INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Vitor Furtado de Oliveira

Nota Fiscal do Consumidor do Amazonas: Aplicativo para controle de gastos vinculados à NFC-e

Manaus - AM
2020

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Vitor Furtado de Oliveira

Nota Fiscal do Consumidor do Amazonas: Aplicativo para controle de gastos vinculados à NFC-e

"Trabalho de Conclusão de Curso apresentado à banca examinadora Curso Superior de Tecnologia de Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciências e Tecnologia do Amazonas – IFAM Campus Manaus - Centro, como requisito para o cumprimento da disciplina TCC 2 - Projeto de Software"

Orientador: Jorge Abílio Abinader Neto

Manaus - AM
2020

Vitor Furtado de Oliveira

Nota Fiscal do Consumidor do Amazonas: Aplicativo para controle de gastos vinculados à NFC-e

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia de Desenvolvimento de Sistemas do Instituto Federal de Educação Ciências e Tecnologia do Amazonas - IFAM, como requisito para obtenção do grau de Tecnológo em Análise de Desenvolvimento de Sistemas.

Aprovado em 28 de Setembro de 2020

BANCA EXAMINADORA

_____
PROF. João Guilherme de Morais Silva
INSTITUTO DE TECNOLOGIA DO AMAZONAS – IFAM

_____
ORIENTADOR: Jorge Abílio Abinader Neto
INSTITUTO DE TECNOLOGIA DO AMAZONAS – IFAM

_____
PROF. Miguel Bonafé Barbosa
INSTITUTO DE TECNOLOGIA DO AMAZONAS – IFAM

Biblioteca do IFAM - Campus Manaus Centro

O48n Oliveira, Vitor Furtado de.
Nota fiscal do consumidor do Amazonas: aplicativo para controle de gastos vinculados à NFC -e / Vitor Furtado de Oliveira. – Manaus, 2020.
64 p. : il. color.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas). – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Centro, 2020.
Orientador: Prof. Me. Jorge Abílio Abnader Neto.

1. Desenvolvimento de sistemas. 2. Banco de dados. 3. Aplicativos. I. Abnader Neto, Jorge Abílio. (Orient.) III. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. IV. Título.

CDD 005

Elaborada por Márcia Auzier CRB 11/597

AGRADECIMENTOS

Agradeço primeiramente à todos os professores do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas que durante todo o curso incentivaram e apoiaram o meu aprendizado.

Ao meu orientador, Professor Jorge Abílio, agradeço por todo o suporte, direcionamento e compreensão durante todo o processo de elaboração deste trabalho.

À minha família, em especial à minha querida mãe Maria Helena, que durante toda a vida lutou para proporcionar as melhores oportunidades para mim e para meus irmãos.

Por fim, agradeço à minha amada esposa Késsica Samara, que sempre esteve ao meu lado em todos os momentos ao longo dessa jornada, me incentivando, apoiando e principalmente me motivando à enfrentar as dificuldades para alcalçar o objetivo de conclusão do curso de graduação.

RESUMO

O controle de gastos é uma importante ferramenta para que todo consumidor possa gerenciar suas finanças e planejar suas despesas de forma responsável. Em tempos de crise financeira, a necessidade de controlar as finanças se torna ainda mais relevante para muitas famílias. Conhecendo a importância desse assunto no cenário amazonense, o presente trabalho tem como objetivo apresentar um sistema que foi desenvolvido com o intuito de disponibilizar uma ferramenta para controle de gastos, onde as compras registradas em notas fiscais (NFC-e) e vinculadas ao CPF do consumidor poderão ser consultadas de forma fácil através de um aplicativo móvel. Para isso foi necessário o desenvolvimento de dois sistemas: um aplicativo para dispositivos móvel utilizando a plataforma Android, onde o usuário irá consultar e gerenciar as informações referentes aos seus gastos e um webservice, que disponibiliza as informações para o aplicativo de acordo com as requisições enviadas. O web service desenvolvido tem o objetivo de simular a funcionalidade que poderia ser disponibilizada pelo Governo do Amazonas, que facilitaria a consulta de informações por parte dos contribuinte, além de estimular de modo sustentável a exigência de emissão de notas fiscais.

Palavras-Chave: NFC-e, Aplicativos Móveis, Controle de Gastos, Android, Web Service.

ABSTRACT

Expenses control is an important tool for every consumer to manage their finances and plan their expenses responsibly. In times of financial crisis, the need to control finances becomes even more relevant for many families. Knowing the importance of this issue in the Amazonian scenario, the present paper aims to show a system that was developed in order to provide a tool for expenses control, where purchases made in invoices (NFC-e) and linked to the consumer CPF can be consulted easily through a mobile application. For this, it was necessary to develop two systems: an application for mobile devices using the Android platform, where the user will confer and manage the information related to their expenses and a web service, which makes the information available to the application according to the executed requests. The web service developed aims to simulate the functionality that could be made available by the Amazonas Government, which facilitates the consultation of information by taxpayers, in addition to sustainably stimulating the requirement to issue invoices.

Keywords: NFC-e, Mobile Apps, Expenses Control, Android, Web Service.

SUMÁRIO

LISTA DE IMAGENS

1 INTRODUÇÃO

A alteração do sistema brasileiro de emissão de notas fiscais teve início no começo dos anos 2000, com o projeto Nota Fiscal Eletrônica (NF-e), uma parceria entre órgãos públicos, federais e empresas privadas que tinham como um dos principais objetivos a padronização do envio de notas fiscais via arquivo digital, deixando para trás a emissão da nota fiscal mercantil. (DUARTE, 2012).

Somente no ano de 2018, mais de 240 milhões de notas fiscais do consumidor foram autorizadas no estado do Amazonas. Essa grande quantidade de dados referentes aos produtos comercializados, além de servirem para a fiscalização dos impostos pagos, pode ser utilizada para as mais diversas finalidades. (ENCAT,2019).

