UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS EDUARDO DA SILVA MARTINS

CONSTRUÇÃO DE UM SISTEMA DE GESTÃO DE GASTOS

RIO DE JANEIRO
2023

CARLOS EDUARDO DA SILVA MARTINS

CONSTRUÇÃO DE UM SISTEMA DE GESTÃO DE GASTOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. João Carlos Pereira da Silva

RIO DE JANEIRO

2023

CIP - Catalogação na Publicação

M386c

Martins, Carlos Eduardo da Silva
Construção de um sistema de gestão de gastos / Carlos Eduardo da Silva Martins. -- Rio de Janeiro, 2023.
48 f.

Orientador: João Carlos Pereira da Silva.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2023.

1. aplicativo móvel. 2. web scraping. 3. gestão financeira. 4. automação. I. Silva, João Carlos Pereira da, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

CARLOS EDUARDO DA SILVA MARTINS

CONSTRUÇÃO DE UM SISTEMA DE GESTÃO DE GASTOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 21 de dezembro de 2023

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
JOAO CARLOS PEREIRA DA SILVA
Data: 21/02/2024 10:54:05-0300
Verifique em https://validar.iti.gov.br

João Carlos Pereira da Silva
D. Sc (UFRJ)

gov.br Documento assinado digitalmente
JULIANA BAPTISTA DOS SANTOS FRANCA
Data: 21/02/2024 12:13:28-0300
Verifique em https://validar.iti.gov.br

Juliana Baptista dos Santos França
D. Sc (UFRJ)

gov.br Documento assinado digitalmente
SILVANA ROSSETTO
Data: 21/02/2024 12:09:14-0300
Verifique em https://validar.iti.gov.br

Silvana Rossetto
D. Sc (UFRJ)

AGRADECIMENTOS

Agradeço a todos os meus professores e professoras que me ajudaram a construir as fundações da minha carreira até este ponto, começando em especial pela minha mãe Telma e avó Maria da Conceição, principais pilares para ter chegado ao nível superior e ao final do mesmo. Agradeço também, indiretamente, a todos os desafios aos quais fui exposto ao longo do curso e da minha carreira no mercado que me prepararam para conseguir construir este trabalho. Por fim, ao meu irmão Igor, minha cônjuge Letícia e meus amigos pela compreensão e suporte nos momentos onde a construção deste projeto precisou ser prioridade.

"Software is the language of automation."

Jensen Huang

RESUMO

A tecnologia na palma das mãos tem transformado cada vez mais a nossa sociedade. Alguns anos atrás não conseguíamos fazer compras sem sair de casa e hoje temos essa habilidade em pequenos computadores que cabem no bolso. Essa revolução trouxe novos paradigmas e problemas que direcionam a nossa evolução para novas ferramentas e soluções. Assim como hoje em dia conseguimos gerir nossa conta bancária em nossos celulares, se mostra como necessidade conseguir entender cada despesa que temos no dia a dia. Neste trabalho é proposto um sistema para extração e gerenciamento de gastos onde o usuário terá acesso as suas compras através de um aplicativo de celular e poderá fazer uso de notas fiscais para cadastrar suas despesas. O sistema possui uma série de automações que facilitam a extração das informações da nota fiscal a fim de oferecer uma visualização detalhada dos produtos consumidos.

Palavras-chave: aplicativo móvel; web scraping; gestão financeira; automação.

ABSTRACT

The technology in the palm of our hands has been transforming our society more and more. A few years ago, we couldn't make purchases without leaving home, and today we have this ability in small computers that fit in our pockets. This revolution has brought new paradigms and problems that guide our evolution towards new tools and solutions. Just as we can now manage our bank account on our phones, it becomes necessary to understand every expense we have in our daily lives. This project proposes a system for extracting and managing expenses where the user will have access to them through a smartphone application and can use receipts to register their bills. The system is composed of a series of automations that leverage the extraction of receipts information, aiming to offer a detailed view of the consumed products.

Keywords: mobile application; web scraping; finances management; automation.

SUMÁRIO

1 INTRODUÇÃO

A evolução da tecnologia na palma das mãos tem questionado diversas práticas e revolucionado diversas áreas da vida em sociedade. Muitas são as pessoas acostumadas com a época em que eram utilizados cadernos, agendas ou livros de controle financeiro (SABAB, 2018) ou até mesmo as famosas listinhas de compras anotadas em um papel antes de ir ao supermercado, sem esquecer daquele produto que acabou no meio da semana e alguém teve que correr na venda da esquina para comprar. E o que falar do famoso pagamento de quinto dia útil e vencimentos no dia 10 de todo mês. Práticas e momentos enraizados na sociedade que cada vez mais vêm sido repensados ou otimizados.

Aplicativos de gestão financeira já são uma realidade consolidada no mercado. Há muitos anos já existem à disposição diversas aplicações cujo foco principal é auxiliar no controle de gastos e fontes financeiras de seus usuários. Uma das principais finalidades do famoso pacote Microsoft Office[1] foi trazer a modernidade sobre controle financeiro para empresas nos anos 90. Isso não é diferente na gestão financeira pessoal, que nos últimos anos tem evoluído bastante e recebido a atenção dos principais agentes do mercado.

Em uma pesquisa feita como parte deste projeto pode-se constatar que a grande maioria das aplicações encontradas focam em funcionalidades de gerenciamento financeiro como controle de gastos, gestão de entrada de dinheiro, conexão com conta bancária e gestão de limite de cartões (TECHTUDO, 2021), (BLOGATIVA, 2019) e (VELMURUGAN, 2020). Por conta disso, muitos deles possuem versões grátis com poucas funcionalidades, enquanto que ao se pagar uma taxa de assinatura são disponibilizadas funções mais interessantes (CANALTECH, 2022) (EXAME, 2023). Artigos mais recentes mostram que o cenário ainda se mantém no mercado de aplicativos financeiros (FINANCEONE, 2023) (MOBILIS, 2023). O applicativo mais antigo no mercado, o Guia Bolso (GUIABOLSO, 2012), possui parceria com as principais instituições bancárias com operação no Brasil e por este motivo, suas funcionalidades são concentradas em gerência básica de gastos. Nele é possível se conectar com sua conta bancária e automaticamente ter no aplicativo seus últimos gastos e saldo, além de criar alertas e notificações de entrada e saída de dinheiro. Essas funcionalidades são também encontradas em outras opções de aplicativos como o caso do Mobilis (MOBILLS, 2013), Spendee (SPENDEE, 2018), Money Lover (MONEY..., 2011) entre muitos outros.

Existem também algumas outras alternativas interessantes que podem ser encontradas na literatura. O próprio aplicativo apresentado por (SADAFULE, 2014) mostra apenas funcionalidades simples como notificação de gasto, categorização e visualização dos gastos em um gráfico de pizza. O aplicativo proposto por (VELMURUGAN, 2020) também possui funcionalidades de gastos básicos, com a adição de gastos via mensagens de SMS

[1] https://en.wikipedia.org/wiki/Microsoft$_Office$

recebidas no celular e divisão de despesas entre outros usuários. Já o aplicativo proposto por (SABAB, 2018) além de fornecer um controle de orçamento e despesas, é o único cujo cadastro de gasto se dá através da extração automática do conteúdo de uma nota fiscal impressa através do uso de técnicas de Computação Visual. Esta classe de aprendizado de máquina permite identificar formas, textos e dígitos através do processamento de imagens.

Entre todos os sistemas consultados e analisados, uma funcionalidade não se encontra facilmente: discrimição de gastos. Obviamente que é importante para o usuário saber quanto gastou no final do mês, mas igualmente importante é entender no que o dinheiro foi gasto. Poucas são as aplicações que fornecem uma visualização segregada dos gastos, como por exemplo (SADAFULE, 2014) e (VELMURUGAN, 2020) que o fazem por meio de um gráfico de pizza. Surge a necessidade de se entender qual aspecto da vida é mais caro ou barato, ou quais são os gastos que podem ser otimizados.

Dentre todos os aplicativos do mercado brasileiro avaliados durante a pesquisa apenas um, o Wallet[2], oferece a possibilidade de relacionar um boleto ou nota fiscal a um gasto. Nenhum oferece como possibilidade destrinchar uma compra em diversos itens. Além disso, poucas são as formas automáticas de extração de dados como o exemplo do eExpense de (SABAB, 2018) ou fazer uso de leitura de código de barras ou QR Code para coletar informações (embora alguns façam até pagamento online). Em sua grande maioria a entrada e saída de valores é feita manualmente ou via conexão com a instituição bancária do usuário, o que levanta problemas de segurança.

Neste projeto, o objetivo principal é ajudar o usuário a dar o primeiro passo para entender seus próprios gastos. Para isso foram desenvolvidas formas automáticas de se extrair informações sobre compras através das já conhecidas notas fiscais, emitidas obrigatoriamente em qualquer compra feita no Brasil. Com isso, é possível construir uma base histórica dos gastos do usuário e fazer uma análise de seu comportamento e frequência de compras.

No próximo capítulo, são expostos os principais conceitos utilizados na construção desse sistema, desde os regulamentos governamentais sobre o mercado financeiro até as técnicas de programação utilizadas. No capítulo 3, são expostos os componentes arquiteturais necessários pelo sistema e quais tecnologias foram utilizadas para a implementação, desde a porta de entrada até os agentes construtores da base de dados. No capítulo 4 são expostos os resultados do uso contínuo do sistema, além de ferramentas utilizadas para analisar o comportamento de compras.

[2] https://budgetbakers.com/

2 GERÊNCIANDO GASTOS

Quando se fala de um sistema de gestão de gastos, fica fácil começar a pensar a partir do ponto em que os dados já estão inseridos em alguma base e iniciar a analisar de fato. Entretanto é preciso começar do começo e entender o que são os dados que estão disponíveis para se trabalhar, como obtê-los e como remover o que não serve de nada. Os principais conceitos a serem entendidos são: a definição de uma compra, quais itens a constituem e como identificar os produtos comprados. Para ter uma compra corretamente inserida no sistema é preciso entender e aprender trabalhar com os dados disponíveis para o consumidor. Dessa forma, neste capítulo, serão expostos os principais conceitos necessários para entender e trabalhar com uma compra, além do necessário para fazer a extração das informações relevantes e a inserção na base de dados.

2.1 A NOTA FISCAL ELETRÔNICA

A legislação brasileira institui que toda compra, seja entre pessoas física e jurídica ou entre duas entidades jurídicas, deve ser rastreada por meio de um documento virtual, a nota fiscal, emitida pela parte vendedora, conforme define o Manual NFC-e (GOV-RJ, 2021). Este documento possui uma versão para operações entre empresas e outra para operações que envolvem o público geral. A primeira é chamada de NF-e (Nota Fiscal eletrônica), enquanto a segunda se chama NFC-e, Nota Fiscal de Consumidor eletrônica. As duas versões são muito parecidas, diferindo apenas nas informações sobre as partes contidas no documento. Enquanto a NF-e possui informações mais detalhadas das duas empresas envolvidas na transação, a NFC-e possui apenas informações detalhadas da entidade vendedora, podendo o consumidor ser anônimo.

Para facilitar a construção desse trabalho, o foco estará na NFC-e cujo acesso é mais fácil de automatizar, visto que é possivel acessá-lo eletronicamente sem captchas. Normalmente esse documento é entregue ao consumidor no momento da compra, na forma de um papel como o da Figura 1. Esse papel, entregue pelos estabelecimentos, é nada mais que uma versão mais resumida da NFC-e, mas essa versão possui quase todas as informações necessárias para a análise da compra e dos produtos.

A versão resumida do documento de nota fiscal possui todos os itens da compra com quantidades e preço. Além disso, possui informações do estabelecimento e consumidor. Por fim, o mais importante, o QRCode e a Chave de Acesso para consulta manual[1], caso o mesmo não tenha sido impresso corretamente. Esse QR Code vai ser bastante importante para que sejam extraídas as mesmas informações presentes no papel e sejam salvas na base de dados. (SABAB, 2018) e (VELMURUGAN, 2020) mostram exemplos de soluções

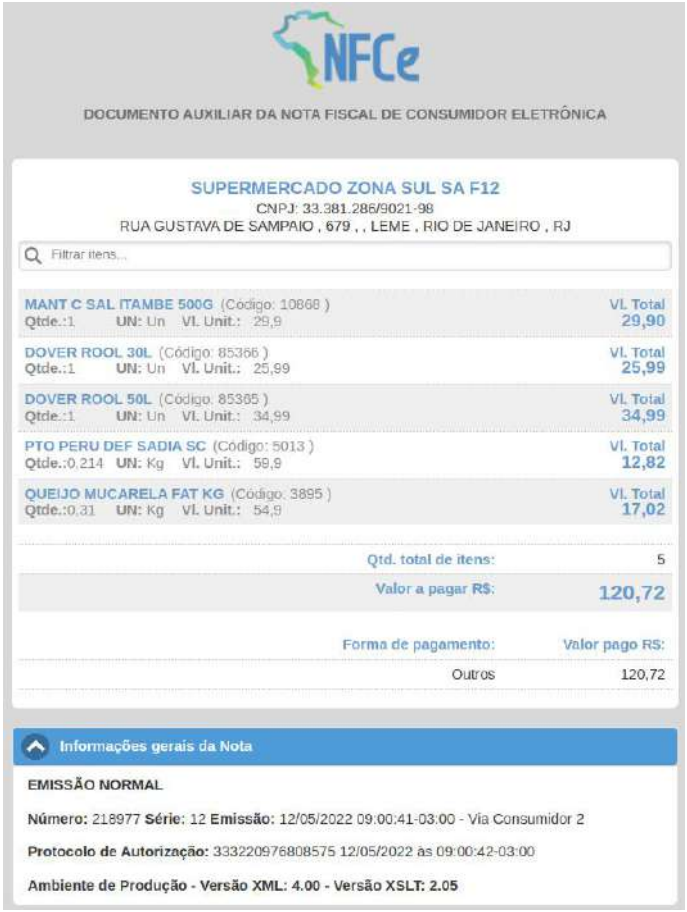
[1] http://www.nfce.fazenda.rj.gov.br/consulta

Figura 1 – Nota Fiscal Consumidor

para casos onde a nota fiscal não possui um QR Code ou código de barras. No caso do primeiro a técnica Optical Character Recognition (especializada em conhecimento de dígitos impressos ou escritos a mão (MORI; NISHIDA; YAMADA, 1999)) foi utilizada para extrair da nota fiscal letras e números e posteriormente foi preciso fazer um tratamento dos dados, visto que existem limitações no reconhecimento. Já no segundo foram utilizadas as mensagens SMS que a instituição financeira do usuário envia toda vez que uma transação ocorre. Embora essa solução seja mais confiável sobre os dados recebidos, são poucas as instituições que fazem envio de mensagem.

