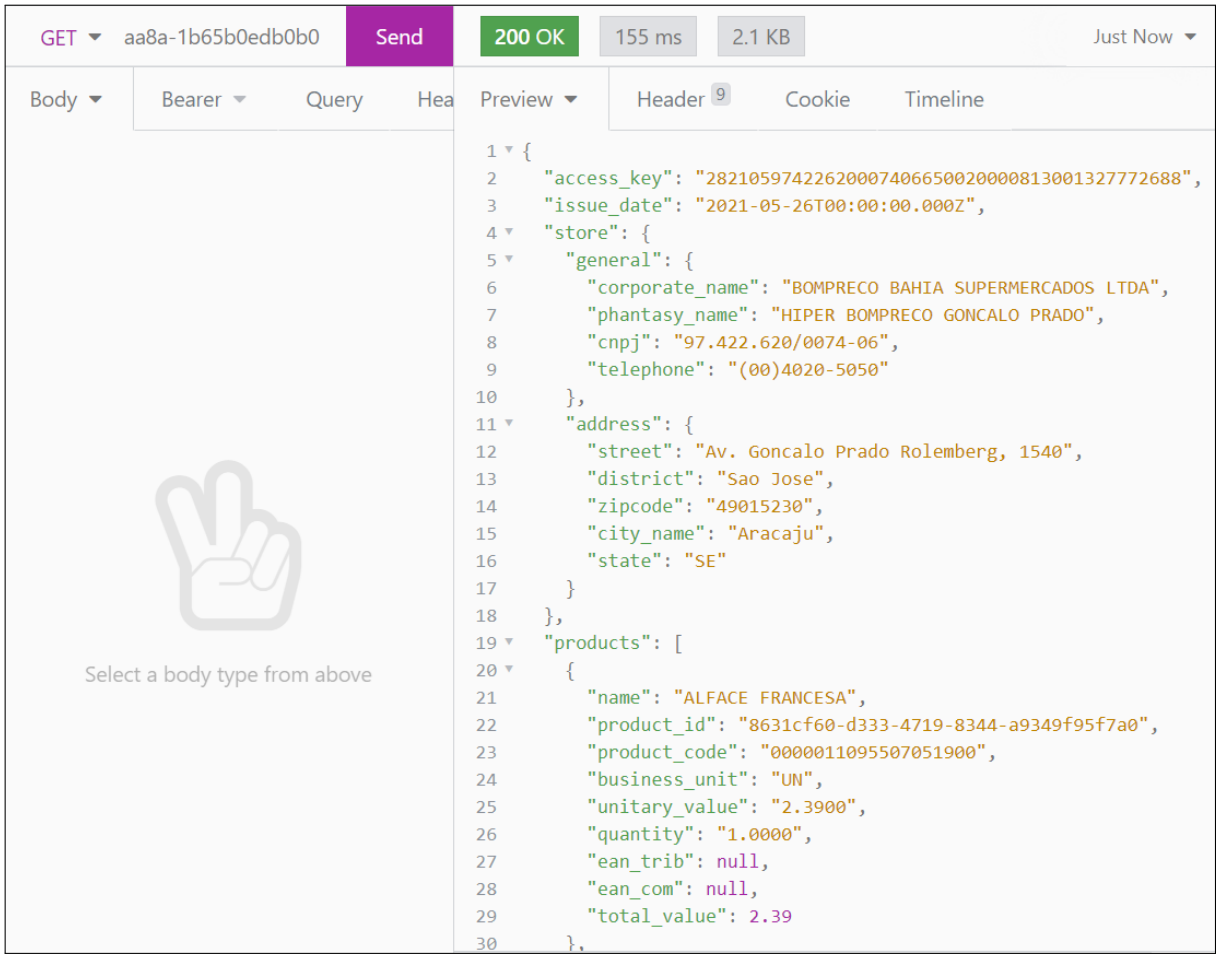


Figura 30 – Busca por produtos

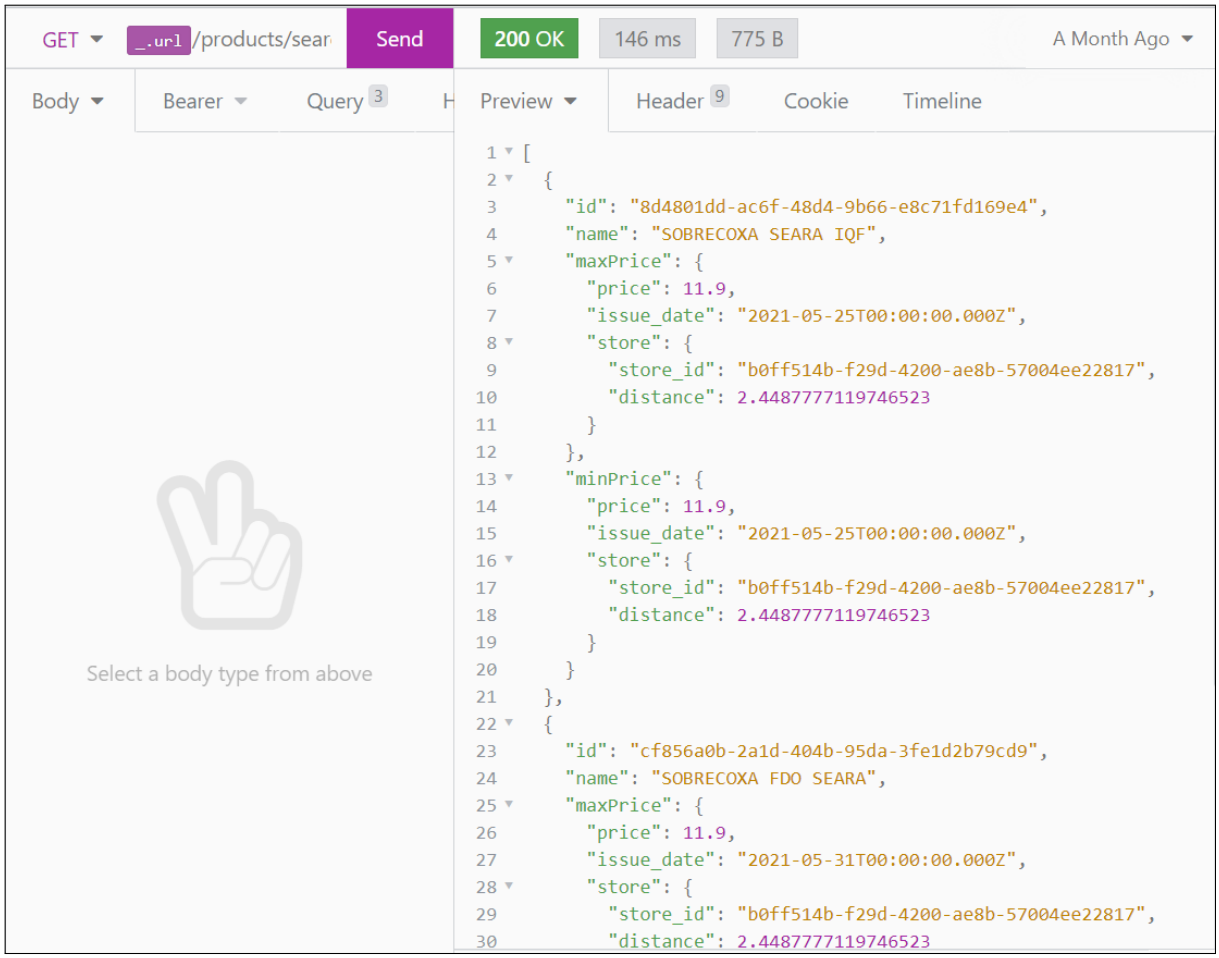