Atualmente diversos estados incentivam o consumidor a vincular seu CPF nas NFC-e (Nota Fiscal de Consumidor Eletrônica), em troca de benefícios como descontos em impostos (IPTU e IPVA) e até a participação em sorteios com prêmios em dinheiro. (PORTAL DA FAZENDA - SP,2019).

Além dos benefícios já citados, o vínculo entre o CPF de determinada pessoa com as informações presentes nas notas fiscais gera uma grande massa de dados que pode ser aproveitada para outros fins, como será demonstrado neste trabalho.

1.1 PROBLEMATIZAÇÃO

De posse do conhecimento que existe uma massa de dados contendo o histórico de compras de grande parte da população do Amazonas, e que o acesso a estas informações não é de fácil obtenção para uma ampla parcela da sociedade, surgem dois questionamentos importantes: como seria possível simplificar o acesso a estes dados e como utilizar estas informações de uma forma que, além dos sorteios de prêmios já existentes, gere benefícios para a população ?

Como solução para esta questão,apresentamos neste trabalho a construção de uma ferramenta de assistência para a população, voltada para o controle de gastos, que utilize as informações salvas na nota fiscal do consumidor e que estejam vinculadas ao cadastro de pessoa física de determinado cidadão do Amazonas.

1.2 JUSTIFICATIVA

Segundo o serviço de proteção ao crédito (SPC), 68% dos consumidores brasileiros passaram a controlar suas finanças utilizando alguma forma de mecanismo de controle, mas somente 10% destes usam aplicativos para smartphones voltados para este fim.(SPC BRASIL,2019)

A necessidade de registrar os gastos manualmente ou a obrigatoriedade em vincular contas bancárias, pode dificultar a utilização dos aplicativos de controle financeiro já disponíveis no mercado para grande parte da população.

A proposta deste trabalho é disponibilizar um aplicativo de controle financeiro em que não seja necessário nenhum tipo de registro manual ou integração com outros aplicativos. Para isso, serão utilizadas as informações contidas nas notas fiscais do consumidor, registradas na SEFAZ do Amazonas.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Este trabalho tem como objetivo o desenvolvimento de uma aplicação para dispositivos móveis que permita que o consumidor do Amazonas possa consultar e gerenciar seus gastos a partir de suas compras vinculadas ao seu CPF.

1.3.2 OBJETIVOS ESPECÍFICOS

Para que o objetivo deste trabalho seja alcançado, foram determinados os seguintes objetivos específicos:

a) Fornecer um assistente pessoal personalizado para controle de gastos na forma de um aplicativo móvel;
b) Consultar os dados armazenados no banco de dados através de um web service.
c) Parceria com a SEFAZ Amazonas para acesso aos dados através de um web service.
d) Em caso de impossibilidade de parceria com a SEFAZ AM, desenvolver uma ferramenta para simular o funcionamento do serviço que seria disponibilizado pela entidade.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 4 capítulos: introdução (1), fundamentação teórica (2), desenvolvimento (3) e considerações finais (4).

O segundo capítulo apresenta os sistemas correlatos ao sistema proposto, além das ferramentas utilizadas para o desenvolvimento do projeto.

O terceiro capítulo apresenta o desenvolvimento das aplicaçãoes, com seus requisitos, modelagem, interface e regras de negócio.

O último capítulo apresenta as considerações finais deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os sistemas correlatos utilizados como referência de base técnica e relevância científica para aplicação proposta neste trabalho, além das tecnologias utilizadas para o desenvolvimento do software.

2.1 SISTEMAS CORRELATOS

Atualmente existem ferramentas/aplicativos que oferecem ao consumidor a possibilidade a controlar seus gastos/despesas de várias formas diferentes. Serão apresentados as duas ferramentas que consideramos mais relevantes (devido ao alcance e semelhança).

2.1.1 NOTA FISCAL PAULISTANA

Ferramenta desenvolvida pela Companhia de Processamento de Dados do estado de São Paulo, o aplicativo nota fiscal paulistana oferece ao consumidor várias funcionalidades para acompanhamento de gastos com compras e obtenção de créditos a partir da solicitação de emissão de notas fiscais.

Através da emissão de notas fiscais com o CPF vinculado, o consumidor ganha créditos, que podem ser doados para instituições de caridade ou resgatados através de transferência bancária para a conta cadastrada pelo usuário. Na figura 01 podem ser visualizadas várias doações que foram efetuadas e compras realizadas pelo usuário que estão em análise dos créditos que serão disponibilizados, de acordo com o valor total da nota fiscal.

Figura 01 - Tela de Histórico App Nota Fiscal Paulistana

Fonte: Aplicativo Nota Fiscal Paulistana

O aplicativo também permite o acompanhamento dos gastos realizados em um determinado período através da funcionalidade de gerenciamento financeiro. Após a seleção do período desejado (mês e ano), os valores de todas as notas registradas nesse período são exibidas, separadas por categorias, como pode ser visto na figura 02. As categorias são exibidas de acordo com o cadastro das lojas que emitiram as notas fiscais.

Figura 02 - Gerenciador financeiro App Nota Fiscal Paulistana

Fonte: Aplicativo Nota Fiscal Paulistana

Na figura 03 pode ser visualizada a tela de gerenciamento de créditos da nota fiscal paulistana. Os créditos são adquiridos à medida que o usuário efetue compras com notas fiscais emitidas. Com os créditos vinculados ao seu CPF, o usuário tem a opção de cadastrar uma conta bancária de sua titulariadade e efetuar a solicitação de transferência de valores.