Em específico o aplicativo utiliza a técnica OCR, especializada em conhecimento de dígitos impressos ou escritos a mão (MORI; NISHIDA; YAMADA, 1999).

Figura 2 – NFC-E QRCode



Fonte: Secretaria de Fazenda do Estado do Rio de Janeiro

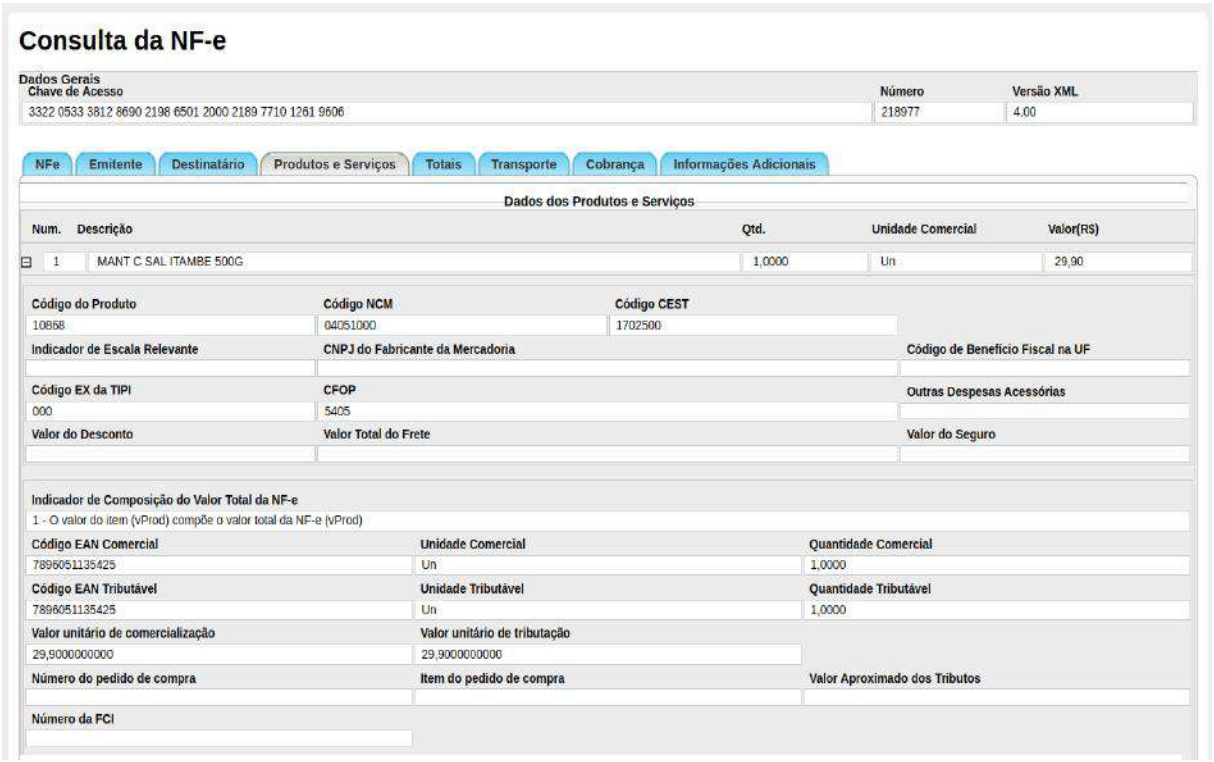
2.2 CÓDIGO DO PRODUTO

Abrindo o QRCode podemos chegar na página da Figura 2. Como é possível ver na Figura 2, cada um dos produtos impressos na nota fiscal possui um código relacionado. Esse código é conhecido como SKU (*Stock Keeping Unit*), um código usado pelos estabelecimentos para rastrear os produtos. Infelizmente esse código só é usado internamente por cada loja. Isso se apresenta como um problema, pois através desse código não seria possível relacionar o mesmo produto em lojas diferentes, dificultando a comparação de valores de produtos comprados em diferentes estabelecimentos. O processo de extração de dados será melhor explicado no Capítulo 3.

Entretanto, a NFC-e precisa ter alguma forma dos órgãos reguladores rastrearem os produtos, alguma classificação geral que não dependa de estabelecimento comercial ou entidade vendedora. A versão completa da NFC-e contém o NCM (*Nomenclatura Comum do Mercosul*). Esse código é uma convenção de categorização de mercadorias adotada desde 1995 pelos países pertencentes ao Mercosul e basicamente é uma forma de rastrear qualquer tipo de produto ou serviço que está envolvido numa venda comercial. Esse código

é único dentre os países do MERCOSUL, de forma a identificar os produtos. Para ter acesso a estes dados é preciso acessar a página http://www.nfce.fazenda.rj.gov.br/consulta e digitar a Chave de Acesso para obter uma página similar a da Figura 3.

Figura 3 – Consulta Completa NFC-E



Fonte: Secretaria de Fazenda do Estado do Rio de Janeiro

O NCM é usado para identificar os diversos tipos de produtos à venda no mercado, produtos iguais porém de marcas diferentes recebem o mesmo código. Assim como produtos da mesma marca, mas de tamanhos, cores ou sabores diferentes possuem o mesmo código. Por exemplo, o saco de lixo de 30l da nota fiscal das Figuras 1 e 2 possui código NCM 39232190, o mesmo para o saco de lixo de 50l. Para exemplificar a diferença entre SKU e NCM, o código SKU do saco de lixo no mercado carioca Zona Sul é 853666, enquanto que no mercado Hortifruti é 140201.

2.3 WEB SCRAPING

Web Scraping é o método pelo qual se extraem dados de páginas na Internet de forma automática (MUNZERT et al., 2014). Essa técnica é muito útil para extrair programaticamente dados acessando sites e possivelmente executando interações com o mesmo. É muito comum acessar o código HTML de sites, encontrar links entre os dados e enfileirar os endereços para repetir o processo de forma recursiva, além de realizar coletas. Também é comum utilizar bibliotecas que simulam a interação do usuário na tela para poder extrair dados de sites que fazem uso elementos dinâmicos.

Na construção desse trabalho foi utilizado este conceito para extrair dados da nota fiscal eletrônica através do link fornecido pelo QR Code presente no papel. A implementação dessa técnica produz o que é chamado de *scraper*. Um *scraper* é um programa desenvolvido utilizando bibliotecas ou ferramentas que fornecem abstrações para facilitar a interação com os elementos dos sites. Podem ser utilizadas desde as ferramentas mais básicas como Wget[2] ou cURL[3], até outras mais sofisticadas a exemplo do Selenium[4]. Diferente das primeiras, o Selenium consegue simular a interação do usuário na tela do site. Essa funcionalidade foi necessária pois a página dada pelo QR Code possui o carregamento lento e muitos elementos dinâmicos na tela, sendo impossível apenas utilizar o HTML.

[2] https://www.gnu.org/software/wget/
[3] https://curl.se/
[4] https://www.selenium.dev/

3 ARQUITETURA DO SISTEMA

A arquitetura do projeto foca em oferecer as funcionalidades listadas no capítulo introdutório bem como ser escalável, a fim de que o desenvolvimento seja feito de forma incremental e fácil. Para entender quais elementos se encaixam melhor no sistema, foi preciso abstrair os conceitos e entender o fluxo e as limitações existentes. O sistema como um todo precisa fornecer uma porta de entrada simples e prática para o usuário, para facilitar isso o QR code da nota fiscal facilitará a entrada de dados. Através do QR Code o sistema precisará extrair os dados da nota fiscal de forma automática e sem intervenção humana, entretanto precisará acessar não só a página da Fazenda, como também o site do estabelecimento para completar possíveis dados abreviados ou faltantes.

A Figura 4 mostra o fluxo de dados do sistema como um todo. Tudo se inicia com o usuário enviando os dados da nota fiscal de alguma forma para o sistema, caso o link da nota fiscal seja válido, um *scraper* fará o acesso ao site da Fazenda para extrair os dados da mesma e caso contrário o processo finaliza. Em seguida, após ter os dados extraídos o sistema checa se o estabelecimento da compra possui suporte, caso favorável um *scraper* especializado fará o acesso ao site da loja para completar os dados abreviados de cada um dos produtos. No caso em que a loja não é suportada pelo sistema, o processo também finaliza.

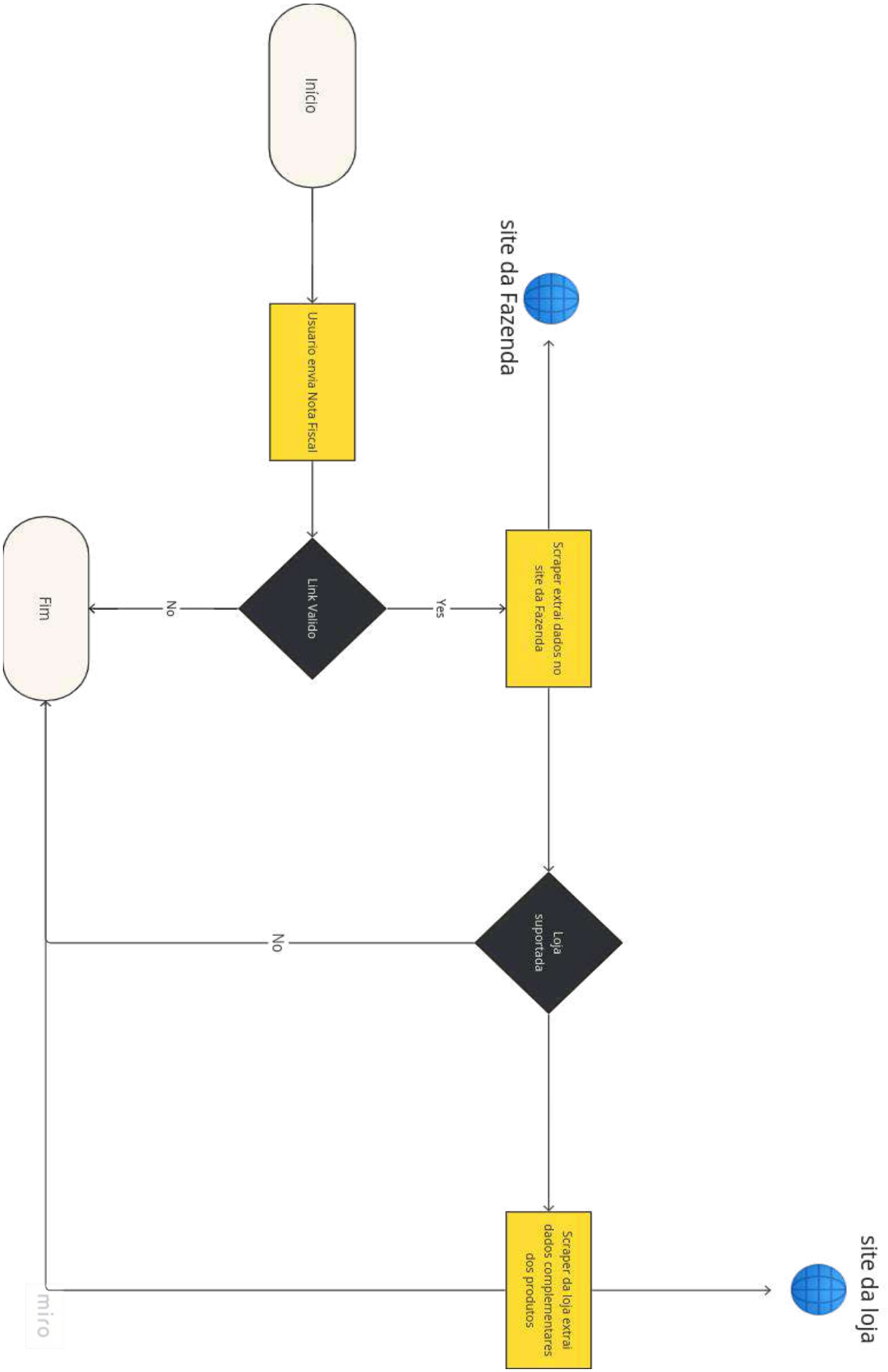


Figura 4 – Fluxograma de Execução

Para instanciar cada uma das etapas desse processo é necessário se utilizar de técnicas de programação que façam a experiência do usuário ser prática e simples, dessa forma a porta de entrada escolhida será um aplicativo de celular que consegue ler o QrCode da nota fiscal e enviar para o sistema. Dessa forma o usuário não precisa inserir manualmente os dados da compra e consegue ter de forma simples todas as informações dos produtos comprados. O aplicativo enviará o link da nota fiscal para um componente intermediário que fará a validação e inicia a execução das extrações, chamado de API, um acronimo para *Application Programming Interface* (Interface de Programação de Aplicação). Para os outros componentes será necessário que cada um deles seja autônomo e resiliente, de forma que consiga receber uma tarefa e executá-la de forma atômica e especializada. Dessa forma que a extração de dados tanto no site da Fazenda quanto da loja precisará ser feita por *scrapers* diferentes. Por fim, será necessário guardar tudo em uma base de dados para evitar execuções repetidas do sistema inteiro toda vez que o usuário desejar visualizar seus dados, trazendo lentidão e uma experiência ruim.

Na Figura 5 vemos os três principais componentes do sistema: um aplicativo de celular, o sistema de extração e a base de dados. O aplicativo funciona como principal ponto de interação do usuário, mas também é possível executar operações diretamente na API a partir de algum terminal ou cliente HTTP. A decisão de possuir esses 3 componentes principais se baseia na ideia de fornecer para uma interface *frontend* rápida e que não consuma processamento do aparelho pessoal para executar as extrações, de forma que essa responsabilidade recai no *backend* do sistema. Esse por sua vez precisará ser escalável e receber diversas tarefas, além de executar um rápido processamento das extrações de forma confiável e eficiente. Para alcançar esse estado será utilizado um sistema de filas de onde os scrapers retiram as tarefas para serem executadas. Para ilustrar, os estabelecimentos escolhidos foram supermercados conhecidos no cenário brasileiro, Zona Sul e Hortifruti.