Figura 31 – Detalhes dos produtos

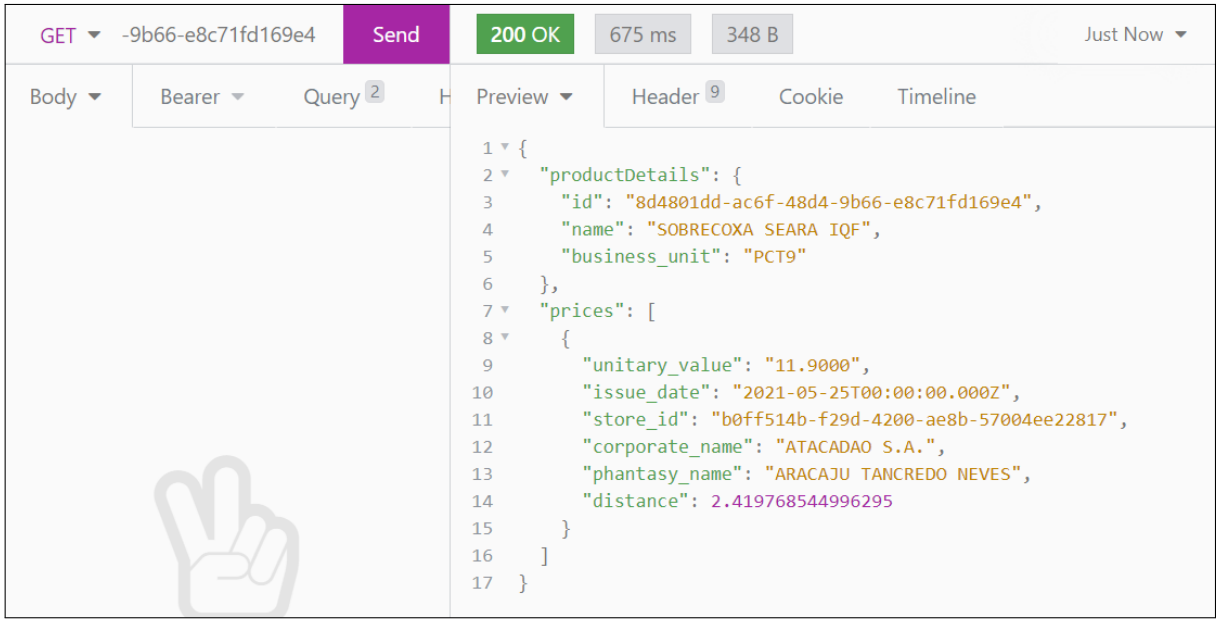