Figura 03 - Créditos NFP App Nota Fiscal Paulistana

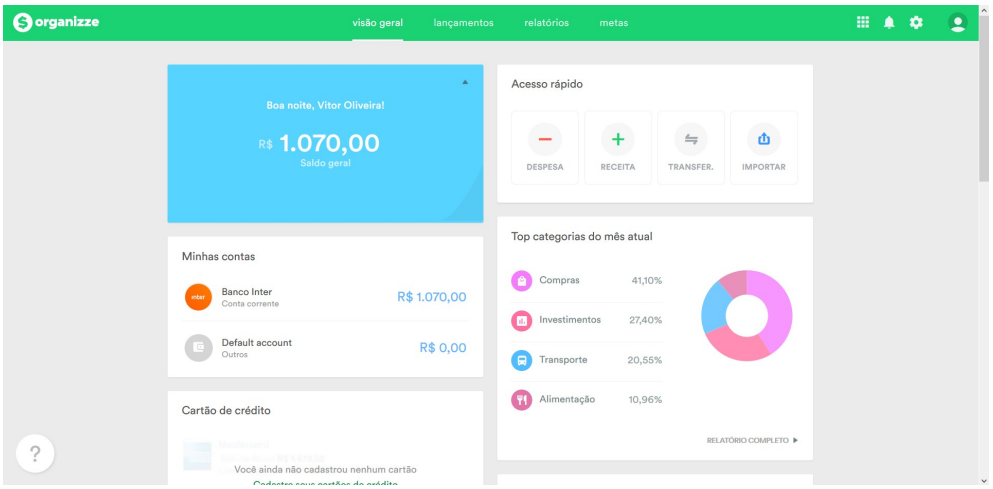
Fonte: Aplicativo Nota Fiscal Paulistana

2.1.2 ORGANIZZE

A ferramenta Organizze foi desenvolvida pela empresa Organizze Tecnologia Ltda. (Av. Hercílio Amante, 235 - sala 03 - Próspera - Criciúma, SC - Brasil - CNPJ 35.381.093/0001-26), utilizada por mais de 2 milhões de usuários e que oferece diversas funções para o gerenciamento de finanças, desde controle de gastos até os investimentos existentes. (Organizze, 2020).

A ferramenta possui diversos planos de mensalidades, desde o gratuito até o anual, diferenciando as funcionalidades disponíveis de acordo com a assinatura escolhida. Para acessar o sistema, existem duas possibilidades: utilizar a versão web, através de um browser compatível (Figura 04) ou o acesso via aplicativo, que pode ser baixado nas plataformas Android e IOS.

Figura 04 - Página do Organizze (Versão browser)

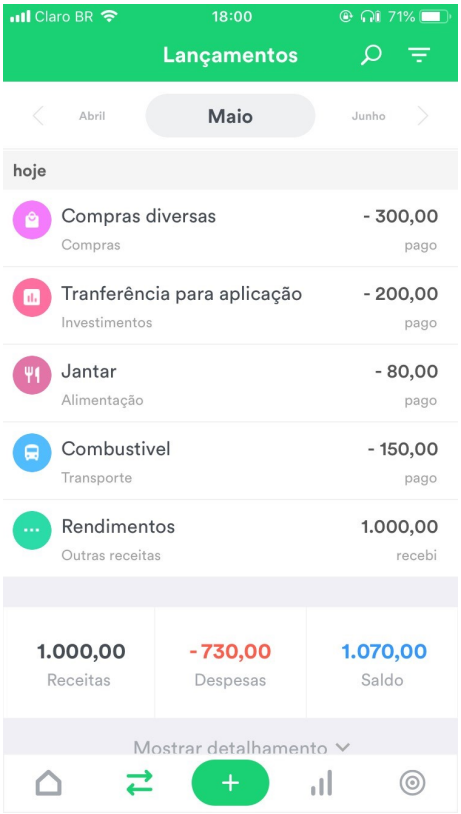


Fonte: Organizze, 2020

O sistema permite que o usuário registre o saldo de uma ou mais contas bancárias, efetuando sua atualização após os lançamentos de receitas e/ou despesas realizados. Também é possível vincular cartões de créditos, para que os lançamentos feitos sejam registrados no sistema, permitindo um acompanhamento preciso das despesas em determinado período.

Os lançamentos de despesas e/ou receitas podem ser feitos de forma automática, através do vínculo com os cartões de crédito do usuário, ou através de apontamentos manuais realizados diretamente no sistema. Os lançamentos feitos podem ser categorizados, como pode ser visto na figura 05, permitindo o gerenciamento de recursos de forma mais clara e simplificada.

Figura 05 - Consulta de lançamentos do aplicativo Organizze



Fonte: Aplicativo Organizze - Versão IOS

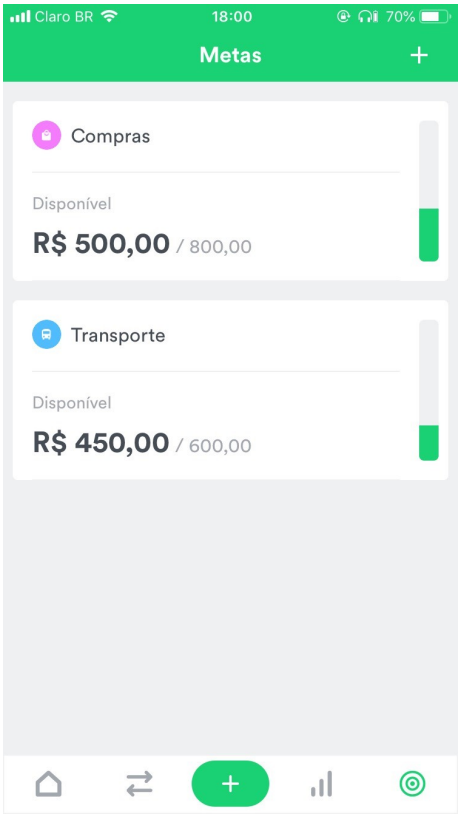
Também é possível a geração de indicadores para acompanhamento de gastos, como relatórios consolidados de receitas/despesas e gráficos com lançamentos divididos por categorias (Figura 06). A funcionalidade de geração de indicadores é uma das mais importantes do sistema, visto que através dela o usuário pode acompanhar um resumo de sua situação financeira, permitindo análise de seu perfil de consumo e planejamento para períodos futuros.