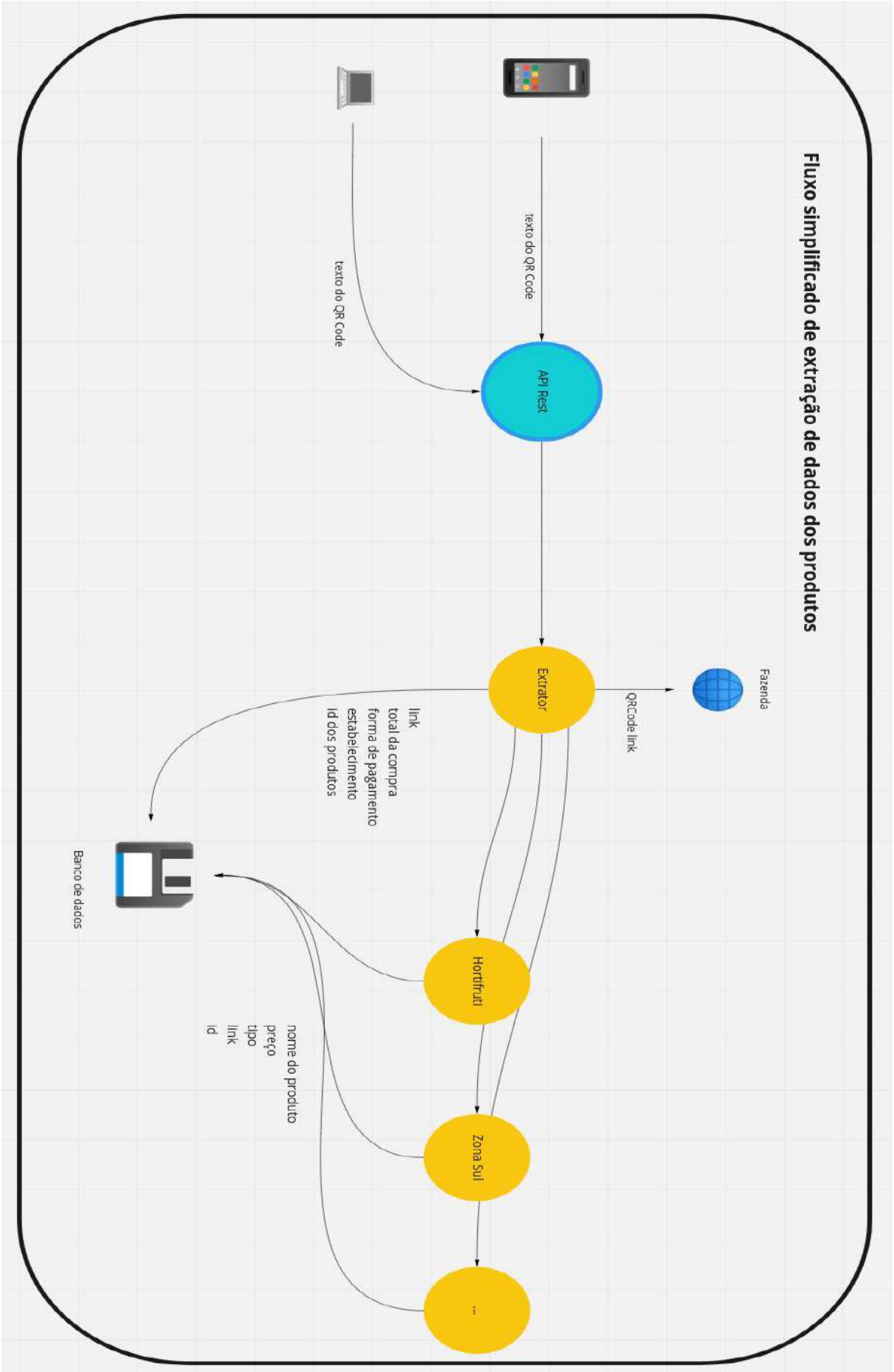


Figura 5 – Arquitetura simplificada

3.1 O APLICATIVO

O aplicativo de celular é a porta de entrada de dados no sistema, assim como o único meio de interação do usuário. Desenvolvido na linguagem Kotlin[1] utilizando os padrões e bibliotecas mais comuns da linguagem que nos ultimos anos se tornou a melhor para a construção de aplicativos de *smartphone*. Ele foi feito para fornecer em uma interface simples as principais informações de compras do usuário, assim como possibilitar a entrada de notas fiscais.

Ao abrir o aplicativo o mesmo executa uma busca no *backend* recuperando todas as notas fiscais cadastradas no sistema (Figura 7). Internamente o aplicativo implementa uma abstração das entendidades do sistema em classes. O aplicativo possui conhecimento de uma classe *Receipt*, a qual possui como atributos a *URL, loja, valor total, forma de pagamento, data da compra, logo da loja e lista de produtos*. Já a classe *Product* possui como atributos a *nome, tipo, quantidade, o tipo de unidade e o valor pago*, como mostra a Figura 6. Essa modelagem foi escolhida por seguir de forma simples o mesmo padrao de uma compra da vida real, onde de uma única vez podemos comprar varios produtos com quantidades diferentes de cada.

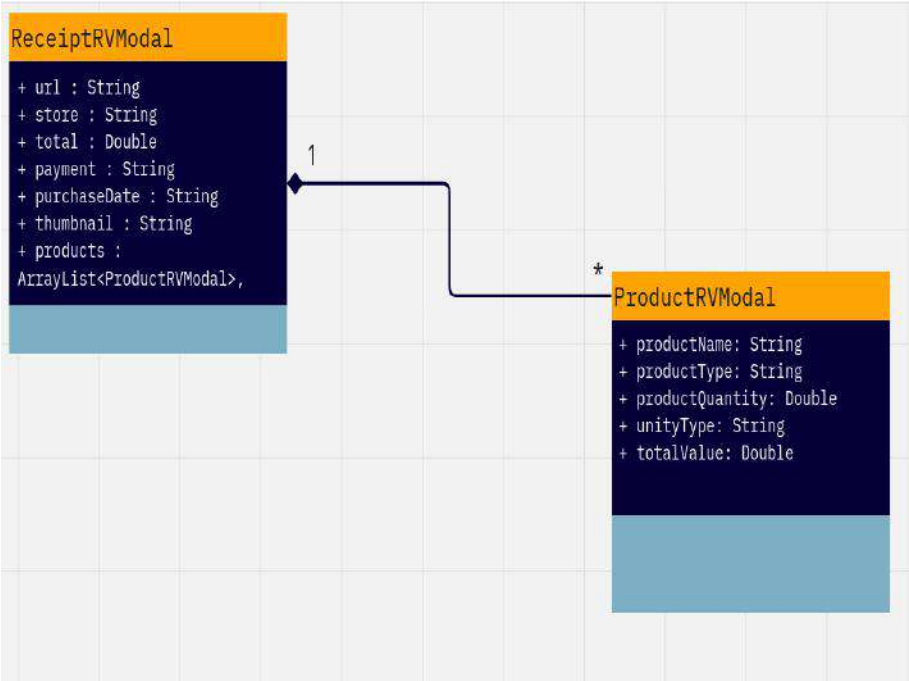


Figura 6 – Diagrama de Classes

As notas são mostradas em uma lista ordenada pela data, da mais nova para a mais antiga, independente do estabelecimento de origem. Nessa visualização são mostradas informações básicas para facilitar a identificação da compra pelo usuário. Informações

[1] https://kotlinlang.org/

como o nome, logo do estabelecimento e data são as que ajudam a identificar cada uma de forma independente.

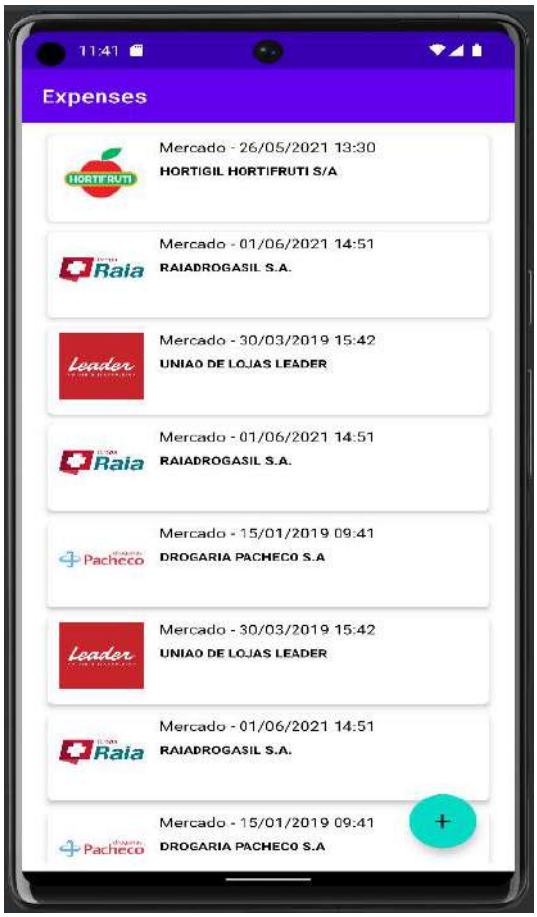


Figura 7 – Tela inicial do aplicativo

Ao adentrar em um dos itens da lista o usuário é apresentado a todas as informações da nota fiscal incluindo o valor total da compra e a forma de pagamento, além da quantidade total de produtos. Dessa forma, caso o usuário esteja procurando uma nota específica ele pode identificá-la pelo tamanho da compra.

Para complementar, é apresentada a lista de produtos comprados. Nessa lista é mostrado o nome, valor, unidade, quantidade e imagem (esta retirada diretamente do site do estabelecimento) como mostrado na Figura 8

Caso o usuário queira adicionar uma nova nota fiscal, o aplicativo fornece duas formas por meio de um Menu. Este menu pode ser acionado através do botão localizado no canto inferior direito, mostrando duas opções disponíveis: **Adicionar nota com chave de acesso** ou **Adicionar nota com QR Code** (Figura 9). A primeira opção abre para o usuário um pequeno formulário onde o mesmo pode digitar a Chave de Acesso presente na nota fiscal (Figura 10). Enquanto que a segunda opção abre para o usuário um leitor de QR Code, assim basta apontar para a nota e escanear (Figura 11). Ambas as opções

Figura 8 – Tela com detalhes da nota fiscal

utilizam as informações lidas para fazer uma requisição de adição na API, o que inicia o processo de extração no *backend*.

3.2 A API

Para funcionar como uma interface entre o aplicativo e as filas de extração, sendo o *backend* da interface do usuário, uma API HTTP REST fornece operações básicas nas principais entidades do sistema. O protocolo HTTP foi escolhido por garante para o sistema entrega de requisicoes, ou seja, quando o aplicativo se conectar e' garantida a entrega da mensagem enviada (SOCIETY, 1999). Já o padrão REST é um conjunto de práticas utilizadas para criar APIs com o intuito a fornecer através de seus comandos sempre o estado das entidades e dados gerenciados pelo sistema (FIELDING, 2000). Assim como os extratores de mercado servem de interface para os dados nos sites das lojas, a API fornece abstrações para que outros componentes do sistema trabalhem com os dados extraídos. Desenvolvida na linguagem Python (para manter a sinergia com os outros componentes) ela utiliza o framework FastAPI[2] para facilitar o desenvolvimento

[2] https://fastapi.tiangolo.com/

Figura 9 – Menu

das operações disponibilizadas.

Por ser uma API relativamente simples, foi tomada a decisão de utilizar o clássico modelo *MVC (Model View Control)*. Neste modelo de organização de projeto possuímos 3 tipos de componentes distintos que possuem funções bem definidas (REENSKAUG; COPLIEN, 2009). O Modelo agrupa todos os componentes que definem os dados, que na prática normalmente são as definições das classes das entendidades de dados do sistema.

No caso da API temos como modelo a representação de uma nota fiscal (por uma classe chamada de *Receipt*) e a representação de um produto (por uma classe chamada *Product*). A classe *Product* possui como atributos o *nome, código, tipo, tipo da unidade e a loja* da compra feita. A classe *Receipt* possui como atributos a *url, loja, total da compra, forma de pagamento, data e uma lista de produtos*. Essa lista de produtos é representada por uma classe de relacionamento chamada *ReceiptProductSchema* que encapsula um *Product* adicionando atributos como *quantidade da compra, tipo de unidade e valor*.

A View nesse modelo seriam as telas ou interface utilizadas pelos usuários. No caso específico da API pode-se dizer que não há View, dado que o usuário final não utiliza diretamente a API. Mas também é possível interpretar o aplicativo de celular como a View do conjunto App+API.

Figura 10 – Menu

Finalmente o Controle na API são os componentes que executam ações com os dados recebidos através do ReceiptControler, responsável por receber uma URL para extração e recuperar dados das notas fiscais no banco. Além disso, a API possui controladores responsáveis pelas operações com os componentes externos, nesse caso o banco de dados MongoDB e a fila em SQS, possuindo um controlador cada. O controlador de MongoDB encapsula toda a lógica necessária para inserir e recuperar dados no banco de dados e o controlador de SQS apenas controla o envio de URLs para a fila de extração.

Para enviar uma URL de nota fiscal para extração basta executar um POST na rota “/api/receipt"ou “/api/extract", enviando no payload um JSON com a URL da nota, a mesma fruto da leitura do QRCode. Esse JSON consiste apenas de um objeto com um unico atributo de chave ‘url’, como no exemplo.

```
\textit{POST /api/receipt}
{
    "url":"http://www4.fazenda.rj.gov.br/consultaNFCe/QRCode?p=33213049..."
}
```

Essa rota salva a URL recebida no banco de dados e a envia para a fila de extra-

Figura 11 – Menu

ção. Para recuperar os dados de uma nota fiscal basta executar um GET em "/api/receipt/<url>" passando a URL da nota fiscal. Adicionalmente a API permite recuperar todas as notas fiscais em "/api/receipts".