Figura 32 – Produtos mais comprados

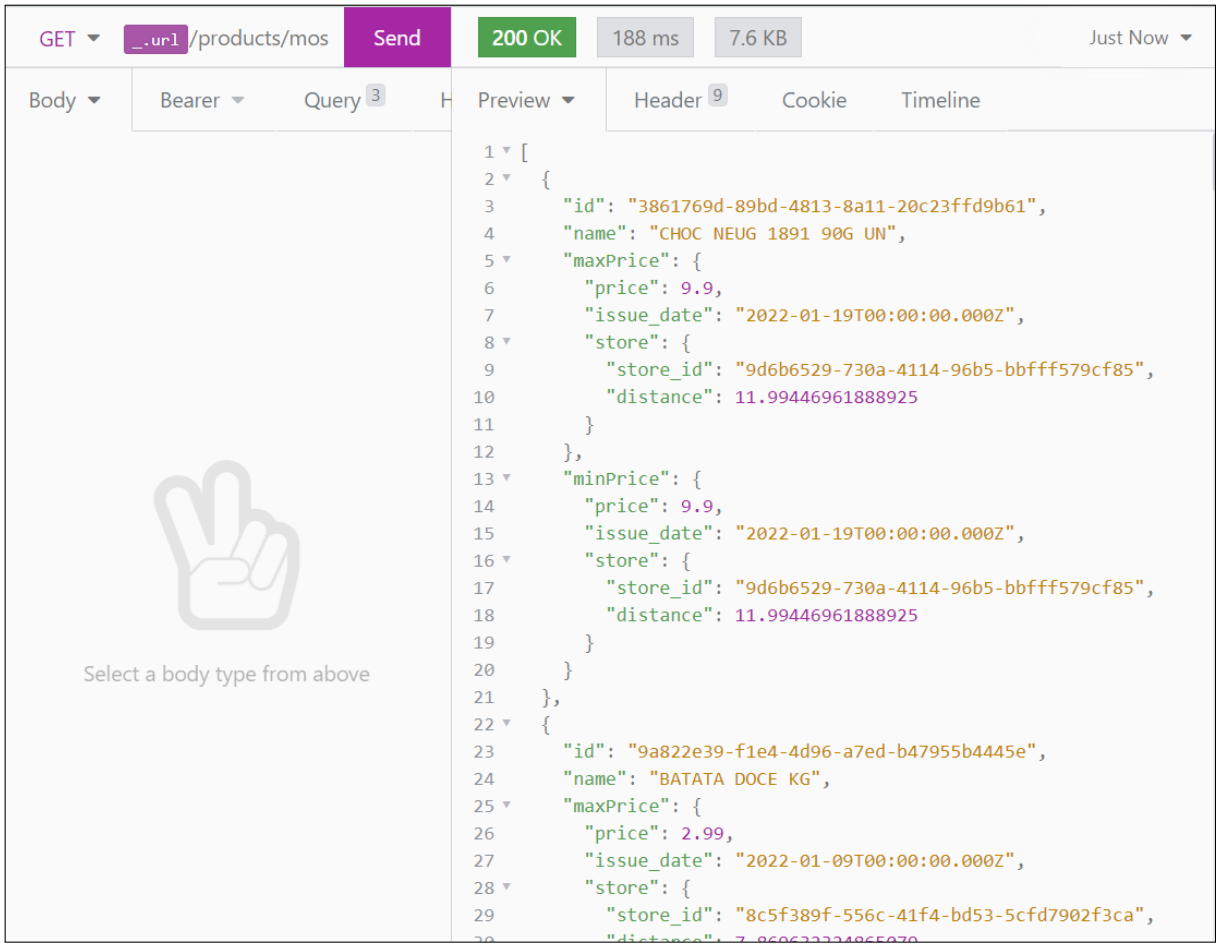


Figura 33 – Criar lista de compras