Figura 06 - Indicadores do aplicativo Organizze

Fonte: Aplicativo Organizze - Versão IOS

O sistema também possui a opção de definição de metas, onde o usuário pode determinar o máximo de gastos que deve ter em determinado período. A medida que são feitos lançamentos de despesas, o sistema irá identificar os que estejam nas categorias cadastradas nas metas e irá atualizar o valor total que ainda está disponível, como pode ser visto na figura 07. Com este controle, o usuário pode planejar seus gastos/investimentos de forma antecipada, permitindo um maior controle de suas finanças.

Figura 07 - Controle de metas do aplicativo Organizze



Fonte: Aplicativo Organizze - Versão IOS

2.2 TECNOLOGIAS UTILIZADAS

2.2.1 ANDROID

Para o desenvolvimento do aplicativo móvel foram consideradas diversas tecnologias, sendo a escolhida a plataforma Android, visto que possui uma infinidade de recursos para desenvolvimento de aplicativos, como bibliotecas e API's, além de documentação extensa disponível para consulta.

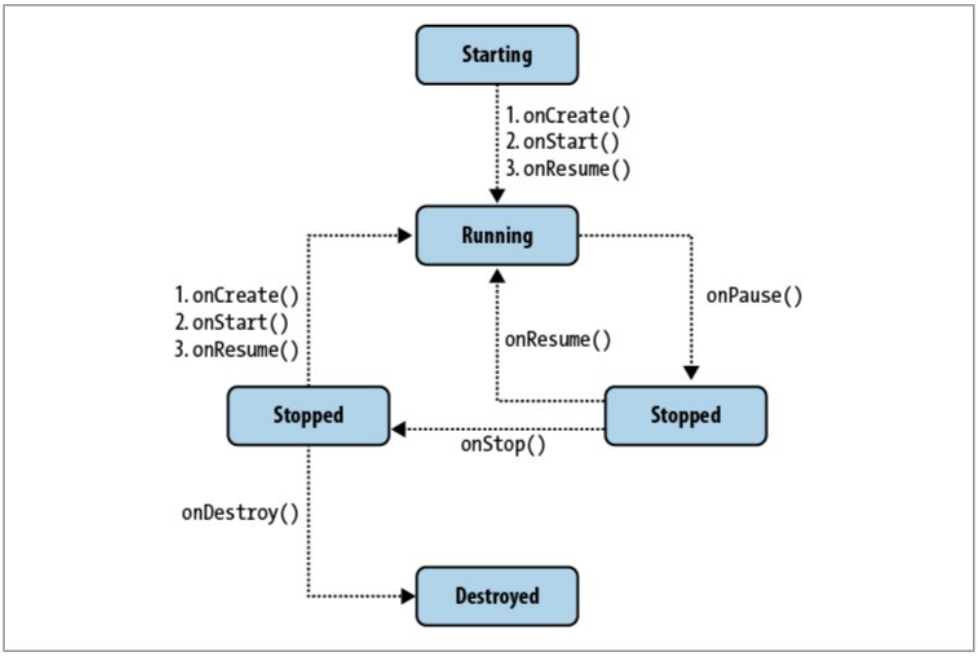
Por se tratar de uma plataforma de código-fonte aberto, voltada para dispositívos móveis, a plataforma Android tem o foco em equipamentos que possuem recursos tecnológicos limitados, sendo necessário que os aplicativos desenvolvidos sejam otimizados para rodar em dispositivos com capacidades de processamento de vários níveis.

O desenvolvimento de aplicações Android é conceitualmente diferente do desenvolvimento em outras plataformas. Aplicações em dispositivos móveis devem estar preparadas para responder à diversos estados diferentes, sejam estas alterações causadas pela própria aplicação ou por outros aplicativos sendo executados em paralelismo. (Gargenta, 2011).

Aplicativos Android tem como base as Activitys, módulos onde o usuário irá interagir com as informações e funcionalidades disponíveis. Toda Activity possui um ciclo de vida, desde sua criação até sua destruição, passando por diversos estados ao longo do tempo. Todo esse processo é controlado pelo gerenciador de Activitys, que é reponsável por criar, destruir e gerir a Activity durante seu ciclo de vida.

A figura 08 mostra um esquema com os estados de uma Activity durante seu ciclo de vida. Ao iniciar um aplicativo, a Activity principal tem seus recursos criados e exibidos para o usuário. Caso o usuário inicie um novo aplicativo ou receba uma ligação, a Activity em foco irá para o estado em pausa, sendo deixada em plano de fundo enquanto o foco não for retomado. Caso queira, o usuário pode retornar ao aplicativo e continuar sua utilização de forma rápida, visto que a Activity foi armazenada em memória.

Figura 08 - Ciclo de vida da Activity



Fonte: GARGENTA, 2011

Um dos pontos principais em aplicações Android atuais é a utilização de web services, que permitem que o aplicativo utilize recursos disponíveis através da internet, apresentando os seguintes benefícios: o processamento e armazenamento de informações são realizados pelo web service, reduzindo o uso de recursos do dispositivo, tornando o aplicativo mas leve e eficiente; Permite também a utilização de recursos devenvolvidos por terceiros, que podem ser agregrados na aplicação, como o uso de web services para consulta de mapas ou até validação de CEP.

O Android possui recursos nativos para a utilização de web services, sendo as classes HttpURLConnection e AsyncTask as principais utilizadas para realizar a requisição e processamento.

A classe HttpURLConnection é responsável por efetuar a chamada para o web service, através de uma URL e parâmetros, caso estes sejam necessários. No exemplo da figura 09, é feita uma solicitação para a URL definida na linha 02, utilizando o objeto da classe HttpURLConnection para recuperar a página web www.android.com.