3.3 FILAS DE EXTRAÇÃO

A fim de tornar a execução do sistema mais rápida e eficiente utiliza-se um sistema de filas que torna todo o ambiente desacoplado. Cada um dos extratores explicitados abaixo pode ser executado em paralelo e simultaneamente sem a necessidade de implementar gestão de threads e sub-processos, se tornando um grande ambiente distribuído. Cada componente é independente dos outros e pode trabalhar de forma própria quando os dados são apresentados em forma de mensagens nas suas correspondentes filas.

Para o desenvolvimento deste trabalho foram utilizadas simples filas FIFO através do SQS (Simple Queue Service)[3] da AWS, que pode ser executado localmente ou na nuvem. Essa ferramenta foi escolhida pois não há a necessidade de nada além de uma simples fila

[3] https://aws.amazon.com/pt/sqs/

ou outras funcionalidades que ferramentas mais consolidadas como RabbitMQ[4] ou Kafka[5] fornecem, porém mais complicados de gerenciar e integrar, o que tornaria o sistema mais complexo. A Figura 12 mostra um diagrama completo dessa arquitetura de filas e as seções seguintes explicam cada um dos componentes.

[4] https://www.rabbitmq.com/
[5] https://kafka.apache.org/intro

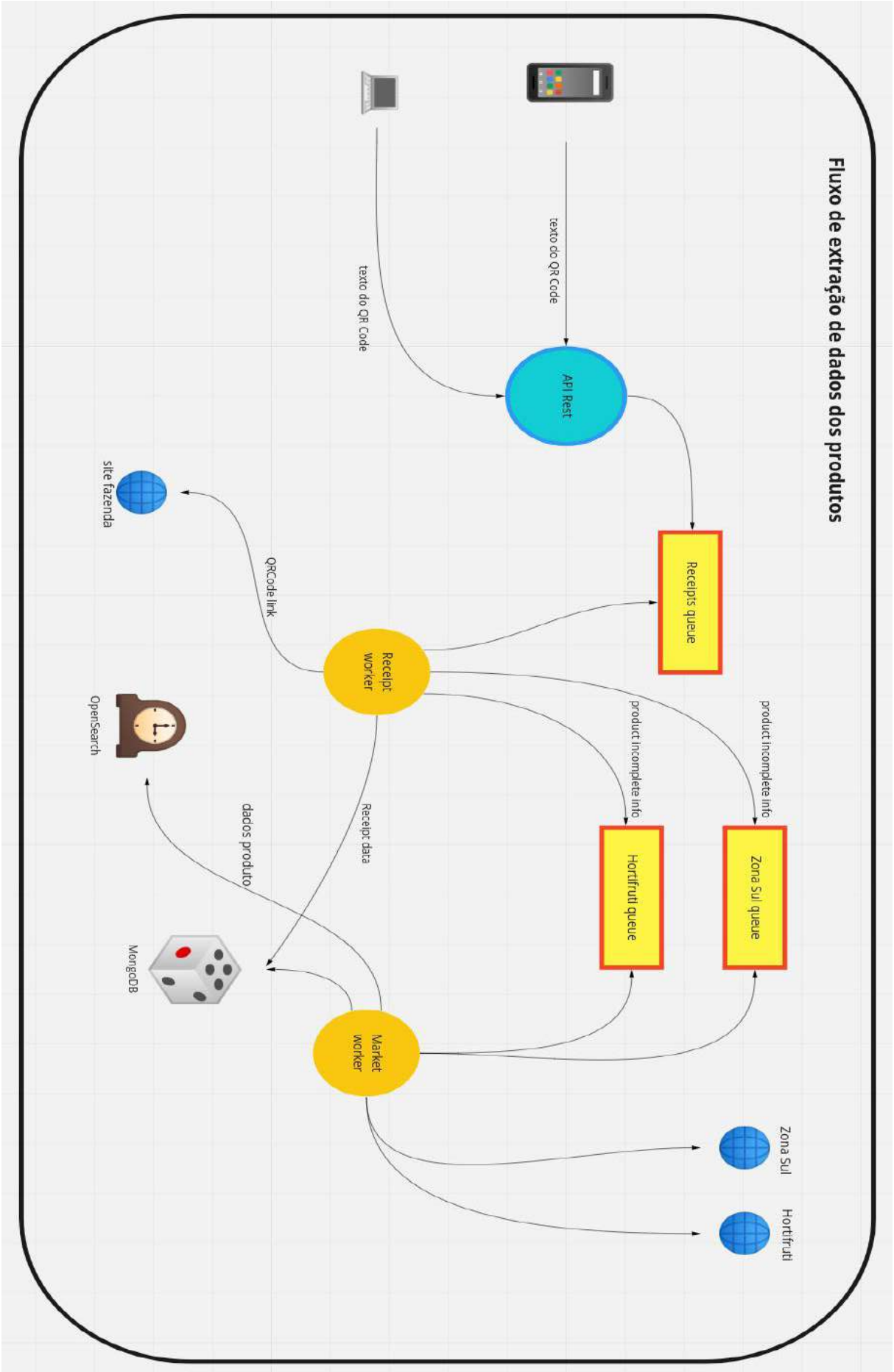


Figura 12 – Arquitetura completa

3.4 SCRAPERS

Os scrapers produzidos utilizam Selenium para acessar os sites, aguardar os elementos carregarem os dados e assim extrair as informações necessárias. Também foi considerado o uso dessa técnica para extrair os dados completos da NFC-e (como mostrado na Figura 3), porém o formulário de acesso possui um captcha, o que torna impossível chegar nos dados finais. Quanto aos dados da nota fiscal, o mais importante é ter formas de identificar corretamente um produto para o usuário. O código do produto é uma boa forma de identificá-lo, mas esse número não possui significado para um usuário comum, de forma que o nome real da mercadoria seria necessário. Como pode ser visto na Figura 2 o nome do produto é mostrado na nota fiscal truncado com abreviações e esse padrão não poderia ser usado para identificar um produto.

Para conseguir identificar o produto corretamente foram construídos scrapers específicos para cada estabelecimento suportado por este trabalho e para iniciar foram elencados os mercados Zona Sul e Hortifruti. As razões para a escolha dos mesmos foram basicamente possuírem um website robusto e atualizado. Esses dois scrapers recebem como parâmetro o código do produto (SKU) e acessam o site do mercado para procurar por informações específicas do mesmo, como nome sem abreviações e tipo de produto. Cabe ao extrator principal, o mesmo que extrai os dados da NFC-e, enviar para a fila correta do extrator de mercado com base no estabelecimento da nota.

Cada um dos extratores de mercado possui uma heurística muito semelhante: receber o SKU da fila, acessar a página do mercado, pesquisar por um código e coletar os dados do primeiro produto encontrado, ou não fazer nada caso não tenha encontrado. Foram identificados alguns casos de notas fiscais que possuem produtos cujo SKU não existe no site. Esses casos ou foram notas muito antigas cujo produto foi reidentificado internamente no mercado ou correspondem a itens produzidos e vendidos de forma natural numa certa loja, mas que não estão presentes no site. Para esses casos o produto é salvo usando o nome que está presente na NFC-e, mas não possui a informação adicional do tipo.

3.4.1 O extrator de NFC-e

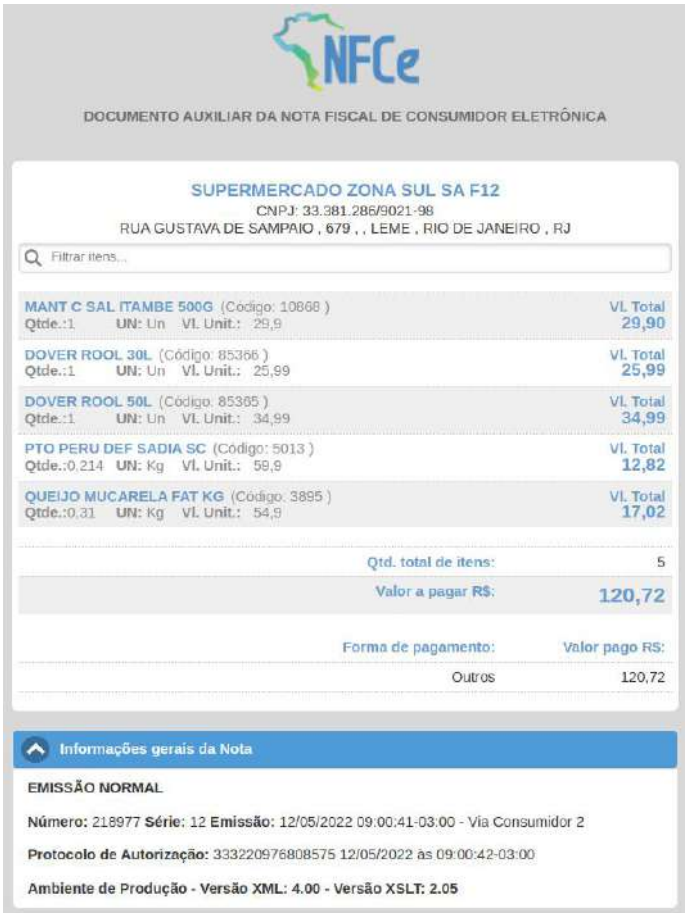
A nota fiscal é o ponto de partida para a coleta de qualquer informação usada para análise, de forma que ter uma forma automatizada de extrair seus dados é fundamental para a escalabilidade do sistema como um todo. Conforme explicado no Capítulo 2, o QR Code presente na nota impressa representa o link para a versão eletrônica da NFC-e, página essa que contém um pouco mais de informações, como por exemplo o NCM.

O extrator principal consome uma fila onde a API insere esse link recebido. Os dados da nota fiscal são extraídos por meio de um web scraper usando Selenium[6]. O uso dessa técnica e ferramenta se justifica para esta página pois seu conteúdo é carregado

[6] https://www.selenium.dev/documentation/webdriver/getting$_started$/

dinamicamente, ao acessá-la, inicialmente é mostrado um pop-up enquanto os dados são carregados. Dessa forma não é possível simplesmente fazer *download* do código HTML e encontrar os dados. É preciso usar uma ferramenta que consiga esperar o aparecimento de algum elemento HTML específico, simulando um usuário real. No caso deste extrator, o script aguarda a presença de uma *div* identificada como "conteudo", que contém as informações necessárias. Assim é possível inferir o carregamento completo da página (como mostra a Figura 13) e extrair os dados.

Figura 13 – NFC-E



Fonte: Secretaria de Fazenda do Estado do Rio de Janeiro

Algumas informações como loja e total da compra são encontrados através de sua identificação por classes CSS únicas, porém outras como forma de pagamento e data não são mostradas da mesma forma e por isso foi preciso usar seu XPATH para encontrá-las, que é basicamente acessar o caminho do elemento dentro da árvore de elementos da página. Já para os produtos comprados, como se encontram em uma tabela HTML, sua extração foi mais simples, pois cada linha da mesma possui um identificador único. Assim foi preciso apenas programar um loop capaz de iterar pelas linhas até a última.

A nota fiscal completa é persistida no banco de dados de documentos como um objeto contendo:

a) o link do QRCode

b) estabelecimento

c) valor total

d) forma de pagamento

e) data da compra

f) uma lista vazia, mas que será populada paralelamente pelos extratores de mercado com um objeto contendo os identificadores referenciando os produtos, valor unitário, quantidade e valor total

Por fim, cada produto dessa lista é enviado para a fila correspondente do mercado, cada um em uma mensagem separada, sem uma ordenação especifica. Esta mensagem contém:

a) o link do QRCode - utilizado pelo extrator para localizar o objeto da nota fiscal

b) nome do produto

c) SKU

d) quantidade

e) tipo da unidade (kg ou unitário)

f) valor unitário

g) total

h) data da compra

i) estabelecimento

Caso ocorra algum erro nessa extração, o link de QRCode é enviado para uma fila de erros para que ocorra retentativa ou tratamento.

3.4.2 Os extratores de mercado

Conforme explicado no Capítulo 2, a NFC-e não possui informações completas sobre os produtos, a exemplo do nome que pode aparecer truncado. Além disso, o código SKU só vale como identificador dentro de cada estabelecimento. Justifica-se assim a implementação de extratores específicos e para isso foram escolhidos alguns supermercados grandes cujo site permitia pesquisa programática fácil. Esses mercados foram o Hortifruti e o Zona Sul, enquanto outros também populares como Guanabara e Mundial não possuem pesquisa em suas páginas na web. Além disso, os extratores são dependentes dos sites, se os mesmos não estiverem disponíveis, a extração falha e o produto é enviado para uma fila de retentativas.

Para ambos os sites a heurística de pesquisa foi bem similar, pois ambos possuem estruturas bem parecidas (provavelmente implementados pela mesma empresa). Esses

extratores ficam aguardando mensagens em suas filas, mensagens estas que contêm as informações do produto e o link do QRCode. Ao consumir uma mensagem os scrapers utilizam Selenium para coletar os dados detalhados que a NFC-e não possui. Como não é possível usar o nome coletado na nota fiscal foi necessário usar o SKU, que acaba por fornecer um melhor resultado, pois é um código único e apenas em poucos casos a busca retorna mais de 1 resultado. Caso a busca retorne mais resultados, o primeiro da lista é o escolhido. A Figura 14 mostra um exemplo de onde utilizar o nome não tem resultado único enquanto a Figura 15 mostra que o SKU é mais eficiente.