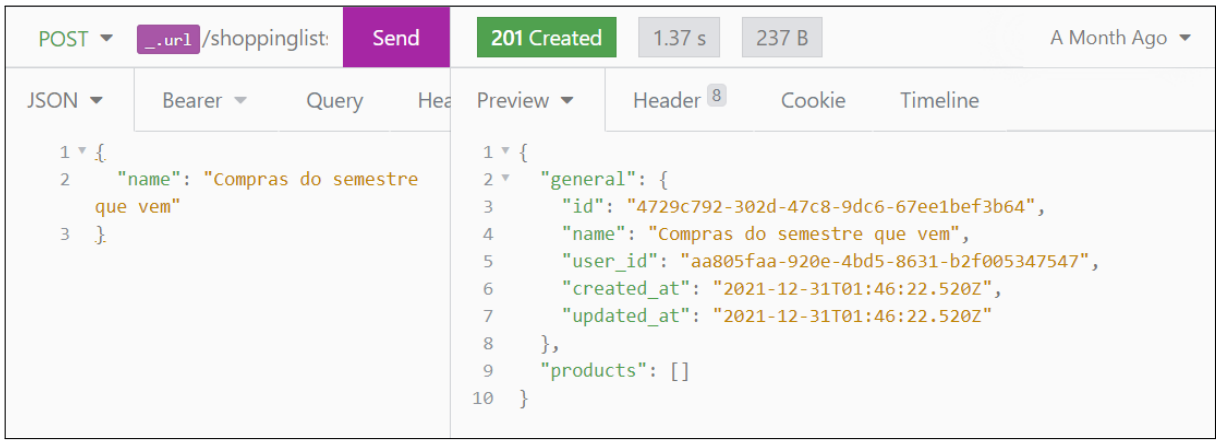


Figura 34 – Lista de compras de um usuário