Figura 09 - Exemplo de uso da classe HttpURLConnection

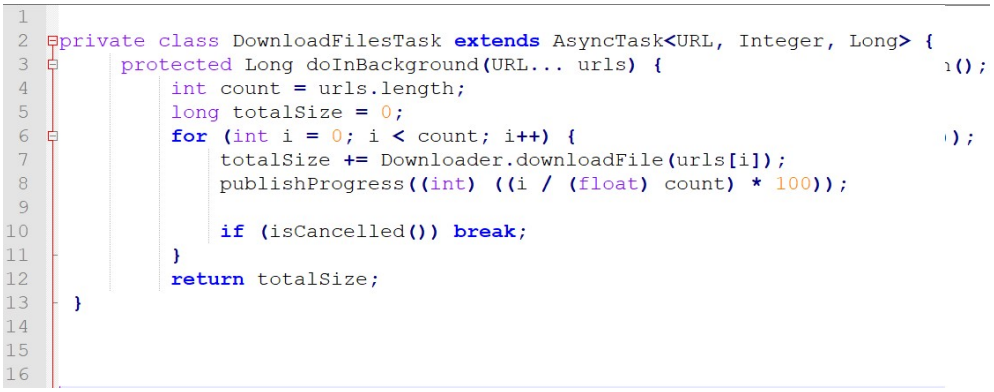
```
1
2  URL url = new URL("http://www.android.com/");
3  HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
4
5  try {
6   InputStream in = new BufferedInputStream(urlConnection.getInputStream());
7   readStream(in);
8  } finally {
9   urlConnection.disconnect();
10 }
11
```

Fonte: PORTAL ANDROID DEVELOPERS, 2020

Quando é feita uma requisição para web service, é considerada uma boa prática a utilização de threads separadas para o processamento da solicitação. Isso é feito para que a aplicação não fique travada no aguardo da resposta do serviço, permitindo que o usuário utilize o dispositivo para outras atividades. Nativamente o Android é um sistema multitarefa, permitindo que utilizemos seus recursos para desenvolver aplicações com essa funcionalidade. No exemplo da figura 10, podemos ver a utiização da classe AsyncTask para extender classes que terão seus métodos executados de forma assíncrona

(linha 02). O método doInBackgroud, utilizado para baixar arquivos de acordo com as URL's informadas, percorre uma estrutura de repetição (linha 06) para baixar os arquivos selecionados. Graças a esse processo ser executado de forma assíncrona, o sistema fará a execução do método e não ficará preso no aguardo do retorno, visto que será processado em uma thread diferente da principal.

Figura 10 - Exemplo de uso da classe AsyncTask



```
1
2  private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
3      protected Long doInBackground(URL... urls) {
4          int count = urls.length;
5          long totalSize = 0;
6          for (int i = 0; i < count; i++) {
7              totalSize += Downloader.downloadFile(urls[i]);
8              publishProgress((int) ((i / (float) count) * 100));
9
10             if (isCancelled()) break;
11         }
12         return totalSize;
13 }
14
15
16
```

Fonte: PORTAL ANDROID DEVELOPERS, 2020

Utilizar os recursos descritos anteriormente é possível durante o processo de desenvolvimento de uma aplicação, mas não é a forma mais indicada, visto que existem diversas APIs que permitem a comunicação com web services de forma mais prática.

É o caso da API RETROFIT, desenvolvida pela empresa SQUARE, que consiste em um client HTTP para Java/Android que abstrai muitas funções necessárias para a comunicação entre aplicações e web services, além de otimizar alguns procedimentos no sistema operacional. Abaixo seguem exemplos de como a API é utilizada em um projeto Android.

Na figura 11, a interface GitHubService é criada para mapear os comandos HTTP que serão utilizados (linha 03) com os métodos definidos, sendo possível visualizar uma parte da URL onde será acessado o serviço Também são definidos os parâmetros que serão utilizados, no caso o User tipo String.

Figura 11 - Exemplo de uso da interface GitHubService

```
1
2  public interface GitHubService {
3      @GET("users/{user}/repos")
4      Call<List<Repo>> listRepos(@Path("user") String user);
5  }
6
```

Fonte: PORTAL RETROFIT, 2020

Na figura 12, durante a declaração a classe Retrofit é definida a URL base onde serão feitas as requisições para o web service (linha 03). A API completa a URL de acordo com o método que será utilizado, conforme definido na interface mostrada anteriormente. Durante a chamada dos métodos, a API efetua a requisição HTTP de forma assíncrona por padrão, abstraindo o gerenciamento das threads, sem que seja necessária a utilização da classe AsyncTask.

Figura 12 - Exemplo de declaração da classe Retrofit

```
1
2  Retrofit retrofit = new Retrofit.Builder()
3      .baseUrl("https://api.github.com/")
4      .build();
5
6  GitHubService service = retrofit.create(GitHubService.class);
7
```

Fonte: PORTAL RETROFIT, 2020

Além da praticidade demostrada anteriormente, o Retrofit possui performance superior quando comparada com outras APIs disponíveis. Uma comparação de performance feita entre o Retrofit e outras três APIs, demonstra o tempo de resposta de requisições feitas em cada delas. Foram realizados vários testes de envio de requisições únicas e foi medido o tempo de resposta para cada um dos clients (BHADE; YADAV, 2019). Em todas as simulações, o tempo de resposta encontrado no uso da API Retrofit foi inferior aos concorrentes, como pode ser visualizado na figura 13.