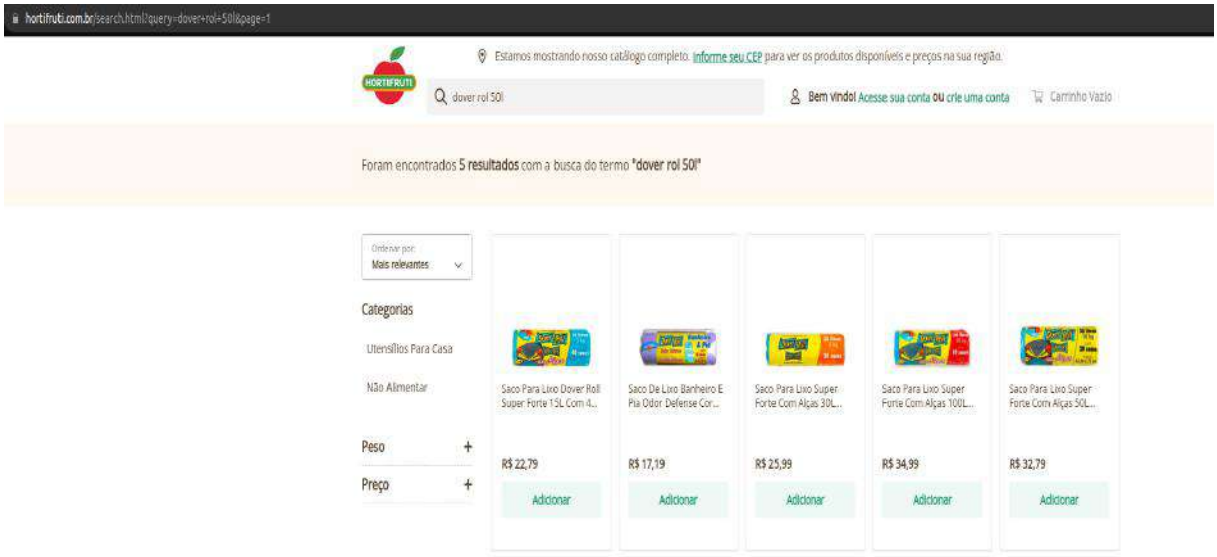


Figura 14 – Busca por nome no Hortifruti

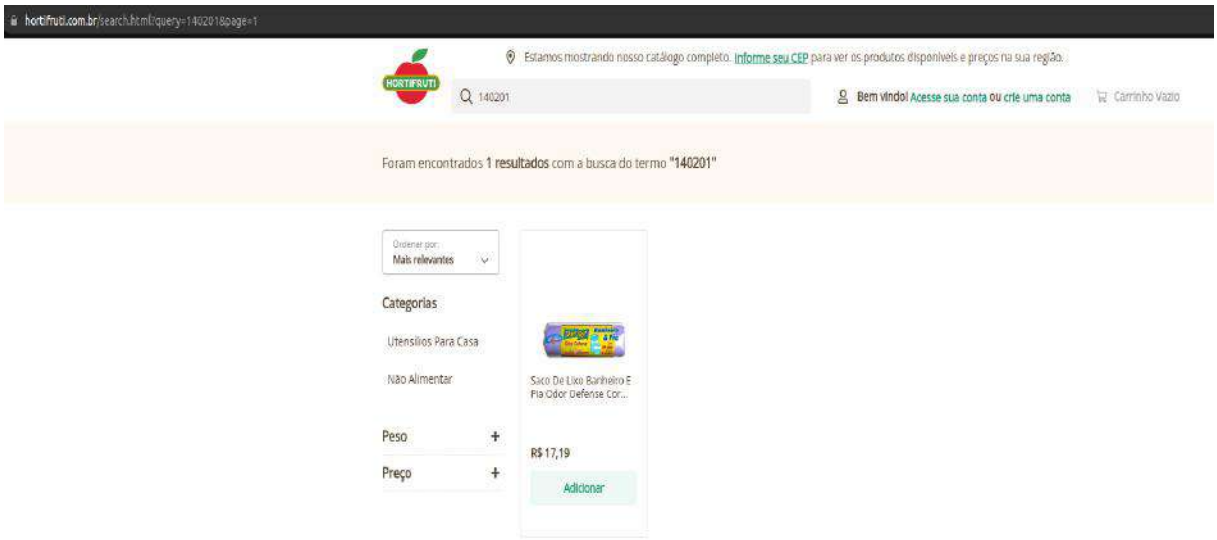


Figura 15 – Busca por SKU no Hortifruti

Para coletar os dados são utilizadas classes CSS para conseguir o nome completo do produto e o tipo. Por serem similares, foi possível fazer a coleta da mesma forma em ambos os supermercados, diferenciando, obviamente, os nomes das classes. Após conseguir os dados, cada produto é representado como um objeto salvo no banco de dados de documentos com os seguintes atributos:

a) nome

b) SKU

c) tipo

d) unidade (kg ou unitário)

e) estabelecimento

Ao final do processamento o extrator pesquisa no banco de dados pelo objeto correspondente à nota fiscal e nele atualiza a lista de produtos adicionando um objeto contendo:

a) identificador do objeto do produto dentro do banco

b) preço unitário

c) quantidade do produto na compra

d) e valor correspondente

Durante a extração, o script salva num banco de dados temporal as informações atualizadas do produto com adição de algumas informações referentes a compra, como data, valor da compra e quantidade comprada. Essa base pode ser utilizada para analisar quando as compras acontecem e qual a frequência de compra de cada produto. Na seção 4.3 são mostrados exemplos dessa análise. Caso ocorra algum erro em todo o processo, o link da nota fiscal recebido inicialmente é enviado para uma fila de erros para que ocorra retentativa ou tratamento.

3.5 O BASE DE DADOS DE DOCUMENTOS

Todos os dados coletados pelos extratores são persistidos em um banco de dados de documentos mesmo que não sejam necessariamente utilizados. Os motivos para isso são claros:

a) Evitar a necessidade de executar novamente os extratores toda vez que os dados forem necessários. Isso adicionaria tempo na execução de qualquer funcionalidade, além de depender dos sites consultados manterem seus formatos depois de muito tempo.

b) Esses dados podem ser utilizados para outras funcionalidades no sistema, como por exemplo mostrar uma lista das últimas compras do usuário no aplicativo ou consulta de compras feitas.

c) Possuir esses dados salvos em uma base desacopla o sistema para que outros componentes se integrem facilmente

Esse modelo de banco de dados (dados representados como documentos, conhecidos como NoSQL[7]) se difere do mais comum no mercado (entidade relacionamento) por não forçar ao usuário salvar dados no formato de linhas em tabelas ou possuir atributos chave. Muitas das implementações desse modelo possuem atributos chave e relacionamento entre os documentos, mas a principal quebra de paradigma é dar ao desenvolvedor a praticidade de ter as mesmas estruturas de dados existentes em tempo de execução sendo persistidas dentro da base. Ao tratar os dados como documentos é possivel persistir os mesmos objetos, listas e dicionários que existem no programa.

Esse trabalho faz uso dessas funcionalidades para corretamente persistir os dados nas estruturas que mais fazem sentido. Por exemplo, uma nota fiscal é um documento que possui vários atributos chave-valor e uma lista de produtos, onde cada um pode ser interpretado como objeto chave-valor por si só, conforme mostra a Figura 16. Ao transpor para a base de dados é possível manter essa mesma forma organizacional, tornando o desenvolvimento mais fácil. Além disso utiliza-se a funcionalidade de relacionamento entre documentos conectando um atributo de um documento com o identificador de outro. Este é o caso da nota fiscal, que possui uma lista dos identificadores de produto.

Este trabalho utiliza a ferramenta MongoDB[8], uma implementação do modelo de base de documentos já consolidada no mercado e de fácil integração ao utilizar sua biblioteca em Python, a pymongo[9].

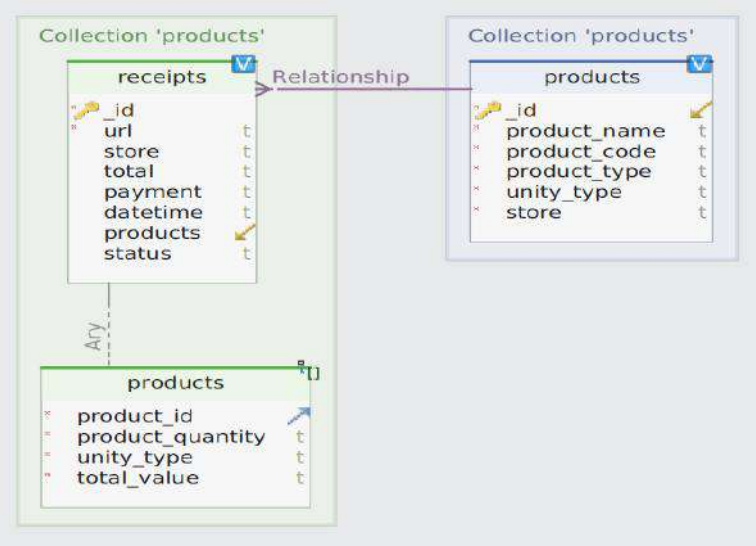


Figura 16 – Diagrama de esquema de banco de dados

[7] https://www.ibm.com/br-pt/topics/nosql-databases
[8] https://www.mongodb.com/home
[9] https://www.mongodb.com/docs/drivers/pymongo/

3.6 BASES DE DADOS TEMPORAIS

Bases de dados temporais são um tipo de bases de dados cujo formato é otimizado para armazenar pares de datas e dados. Normalmente são muito utilizados em tratamento de dados de clima, relacionando temperatura, pressão e umidade do ar com momento no tempo (timestamp).

Nesse trabalho esse tipo de base é utilizado para relacionar uma compra (constituída de itens, quantidade, preços e estabelecimento) com uma data no tempo. A ferramenta especifica utilizada que implementa esse tipo de base foi o Opensearch[10], versão Open Source do famoso Elasticsearch[11]. Nessa ferramenta é possível guardar entradas de ocorrência de uma compra na linha do tempo. Por exemplo, é possível guardar um documento JSON dizendo que no dia 30 de outubro de 2023 às 10:54 am foi comprado no mercado zona sul um cacho de banana custando R$ 5,00.

O Opensearch fornece as funcionalidades de busca e base temporal necessárias para executar análises sobre as compras feitas. Dessa forma o objetivo é conseguir gerar consultas otimizadas através do poder desse tipo de base e achar recorrências nas compras feitas por um usuário. Para esse trabalho os dados dos produtos são salvos em um índice próprio dentro da ferramenta e durante o desenvolvimento a interface gráfica foi utilizada para validação.

[10] https://aws.amazon.com/pt/what-is/opensearch/
[11] https://aws.amazon.com/pt/what-is/elasticsearch/

4 MÉTRICAS E RESULTADOS

Neste capítulo é apresentada uma análise dos tempos de execução de cada um dos componentes do sistema. Fazer essa análise se faz necessário para entender se cada parte do sistema está funcionando como esperado ou se será preciso alguma otimização. Além disso, durante a construção do projeto foi possível testar diversas vezes como usuário final para validar a experiência e a eficiência das funcionalidades.

4.1 EXECUÇÃO DO EXTRATOR PRINCIPAL

Durante o desenvolvimento, diversas execuções do sistema foram feitas de cada componente isolado. Dessa forma se fez necessário testar também a execução de toda a esteira de extração de dados diversas vezes e medir a duração de cada elemento, identificando gargalos e pontos de falha.

Para gerar as métricas necessárias foram utilizadas 38 notas fiscais ao longo da construção deste trabalho e foram coletados os tempos de execução desde o início do projeto. Com essas notas fiscais foi possível obter a execução da extração de 72 produtos de ambos os mercados escolhidos, gerando 210 extrações ao todo. Assim como foi possível identificar diversos errors e gargalos que foram corrigidos ao longo do tempo.

Para facilitar a execução do sistema por completo, componentes de persistência, como os bancos de dados Mongo e Opensearch e as filas foram executados localmente, junto dos extratores. Seria possível implantar esses componentes em serviços de nuvem conhecidos como AWS[1] ou GCP[2], mas executando localmente fatores como latência de rede, problemas nos provedores de serviço e camadas de interface são evitados. A execução destes é feita usando Docker[3] para garantir o isolamento até mesmo dentro do computador que executa os extratores, garantindo que tais processos receberão os recursos de CPU e memória necessários.

A Figura 17 mostra a distribuição do tempo de execução total das 38 notas fiscais utilizadas no exemplo. Na imagem o eixo X representa o tempo em segundos que a execução demorou, enquanto que o eixo Y representa a quantidade de execuções nos intervalos de tempo. Nesse caso foram coletados o tempo total desde início da execução, passando pelo acesso a página da Receita, persistência no MongoDB e o envio dos produtos extraídos para fila de produtos. É possível perceber que a maioria das execuções demoram em média 5,1 segundos, possuindo alguns momentos de instabilidade em que pode durar até 8 segundos.

[1] https://aws.amazon.com/pt/
[2] https://cloud.google.com/?hl=pt-BR
[3] https://www.docker.com/resources/what-container/

É importante pontuar que o site da Receita Federal possui alguns mecanismos de identificação de robôs e infelizmente não foi possível entender como isso funciona. Entretanto em algumas situações (menos de 5% do total), o site negou completamente a requisição, fazendo o script atingir o tempo limite de inatividade (60 segundos). Essas ocorrências foram removidas da distribuição para não poluir o gráfico. Porém, na execução normal do sistema, o extrator salva o endereço do QR Code numa fila de retentativas.

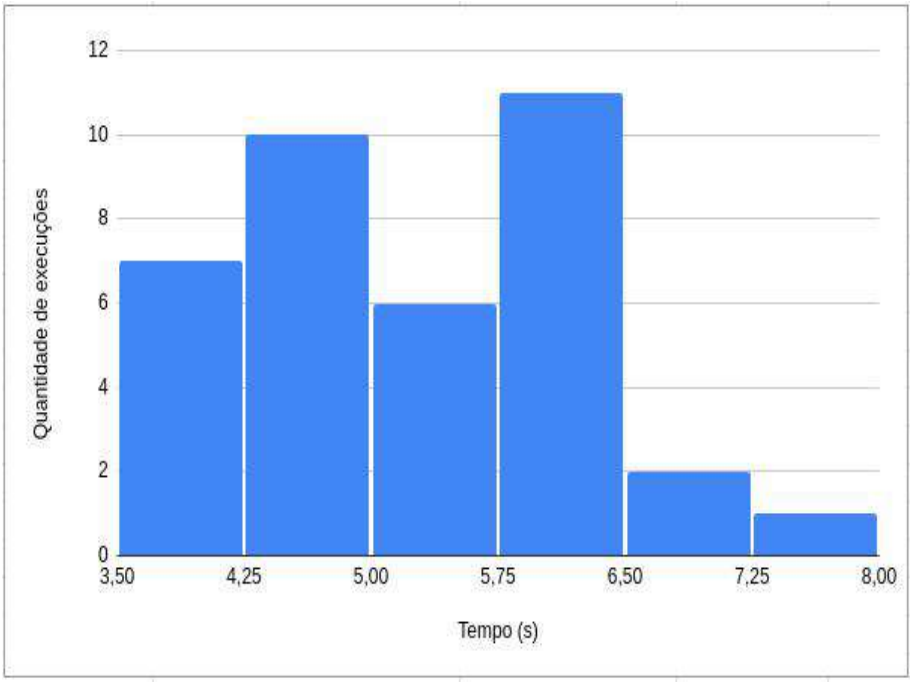


Figura 17 – Tempo de extração total

Na Figura 18 é mostrada a distribuição do tempo necessário para persistir no MongoDB os dados da nota fiscal extraídos. Nesse caso, como os componentes estão no mesmo computador físico, apenas é levado em conta o tempo real levado para salvar no banco. Essa tarefa obviamente é muito mais rápida que acessar o site da Receita Federal, e a maioria das execuções levam em média 44 milissegundos, mas também houve algumas que levaram até 50 milissegundos. Esses casos mais lentos são as notas fiscais com maiores listas de produtos (em média 30), mas embora seja mais demorado, ainda é um tempo muito pequeno.