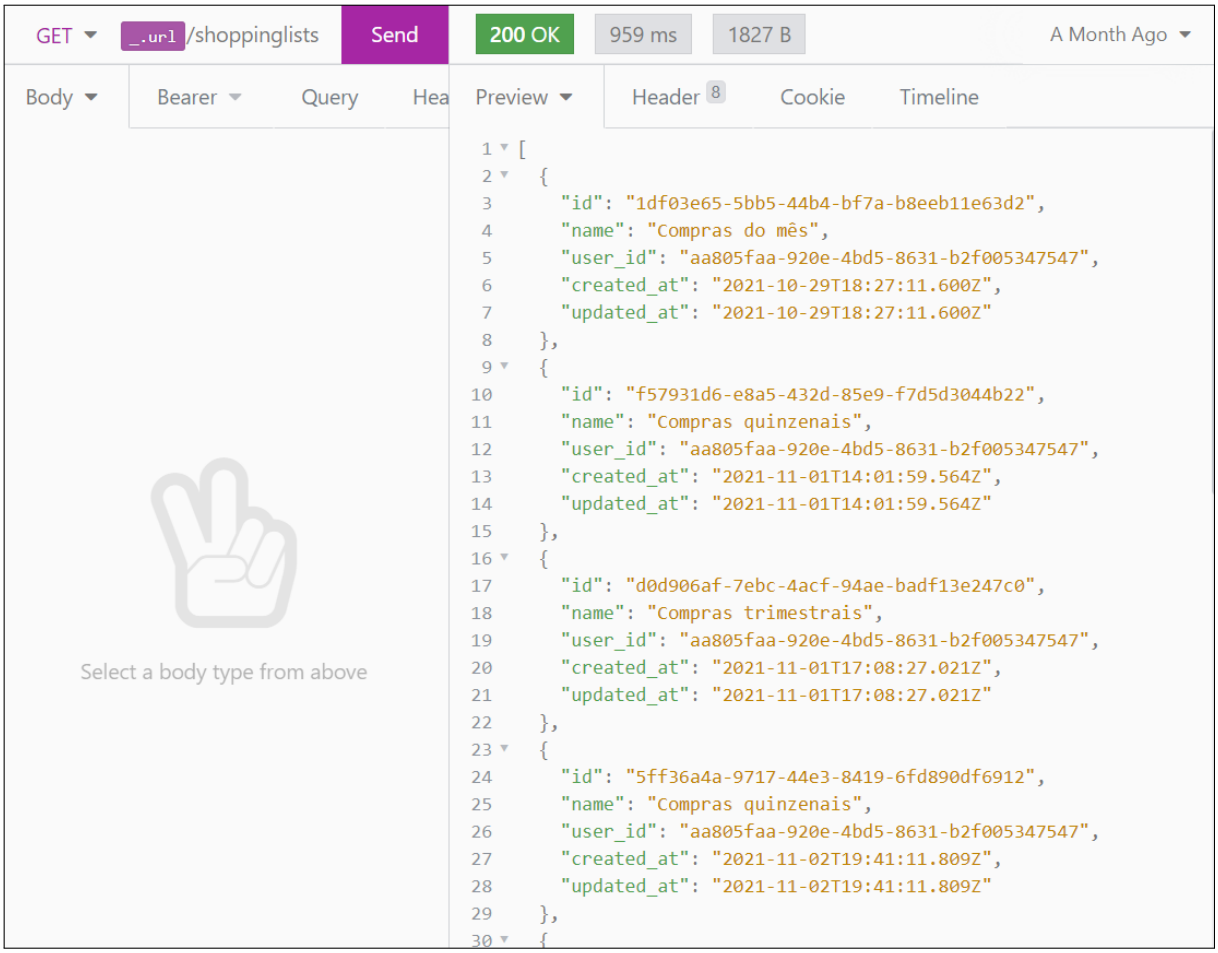


Figura 35 – Detalhes de uma lista de compras

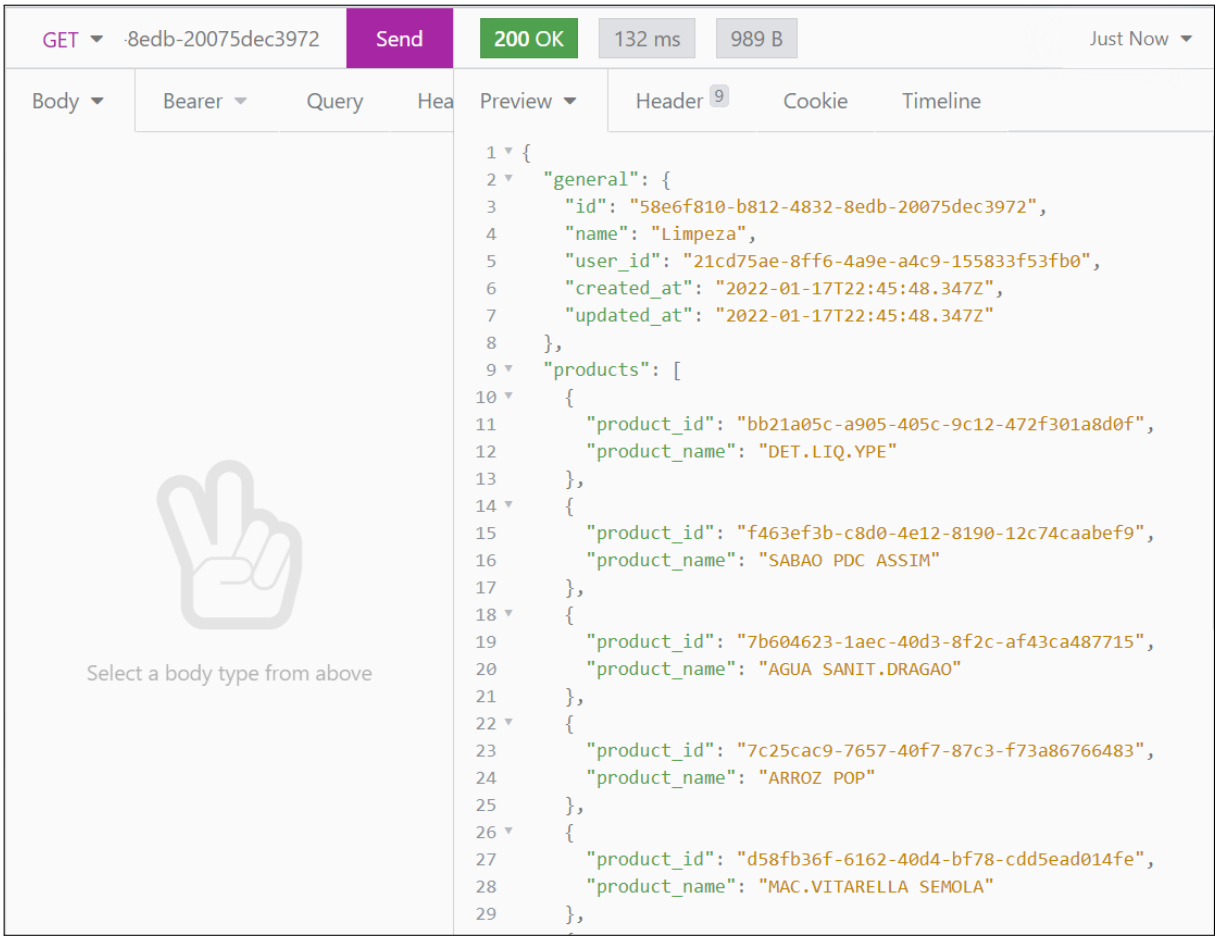


Figura 36 – Gráfico de comparação de preços para produtos de uma lista de compras