Figura 13 - Comparativo de tempo de resposta entre clients HTTP

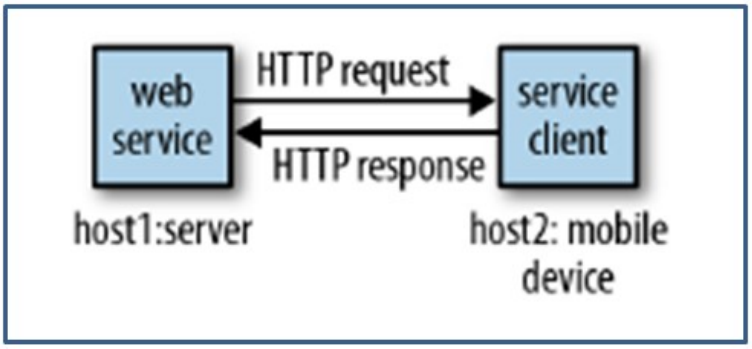
Retrofit	Volley	Http Client	Fast Network Library
22	64	69	44
30	68	65	37
28	65	67	36
28	45	73	34
25	51	62	34

Fonte: BHADE; YADAV, 2019

2.2.2 RESTFUL WEB SERVICE

O termo web service pode ter vários significados, mas pode ser definido como uma aplicação web, que consiste em um serviço e um cliente, ou também conhecidos como consumidor e requisitante. São aplicações que disponibilizam recursos através de HTTP (HyperText Transport Procotol) e podem ser desenvolvidas em diversos tipos de linguagem de programação. (KALIN, 2013). A figura 14 representa a comunicação entre um web service e um service client, através de respostas e requisições HTTP.

Figura 14 - Representação web service x service client

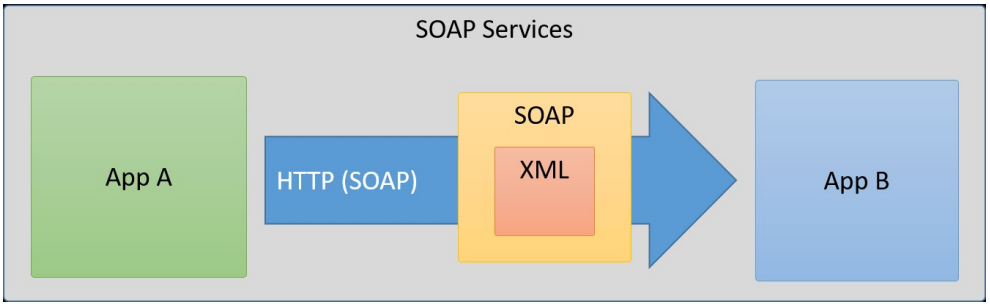


Fonte: KALIN, 2013

Existem dois padrões mais utilizados para efetuar a comunicação com Web services: o protocolo SOAP e o modelo REST.

SOAP, Simple Object Acess Protocol, ou Protocolo Simples de Acesso a Objetos em português, é um protocolo de comunicação baseado em XML que permite a comunicação de mensagens entre aplicações via HTTP. (ZARELLI, 2012).

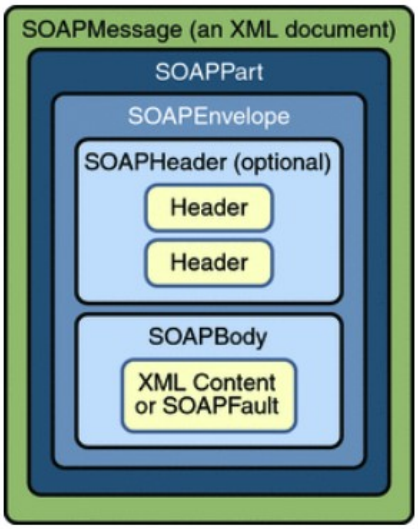
Figura 15 - Esquema web service com protocolo SOAP



Fonte: MEDINA, 2020

Uma mensagem SOAP é um documento XML que possui como estruturas principais: envelope, header e o body. O envelope é o elemento raiz do documento XML. Ele pode conter declarações de namespaces e também atributos adicionais como o que define o estilo de codificação. O estlo de codificação define como os dados são representados no documento XML. O header é um cabeçalho opcional. Ele carrega informações adicionais, como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário. Quando utilizado, o Header deve ser o primeiro elemento do Envelope. Por fim, o elemento body, que é obrigatório e contém o payload, ou a informação a ser transportada para o seu destino final. Ele pode conter um elemento opcional chamado Fault, usado para carregar mensagens de status e erros retornadas pelos "nós" ao processarem a mensagem. (DANTAS, 2020).

Figura 16 - Estrutura de uma mensagem SOAP



Fonte: ZARELLI, 2012

O modelo REST, Representational State Transfer, ou transferência de estado representativo em português, é um modelo de desenvolvimento de web services, que é guiado pelas boas práticas de uso de HTTP: Uso adequado de métodos HTTP, uso adequado de URL's, Uso de codigos de status padronizados para representacao de sucessos ou falhas, uso adequado de cabecalhos HTTP e interligacões entre vários recursos diferentes. (SAUDATE, 2014).

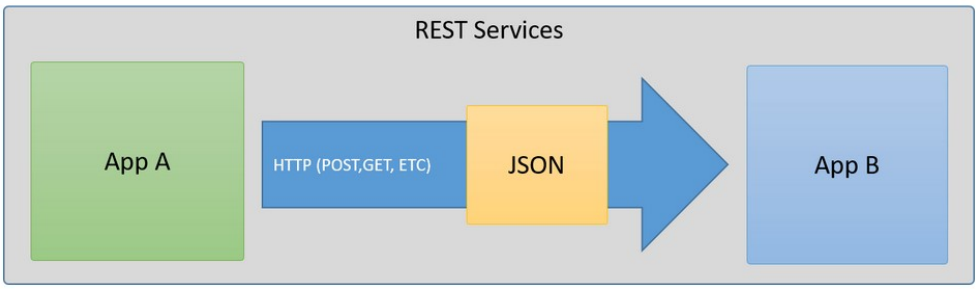
No modelo REST tudo é definido em termos de recursos, sendo estes conjuntos de dados que são trafegados pelo protocolo. Os recursos são representados por URI's, Universal Resource Identifiers, que são utilizados para identificar os recursos disponibilizados pelo web service. (SAUDATE, 2014).