O tempo necessário para enviar as informações de 1 produto para a fila durou em média menos de 10 milissegundos, o que se justifica pelo fato do SQS ser um serviço muito simples, além do já mencionado fato dos dois processos compartilharem a mesma máquina.

Para completar, a Figura 19 mostra a distribuição do tempo necessário para apenas acessar o site da receita e extrair os dados. É possível perceber que essa é a parte mais demorada, dado que demanda acesso a um sistema terceiro, do qual não se possui controle.

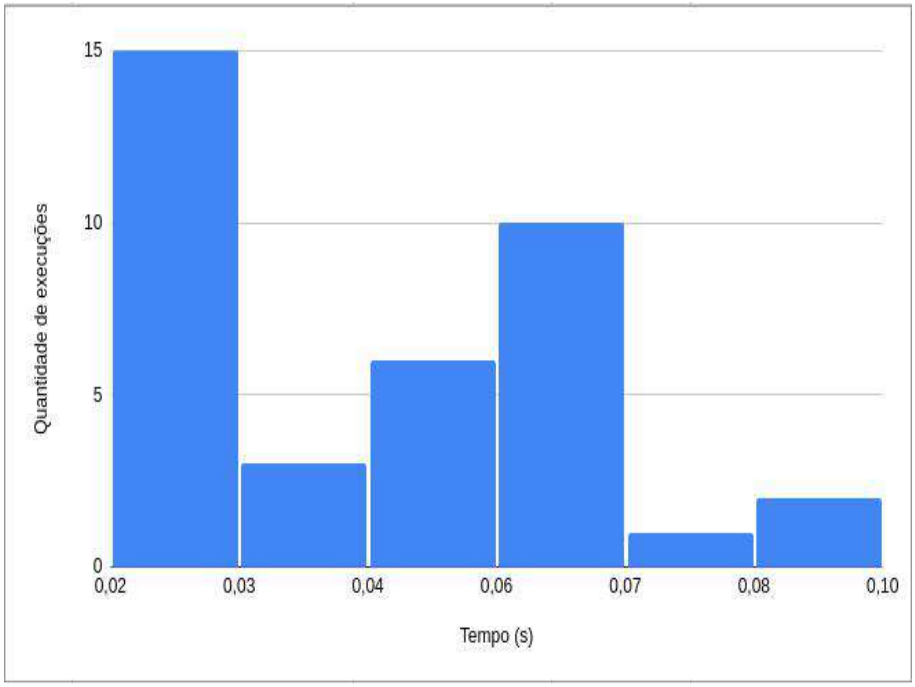


Figura 18 – Tempo para persistir no banco

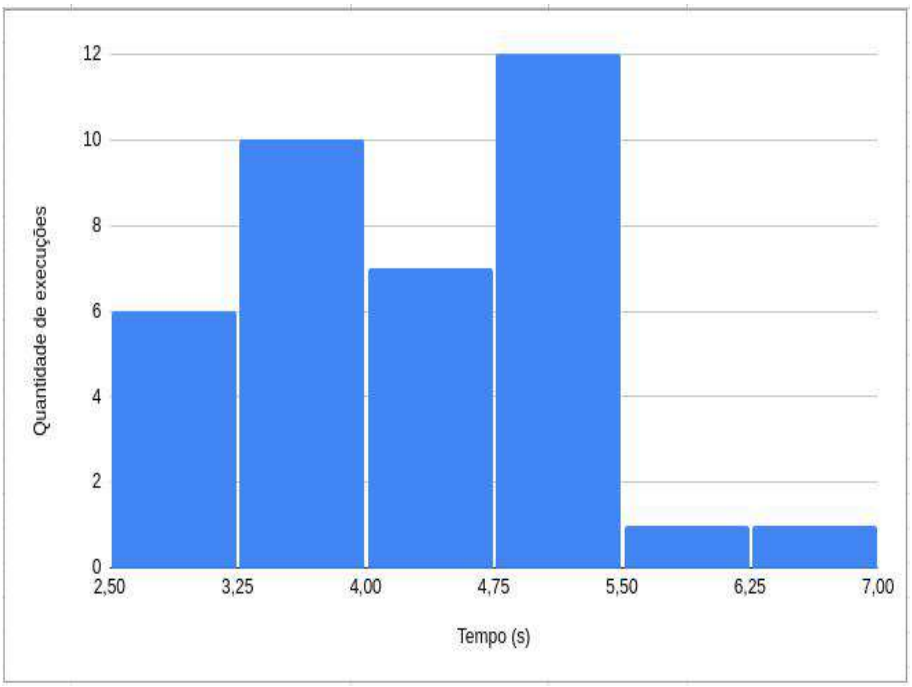


Figura 19 – Tempo para buscar dados no site da Receita Federal

4.2 EXECUÇÃO DOS EXTRATORES DE MERCADO

Para medir a execução dos extratores de mercado, foram aplicadas as mesmas técnicas e heurísticas utilizadas no extrator principal. Foram medidos os pontos principais dos scripts: extrair os dados do produto no site do mercado, salvar no Opensearch, salvar o produto no MongoDB e atualizar o objeto da nota fiscal também no banco.

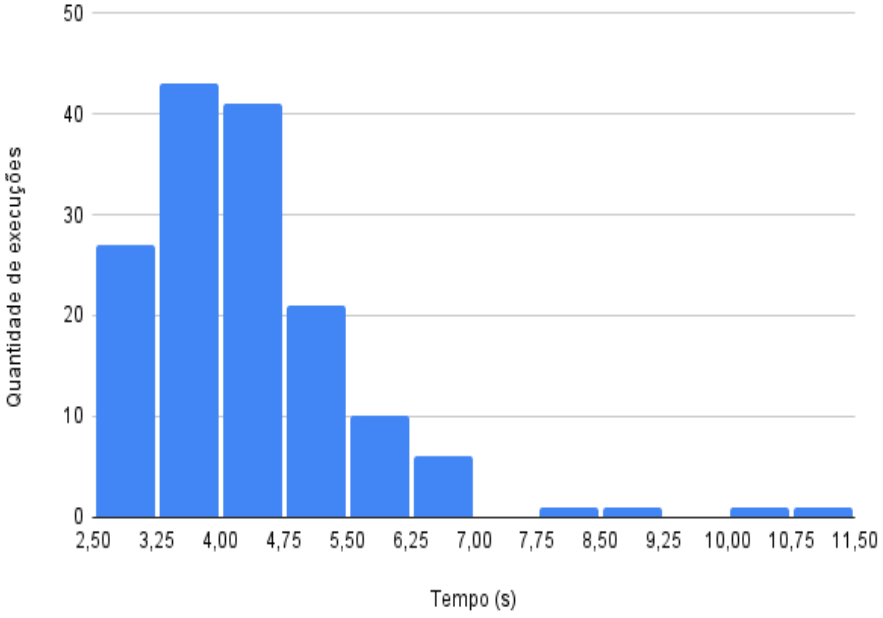


Figura 20 – Extrator de Hortifruti - Tempo total de extração

Iniciando a análise através do extrator do mercado Hortifruti, foram um total de 153 extrações. A Figura 20 mostra a distribuição do tempo total para executar o processo inteiro. Nela é possível entender que a extração inteira durou em média 4,07 segundos. Dentro desse tempo, a grande parte é gasta com o acesso ao site do mercado, como mostra a Figura 21. As outras etapas da execução duram sempre na casa de milissegundos, conforme mostram as Figuras 22, 23 e 24. Em alguns casos para salvar no opensearch levou-se quase 1 segundo inteiro, justificando os casos em que a extração demora, mas é possível perceber que ambas as operações executadas no MongoDB duram em média o mesmo tempo, 38 milissegundos.