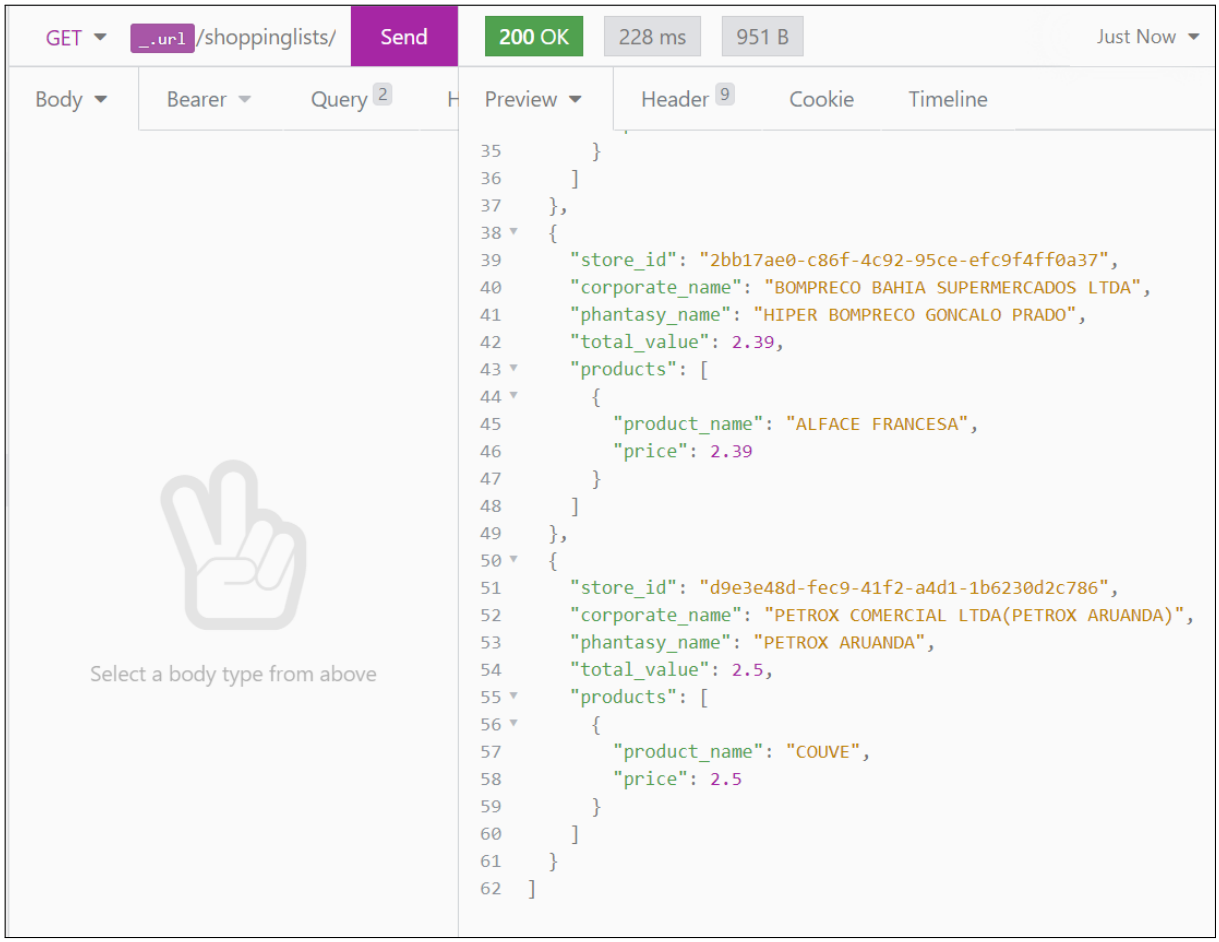


Figura 37 – Gráfico de preços de uma nota fiscal

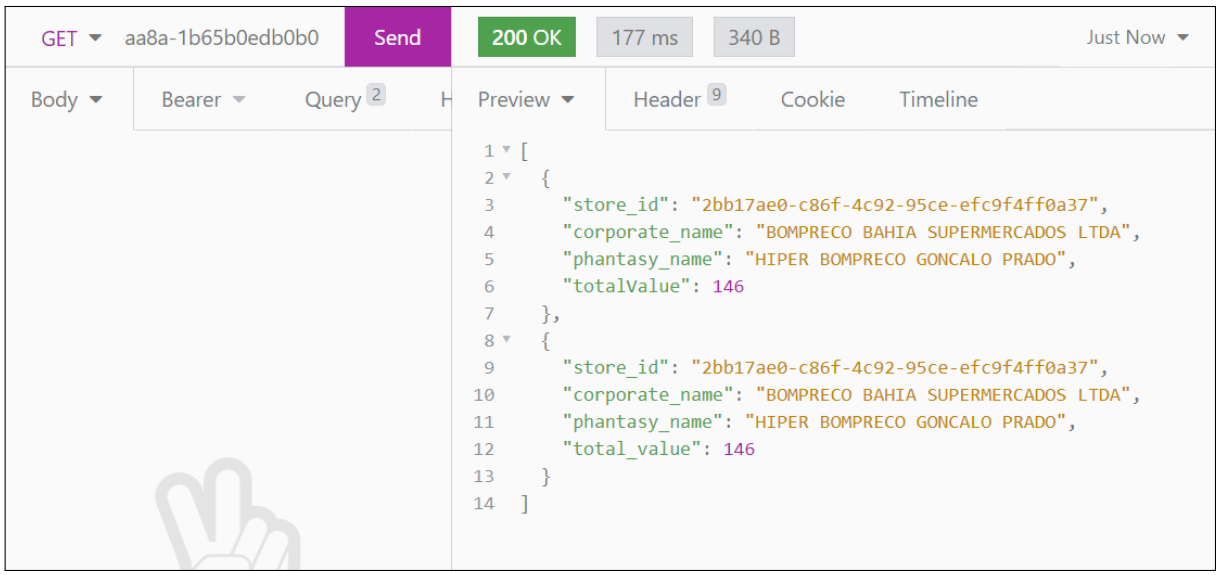


Figura 38 – Gráfico de variação de preços de um produto

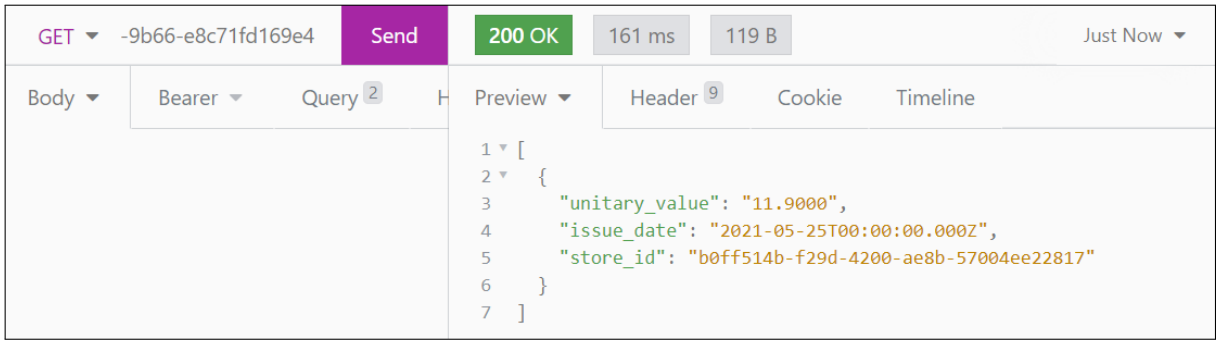
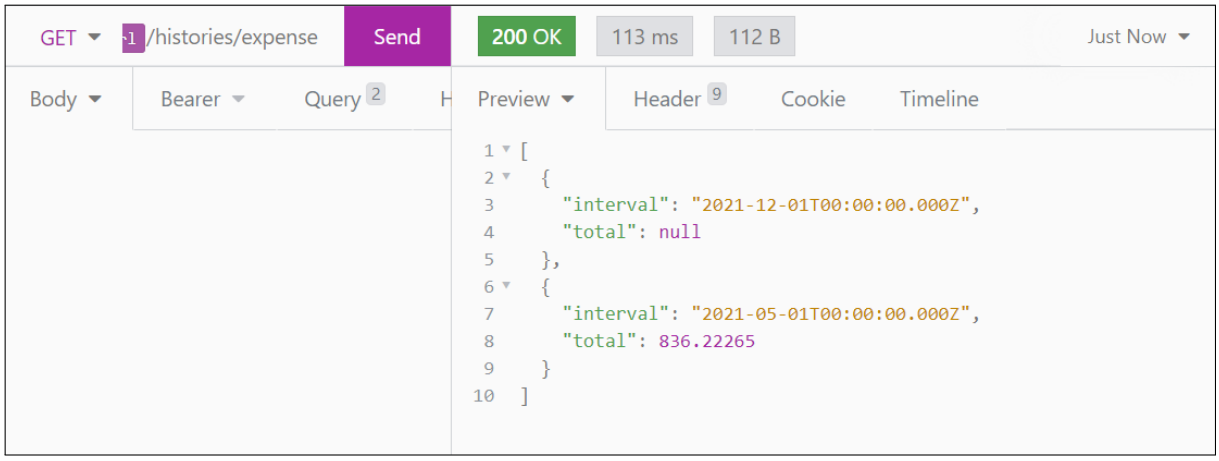


Figura 39 – Histórico de despesas do usuário



APÊNDICE B – Manual de deploy no Heroku

Os pré requisitos para o deploy da aplicação no Heroku são:

- Possuir o Git instalado
- Possuir o Heroku CLI instalado

Antes de realizar o deploy, execute o comando git init em seu repositório local, seguido de um commit do seu código para o seu repositório remoto no Github.

Adicione um controle remoto ao seu repositório local com o comando heroku git:remote. Tudo que você precisa é o nome do seu aplicativo Heroku (ludiiprice): heroku git:remote -a ludiiprice

Para implantar seu aplicativo no Heroku, use o comando git push para enviar o código da branch principal do repositório local para o controle remoto do heroku. Por exemplo: git push heroku main

Caso o seu código não esteja na branch main, é possível utilizar outro comando para redirecionar o deploy para a branch correspondente: git push heroku <nome-da-branch>:main