Utilizando o exemplo de URI (http://localhost:8080/nfe/produtos), podem ser identificados os seguintes elementos:

1) http:// - Indicado o protocolo que está sendo utilizado;
2) Localhost:8080 - Indica o servidor da rede que está sendo utilizado;
3) Nfe - Indica o contexto da aplicação, ou seja, a raiz pela qual a aplicação está sendo fornecida para o cliente.
4) Produtos - Indica o endereço do recurso, para onde serão solicitadas as requisições.

Para interagir com as URI's, os métodos HTTP são utilizados, sendo identificados pelos verbos GET (Utilizado para a recuperação de dados identificados pela URI), POST (Utilizado para inclusão de um novo recurso), PUT (Utilizado para a atualização de um recurso) e DELETE (Utlizado para a exclusão de um recurso). (SAUDATE, 2014).

Figura 17 - Esquema web service padrão REST



Fonte: MEDINA, 2020

Em comparação, o SOAP e o REST possuem várias características únicas, com suas próprias vantagens e desvantagens, e seus usos sendo indicados para cenários diferentes, conforme pode ser visto na figura 18.

Figura 18 - Comparação SOAP x REST

	SOAP	**REST**
Segurança	WS-Security com suporte SSL.	Suporta HTTPS e SSL.
Rendimento	Necessidade de mais largura de banda	Não requer recursos robustos
Formato de Mensagens	Apenas XML	HTML, XML, JSON, entre outros
Protocolos de Transferência	HTTP, SMTP, UDP, entre outros	Apenas HTTP
Recomendado para	Aplicações empresariais, aplicações com nível de segurança elevado, serviços financeiros, comunicações.	APIs públicas para serviços web, serviços móveis e redes sociais.
Vantagens	Alta segurança, padronizado.	Escalabilidade, melhor rendimento, fácil navegação, flexibilidade
Desvantagens	Menor rendimento, maior complexidade, menor flexibilidade.	Menor segurança

Fonte: MEDINA, 2020

Para este projeto foi escolhido o modelo REST, visto que é o mais indicado para aplicações móveis por permitir a transmissão de dados através de conexões limitadas, por utilizar o formato JSON e por possuir um melhor rendimento que seu concorrente. Apesar de ser possível a utilização do protocolo SOAP em web services, a necessidade de uso de XML e a maior complexidade de implementação tornou seu uso inviável durante o desenvolvimento do sistema.

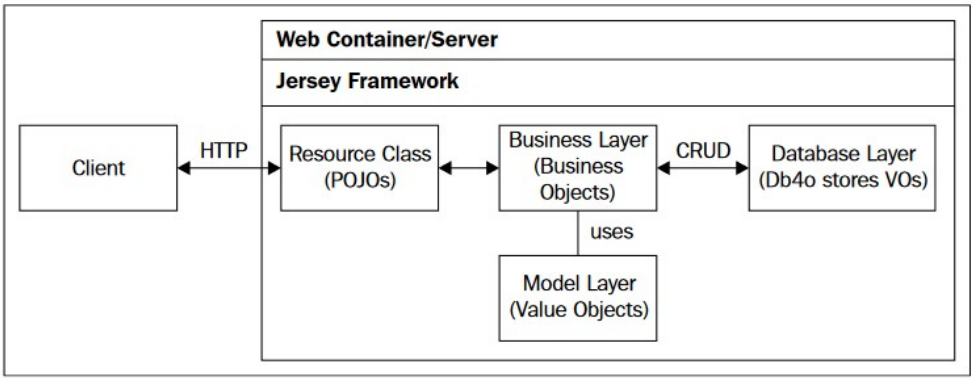
Para o desenvolvimento do web service, foi escolhido o framework JERSEY, que implementa todos os requerimentos da JAX-RS, API para desenvolvimento de web services RESTFUL em JAVA, sendo seu principal objetivo tornar mais prático o desenvolvimento de web services REST. (SANDOVAL, 2009).

Um recurso JERSEY nada mais é que uma classe JAVA no contexto de uma aplicação WEB, que usa anotações para mapear seus métodos, definir as URI's onde os recursos estarão disponíveis e quais os formatos serão recebidos e devolvidos pelo web service. Algumas anotações utilizadas:

1) @Path: Utilizada para determinar onde o recurso estará disponível. Exemplo: @Path("/users");
2) @PathParam: Usado para definir os parâmetros que serão recebidos. Exemplo: @Path("/users/{username}")
3) @Consumes: Define o tipo de dado que será recebido pelo recurso. Exemplo: @Consumes("application/xml")
4) @Produces: Define o tipo de dado que será devolvido pelo recurso. Exemplo: @Produces("application/json")

Na figura 19 pode ser visualizado o esquema de um web service JERSEY, dividido em 3 camadas. A primeira contém as classes com os recursos do JERSEY, que gerenciam a comunicação entre o client e o server, através das requisições HTTP. Na segunda estão as classes de négocios, que gerenciam o armazenamento e recuperação dos recursos, e as classes do modelo, que contém as classes armazenadas no banco de dados. Por último,na última camada, está o banco de dados com as informações armazenadas. (SANDOVAL, 2009).