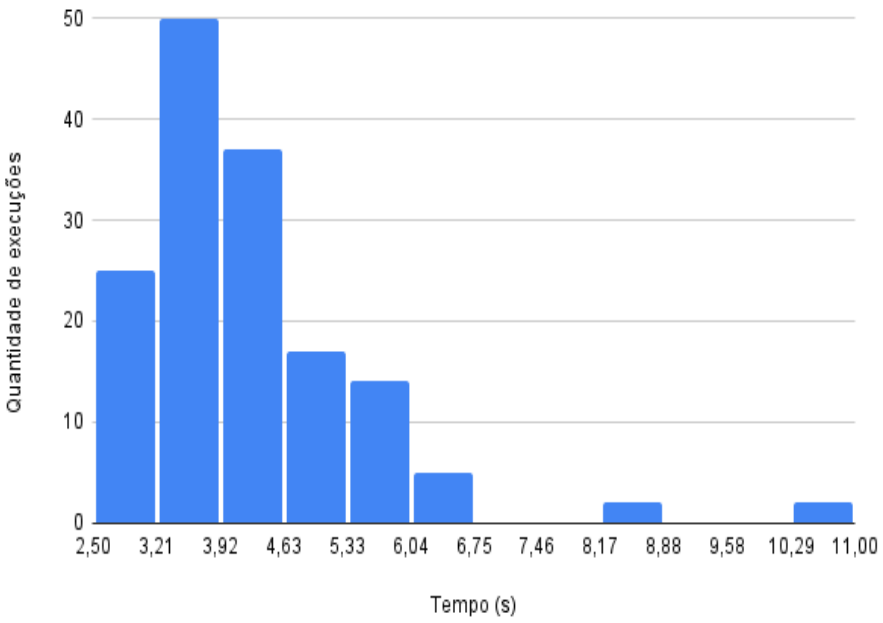


Figura 21 – Extrator de Hortifruti - Tempo de extração no site

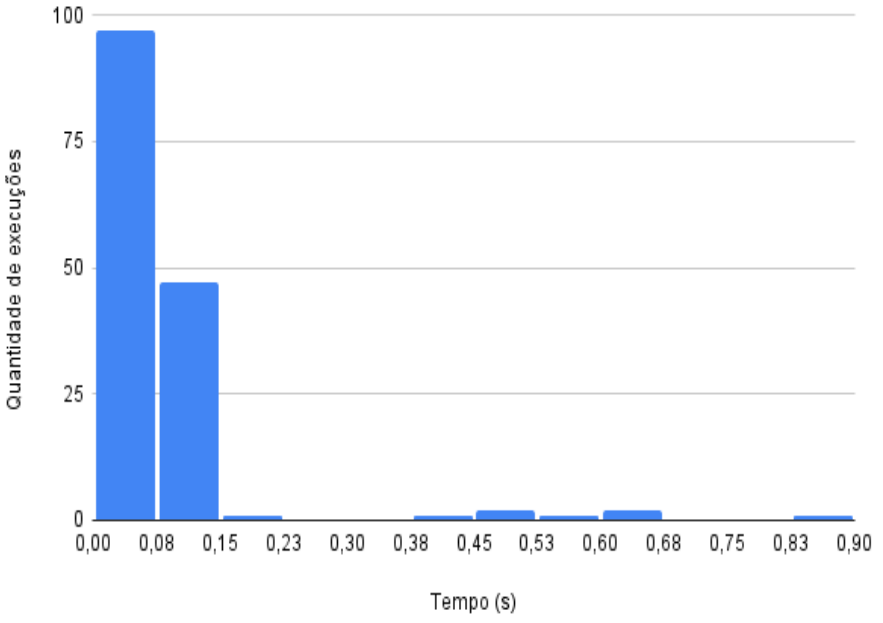


Figura 22 – Extrator de Hortifruti - Tempo para salvar no Opensearch

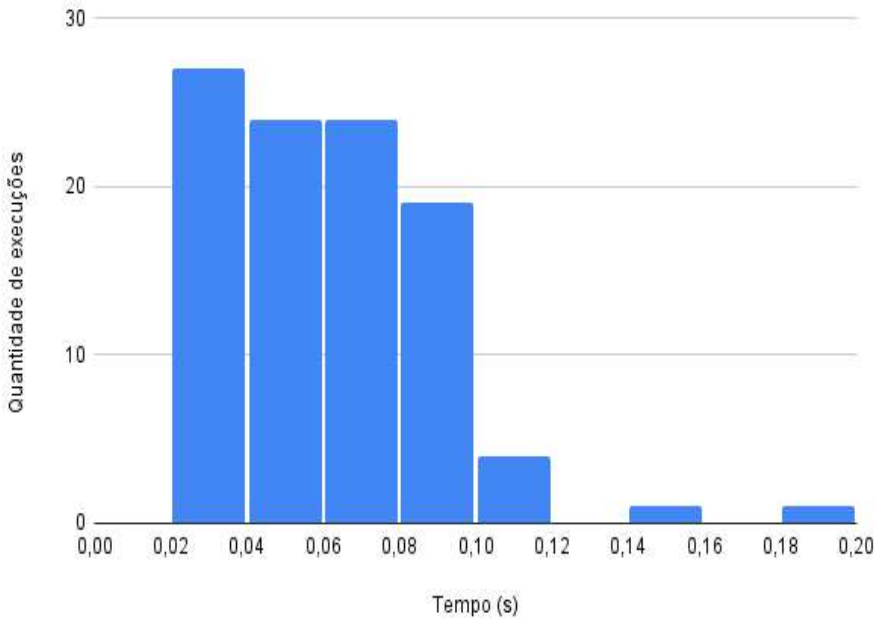


Figura 23 – Extrator de Hortifruti - Tempo para salvar no MongoDB

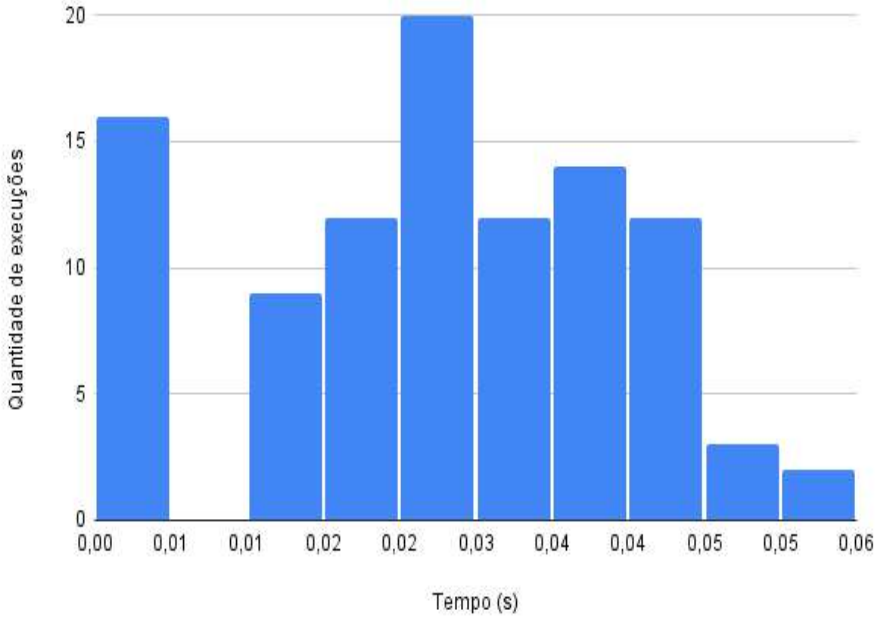


Figura 24 – Extrator de Hortifruti - Tempo para atualizar nota fiscal no MongoDB

Analisando as métricas do extrator do mercado Zona Sul, embora tenham sido feitas apenas 68 execuções, é possível perceber que normalmente o site demora ao menos 30% a mais para responder que o do Hortifruti. Enquanto o site do Hortifruti responde em média em 3,9 segundos, o site do Zona Sul demora em média 8,86 segundos para responder. São poucos exemplos de execuções rápidas na casa dos 4 segundos, como mostra a Figura 25. Com isso o tempo total de extração também segue alto, como mostra a Figura 26, tendo média de execução na casa dos 9,39 segundos.

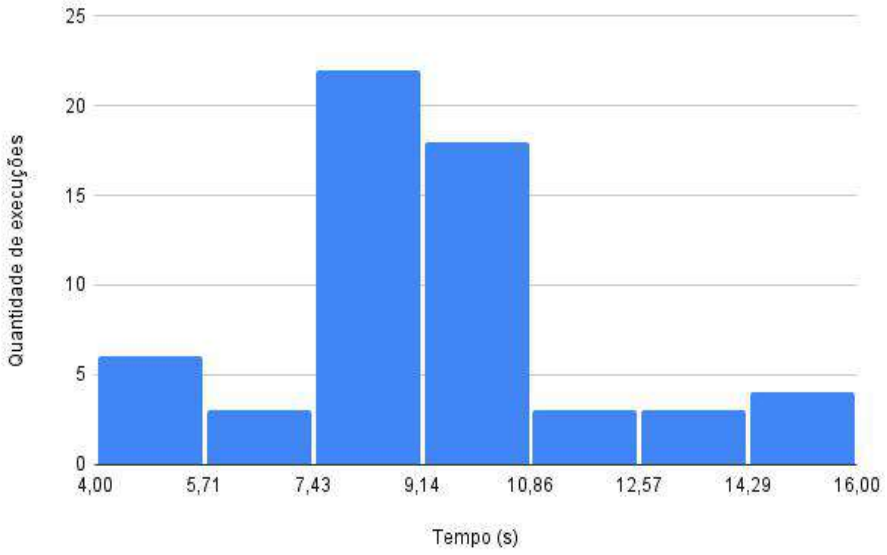


Figura 25 – Extrator de Zona Sul - Tempo de extração no site

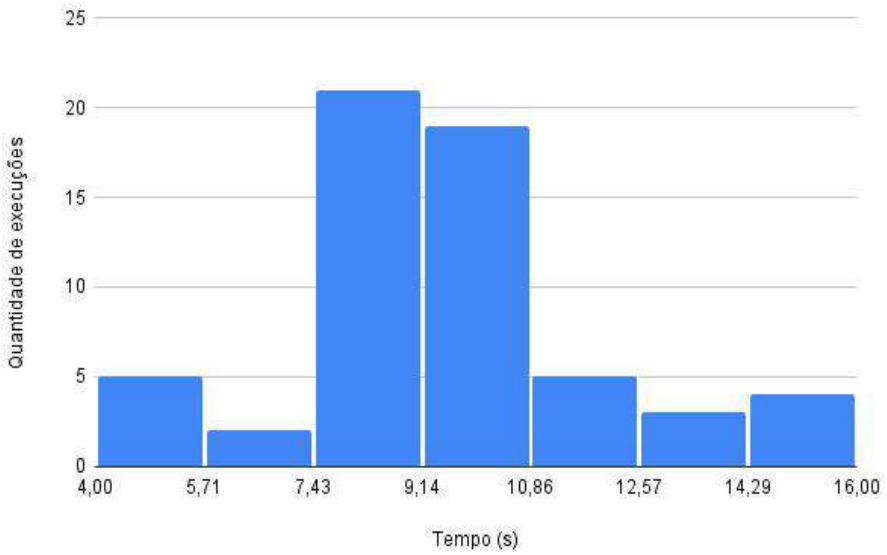


Figura 26 – Extrator de Zona Sul - Tempo total de extração

Assim como no extrator de Hortifruti, salvar no Opensearch demora até 200 milissegundos na grande maioria dos casos (Figura 27) e as operações no MongoDB também seguem a mesma linha, com mediana de 50 milissegundos para salvar um produto novo e de 20 milissegundos para atualizar o documento de nota fiscal com a referência do produto extraído (Figuras 28 e 29).

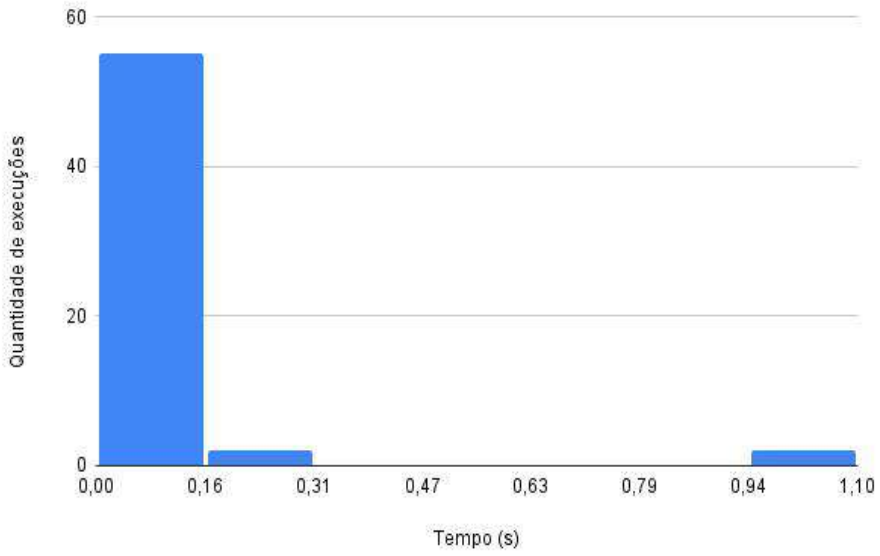


Figura 27 – Extrator de Zona Sul - Tempo para salvar no Opensearch

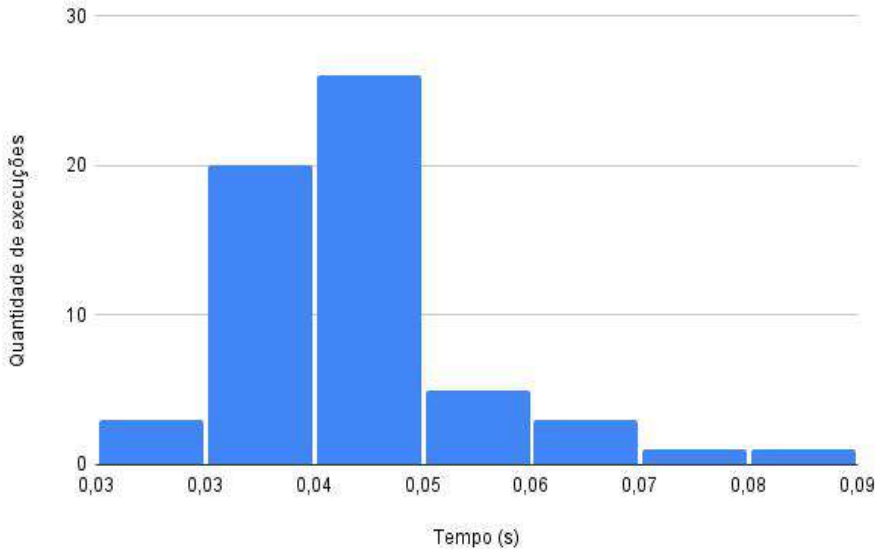


Figura 28 – Extrator de Zona Sul - Tempo para salvar no MongoDB

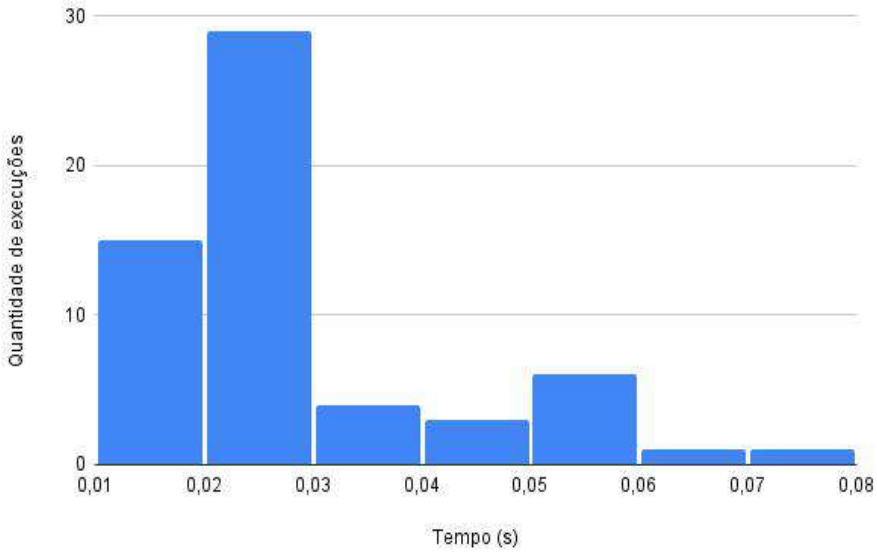


Figura 29 – Extrator de Zona Sul - Tempo para atualizar nota fiscal no MongoDB

4.3 O ANALISANDO COMPRAS

Conforme mostrado nas seções anteriores, todo produto extraído pelos scripts produzidos além de serem salvos no MongoDB também são persistidos no Opensearch (seção 3.4.2). Além de servir como uma base temporal e fornecer um motor de busca, o Opensearch também fornece, através de uma interface gráfica, a possibilidade de visualizar os dados salvos em uma linha do tempo, conforme mostra o exemplo da Figura 30 sem nenhum filtro de busca. Embora a funcionalidade de análise de recorrência de gastos, que utilizaria essa ferramenta, tenha sido despriorizada e não tenha sido implementada durante a construção desse projeto, os dados nela salvos foram úteis para análises das extrações feitas durante o desenvolvimento.

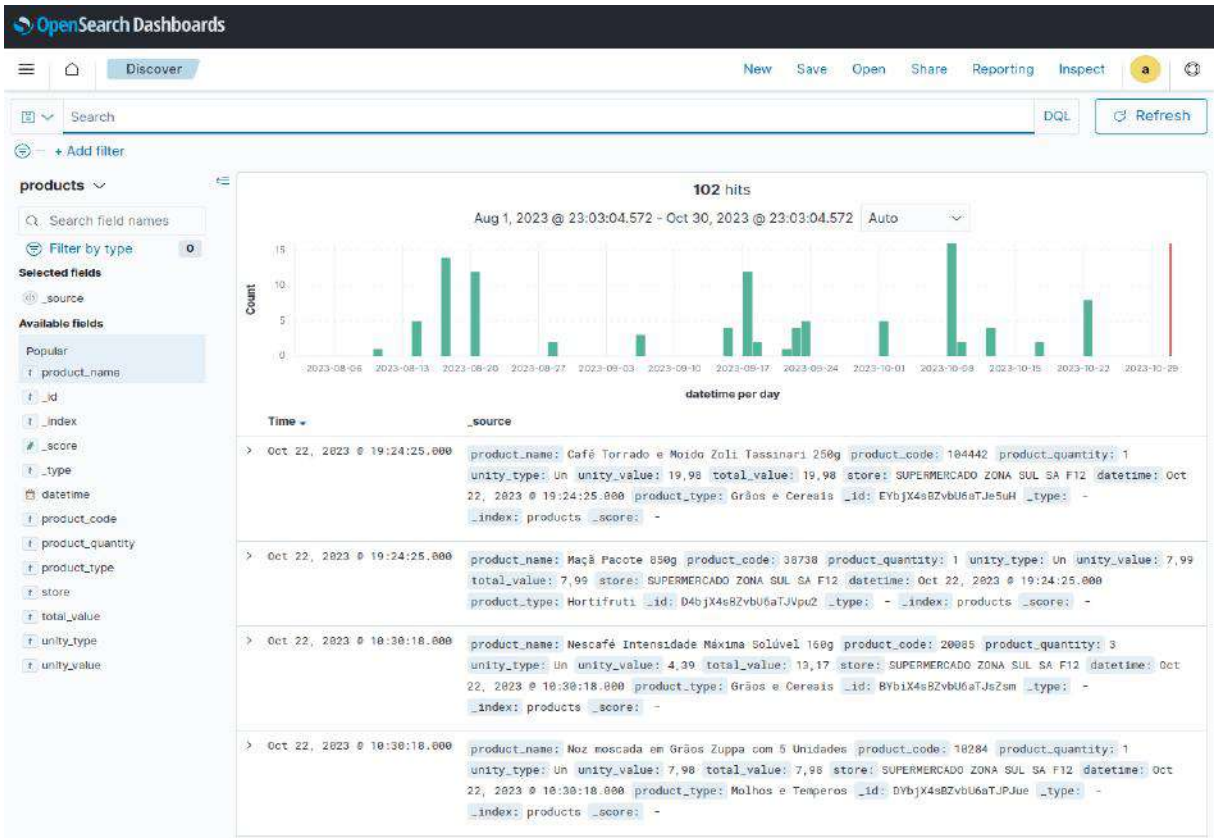


Figura 30 – Opensearch - Busca sem filtro

Utilizando a busca textual da ferramenta é possível, por exemplo, pesquisar por todas as vezes que um usuário comprou suco nos últimos 90 dias, conforme mostra a Figura 31. Dessa forma é possível expandir esse conceito para analisar a recorrência de compras de certos produtos, entendendo também aqueles que sempre aparecem nas compras.

Analisando as compras mais recorrentes vindas das notas fiscais extraídas nos testes da seção anterior, os 5 produtos mais frequentes são alface, suco natural, sorvete, morango e tangerina. Embora este exemplo tenha mostrado uma grande recorrência em alimentos, tal análise poderia mostrar outros produtos, dependendo das compras do usuário. Também é

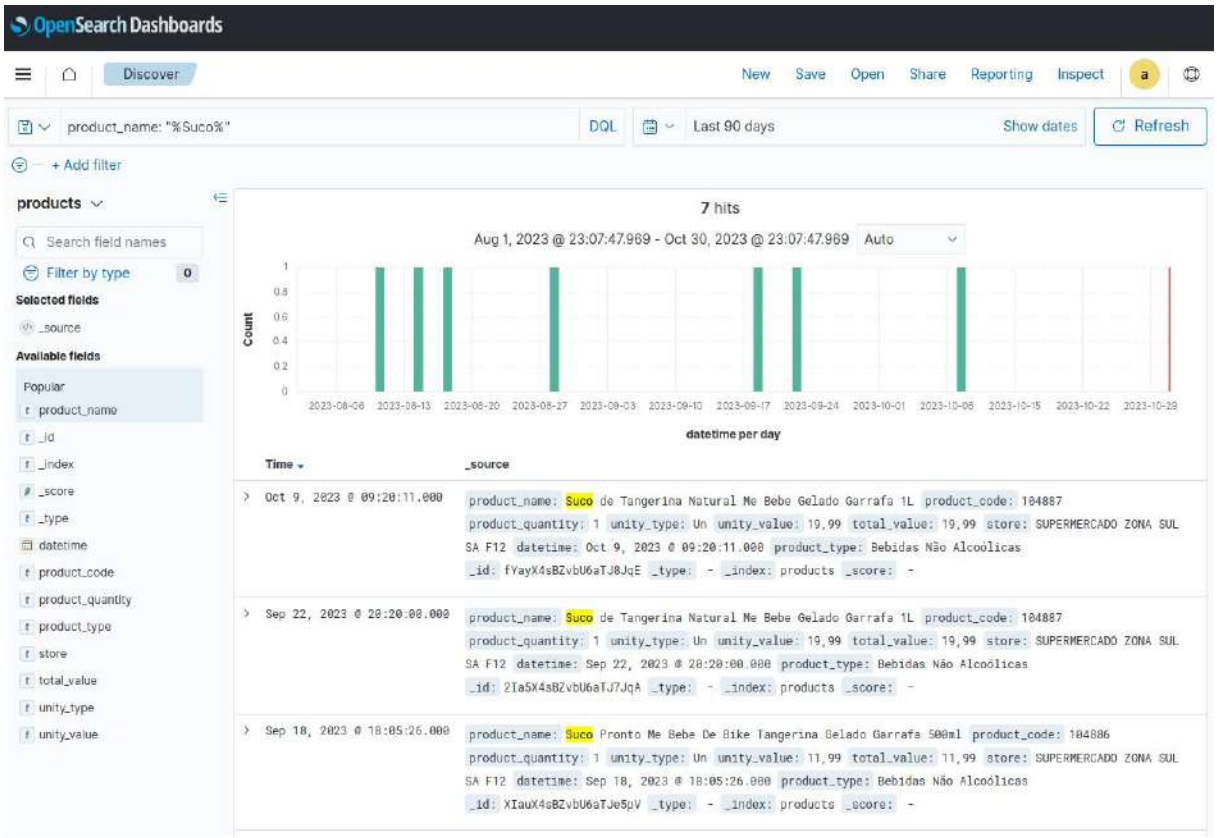


Figura 31 – Opensearch - Busca por produto

possível mostrar a recorrência de certo produto, como por exemplo a alface que os dados extraídos mostram ser uma compra semanal.

Essas funcionalidades mostradas através da interface gráfica também existem na API do Opensearch através de consultas e agregações, tornando possível que essas análises sejam fornecidas no aplicativo. Dessa forma seria possível fornecer os top 5 ou 10 produtos mais frequentes ou as lojas preferidas. Indo além, é possível fornecer uma análise da recorrência de compra de certos produtos e assim montar por exemplo uma lista de compras semanal ou mensal, ou até mesmo fornecer uma análise da evolução de preços dos produtos.

Embora neste trabalho essas funcionalidades não tenham sido implementadas para o usuário final, as mesmas se mostram como possibilidades de trabalhos futuros. Todavia, durante a construção do mesmo, a persistência dos produtos no Opensearch forneceu análises relevantes, como por exemplo produtos cujo SKU da nota fiscal ao serem buscados nos sites de mercado acabam retornando mais de um resultado (um código 123 retorna também os produtos de código 1234 e assim por diante), gerando inconsistências na extração que foram corrigidas.

5 CONCLUSÃO

A construção deste trabalho trouxe diversos desafios. De início apresentou-se o desafio básico de provar a viabilidade da sua construção, envolvendo entender o funcionamento das notas fiscais brasileiras, as informações abertas ao público e quais delas seria possível extrair automaticamente. Além disso, foi necessário entender as possíveis fontes de informações detalhadas de produtos, visto que nem todos os sites de mercado estavam preparados ou passíveis de extração.

Tendo o completo entendimento das fontes de dados envolvidas iniciou-se a preparação da arquitetura do sistema como um todo, sendo nesse estágio necessário entender as possíveis ferramentas, linguagens e plataformas a serem utilizadas, até possuir o completo entendimento e previsão do funcionamento do sistema. A partir daí se iniciou a construção do mesmo e, embora para o usuário final o sistema se apresente como um aplicativo de celular, a sua construção foi a última a ser concluída. Apenas após ter provado a possibilidade de extrair automaticamente os produtos dos sites dos mercados Zona Sul e Hortifruti, foram desenvolvidos todos os componentes utilizados nessa etapa. Isso envolveu a construção dos programas para orquestrar a extração, bem como o teste e implementação das ferramentas de fila e de persistência.

Com um conjunto robusto de aplicações implementadas para extrair dados dos produtos iniciou-se o desenvolvimento dos componentes necessários para interação com o usuário. Com isso foi construída uma API responsável por receber as requisições necessárias pelo aplicativo, construído em seguida. Este componente também serviu de base para a consolidação das entidades básicas do sistema como a Nota Fiscal e o Produto, bem como o entendimento de seu relacionamento como documentos individuais que se referenciam.

Vale também mencionar as diversas dificuldades enfrentadas e superadas durante a construção deste trabalho. Além da já mencionada dificuldade para conseguir encontrar informações concretas sobre as notas fiscais brasileiras, também foi superada a barreira técnica causada pelo uso de ferramentas consolidadas no mercado, porém desconhecidas, a exemplo do desenvolvimento de aplicativos para celular. Embora seja já uma tecnologia bastante difundida, a mesma demandou um período longo de aprendizagem, além de apresentar dificuldades próprias. A exemplo da implementação de leitores de códigos de barra e QR Code, que possuem diversas bibliotecas disponíveis, mas também diversas formas de implementação e incompatibilidades com as diferentes versões da API do Android. Além disso, instabilidades e mudanças nos sites de mercado da Fazenda tiveram de ser superados, embora caso estejam fora do ar ainda impacte severamente todo o sistema. Esses casos fizeram com que o processo de desenvolvimento fosse impactado, seja para adaptar os extratores às mudanças nos sites ou esperar que os sites voltassem ao ar.

Ao final, o trabalho se mostra funcional e útil ao dia a dia, tendo cumprido parte da sua motivação ao ajudar domesticamente na organização dos gastos cotidianos. Entretanto, algumas funcionalidades acabaram perdendo prioridade ao longo do caminho e se apresentam como possibilidade para trabalhos futuros. Por exemplo, um mecanismo de busca por produto no aplicativo, a possibilidade de extrair notas fiscais de outras lojas e mercados, uma visualização no aplicativo da recorrência da compra de produtos, fornecer alertas para quando se aproximar o momento de uma nova compra e montar uma lista de compras para o usuário. Além disso, outras visões e caminhos apresentam-se como formas de expansão do trabalho, como por exemplo a construção de um sistema de monitoramento e alerta de promoções ou quedas no preço nas lojas consumidas pelo usuário. Esta última que já é uma funcionalidade apresentada em aplicações conhecidas no mercado (como Bondfaro, Buscapé e Zoom). Todas essas funcionalidades mostram-se possíveis no momento atual em que as fundações do sistema já estão construídas e funcionando.

Por fim, este trabalho é finalizado de forma satisfatória e com aplicações práticas, assim como possivelmente serve de base para outros trabalhos acadêmicos, tendo em vista a disponibilidade do seu código abertamente na plataforma Github[1] através dos repositories dos extratores[2], aplicativo[3] e api[4]. Assim como diversos outros projetos de código livre encontrados na *internet* serviram como referência e exemplo, espera-se que este também sirva de inspiração para que outras pessoas contribuam para seu crescimento e manutenção.

[1] https://github.com/
[2] https://github.com/csmartins/receipt-extractor
[3] https://github.com/csmartins/expenses-app
[4] https://github.com/csmartins/expenses-api

REFERÊNCIAS

BLOGATIVA. **Controle de gastos: 10 aplicativos para facilitar sua gestão financeira**. 2019. Disponível em: "https://blog.ativainvestimentos.com.br/aplicativos-para-controle-de-gastos/". Acesso em: 30 aug.2022.

CANALTECH. **5 apps para controle de gastos**. 2022. Disponível em: "https://canaltech.com.br/apps/melhores-apps-orcamento-domestico-financas/". Acesso em: 30 aug.2022.

EXAME. **5 aplicativos para controlar melhor os seus gastos em 2023**. 2023. Disponível em: "https://exame.com/invest/minhas-financas/5-aplicativos-controlar-gastos-2023/". Acesso em: 18 out.2023.

FIELDING, R. T. **Representational State Transfer (REST)**. 2000. Disponível em: "https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm". Acesso em: 18 out.2023.

FINANCEONE. **Conheça 15 aplicativos para controlar gastos em 2023**. 2023. Disponível em: "https://financeone.com.br/aplicativos-para-controlar-gastos/". Acesso em: 18 out.2023.

GOV-RJ. **Manual NFC-e**. 2021. Disponível em: "http://www.fazenda.rj.gov.br/sefaz/content/conn/UCMServer/path/Contribution%20Folders/site_fazenda/informacao/sistemaseletronicos/dfe/manuais/DF-e_NFC-e.pdf?lve". Acesso em: 5 jun.2022.

GUIABOLSO. 2012. Disponível em: "https://www.guiabolso.com.br/". Acesso em: 5 jun.2022.

MOBILIS. **Os 10 melhores apps de controle financeiro de 2023**. 2023. Disponível em: "https://www.mobills.com.br/blog/aplicativos/apps-de-controle-financeiro/". Acesso em: 18 out.2023.

MOBILLS. 2013. Disponível em: "https://www.mobills.com.br/". Acesso em: 5 jun.2022.

MONEY Lover. 2011. Disponível em: "http://moneylover.me/". Acesso em: 5 jun.2022.

MORI, S.; NISHIDA, H.; YAMADA, H. **Optical character recognition**. [S.l.]: John Wiley & Sons, Inc., 1999.

MUNZERT, S. et al. **Automated data collection with R: A practical guide to web scraping and text mining**. [S.l.]: John Wiley & Sons, 2014.

REENSKAUG, T.; COPLIEN, J. O. **The DCI Architecture: A New Vision of Object-Oriented Programming**. 2009. Disponível em: "https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.html". Acesso em: 18 out.2023.

SABAB, S. A. e. a. eexpense: A smart approach to track everyday expense. **4th International Conference on Electrical Engineering and Information**, 2018. Disponível em: http://dx.doi.org/10.1109/CEEICT.2018.8628070.

SADAFULE, R. D. Mobile app development for the indian market. **IEEE Software**, 2014. Disponível em: http://dx.doi.org/10.1109/MS.2014.67.

SOCIETY, T. I. **Hypertext Transfer Protocol – HTTP/1.1**. 1999. Disponível em: "https://www.rfc-editor.org/rfc/rfc2616". Acesso em: 18 out.2023.

SPENDEE. 2018. Disponível em: "https://www.spendee.com/". Acesso em: 5 jun.2022.

TECHTUDO. **Aplicativo de finanças: veja cinco opções para gerenciar gastos pessoais**. 2021. Disponível em: "https://www.techtudo.com.br/listas/2021/04/aplicativo-de-financas-veja-cinco-opcoes-para-gerenciar-gastos.ghtml". Acesso em: 30 aug.2022.

VELMURUGAN, e. a. A. Expense manager application. **Journal of Physics: Conference Series**, v. 1712, 2020. Disponível em: https://iopscience.iop.org/article/10.1088/1742-6596/1712/1/012039.