Figura 19 - Esquema web service com JERSEY



Fonte: SANDOVAL, 2009

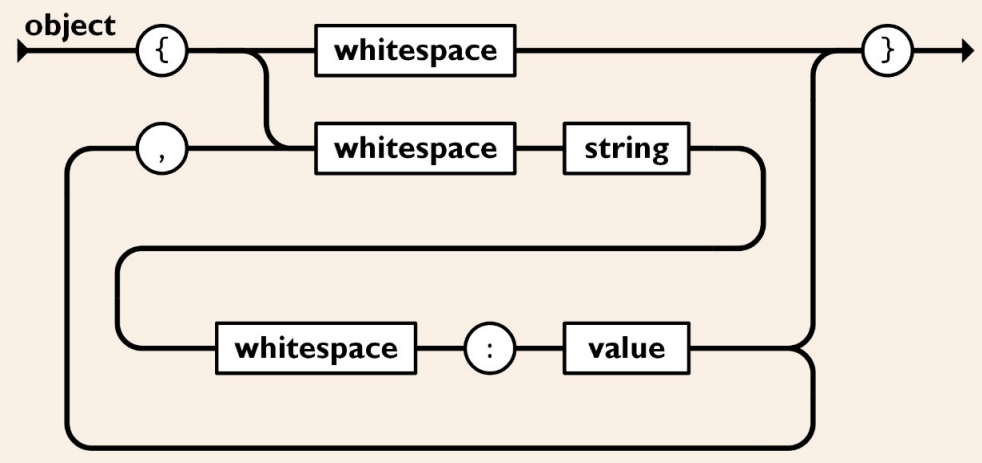
2.2.2 JSON

O JSON (pronunciado como a palavra inglês (Jason) é um acrônimo para JavaScript Object Notation, que consiste em um formato de texto para a serialização de dados estruturados. O JSON pode representar quatro tipos primários de dados: Strings, números, booleanos e nulos, além de dois tipos estruturados (objetos e vetores). (FONSECA; SIMÕES, 2007).

JSON é um formato de texto completamente independente de linguagem, porém utiliza convenções familiares de diversas linguagem de programação como C, C++, C#, Java, JavaScript, Perl, Python, e muitas outras. Isso torna o JSON a linguagem ideal para troca de informações entre sistemas (FRIESEN, 2016).

A figura 20 demonstra o esquema de um objeto JSON, que consiste em um conjunto desordenado de pares de nomes e valores. Um objeto sempre inicia e termina com chaves (“{“ e “}”). Cada nome é seguido por seu valor, separados por dois ponto “:”.

Figura 20 - Esquema de um objeto JSON



FONTE: JSON, 2020

Estruturas JSON podem ser utilizadas para mapear informações de objetos JAVA, como pode ser visto na figura 21. Neste exemplo a estrutura do JSON reflete a classe Produto, com os atributos e valores preenchidos de acordo com a requisição efetuada para o web service. Na linha 06 está definido o atributo nfecapa, que por sua vez é uma classe com as informações da NFe em que o produto está relacionado. Entre as linhas 07 e 12 podem ser vistos os atributos da classe do NfeCapa, com seus valores preenchidos. Esse exemplo demonstra a capacidade de mapeamento de estruturas JSON, podendo estruturar o relacionamento de diversas classes de forma simples e organizada.

Figura 21 - Exemplo de JSON

```
1  {
2    {
3    "codigo": "13",
4    "descricao": "QUEIJO ASSADO",
5    "iddetalhe": 12,
6    "nfecapa":       {
7       "cnpj": "13.742.161/0001-00",
8       "cpf": "93225733253",
9       "id": 3,
10      "loja":          {
11         "endereco": "Avenida Tancredo Neves, 920 A, , PARQUE 10 DE NOVEMBRO, Manaus, AM",
12         "razaosocial": "Adolpho - Assador Oficial"
13      },
14      "valortotal": 72
15   },
16   "qtd": 1,
17   "valor": 14
18 }
```

Fonte: Autoria própria

O JSON apresenta diversas similiaridades com seu principal “concorrente”, o XML (Extensible MarkUp Language). Ambos modelos representam informações no formato texto, são independentes de linguagem, possuem natureza auto-descritiva, possuem a capacidade de representar informações complexas, entre outras características em comum. Mesmo possuindo algumas características semelhantes ao XML, o JSON apresenta algumas particularidades como: não possuir marcações disponíveis (TAGS), não permitir instruções de processamento e somente permitir a troca de informações. (GONÇALVES, 2012).

Além das diferenças listadas anteriormente, o JSON e o XML possuem uma distinção em um aspecto muito importante: a performance. Em um artigo publicado na Universidade Estadual de Montana, (NURSEITOV *et al.*, 2009), Estados Unidos, foram realizados testes de performance entre as duas estruturas. Em um teste específico, foram realizados diversos envios de objetos, com as mesmas quantidades, e foram capturados os tempos de resposta. Como pode ser visto na figura 22, os tempos de resposta do JSON foram consideravelmente menores que os do XML.

Figura 22 - Comparação de performance entre JSON x XML

	JSON	XML
Trial 1 Number Of Objects	20000	20000
Trial 1 Total Time (ms)	2213.15	61333.68
Trial 1 Average Time (ms)	0.11	3.07
Trial 2 Number Of Objects	40000	40000
Trial 2 Total Time (ms)	3127.99	123854.59
Trial 2 Average Time (ms)	0.08	3.10
Trial 3 Number Of Objects	60000	60000
Trial 3 Total Time (ms)	4552.38	185936.27
Trial 3 Average Time (ms)	0.08	3.10
Trial 4 Number Of Objects	80000	80000
Trial 4 Total Time (ms)	6006.72	247639.81
Trial 4 Average Time (ms)	0.08	3.10
Trial 5 Number Of Objects	100000	100000
Trial 5 Total Time (ms)	7497.36	310017.47
Trial 5 Average Time (ms)	0.07	3.10

Fonte: NURSEITOV *et al.*, 2009

3 DESENVOLVIMENTO

Neste capítulo serão apresentados os processos de desenvolvimento da aplicação mobile (Aplicativo Meus Gastos) e o do web service desenvolvido para simular o serviço de informações da SEFAZ AM.

3.1 DESCRIÇÃO DO SISTEMA

O projeto será dividido em 2 módulos: Módulo mobile (Aplicativo Meus Gastos) que será utilizado pelos usuários e um web service que fornecerá as informações para o aplicativo. O módulo mobile permitirá que o usuário acesse através de seu smartphone Android diversas funcionalidades, tais como: permitir o cadastro do usuário no próprio aplicativo, cadastro de categorias de gastos (medicamentos, supermercados, alimentação, gasolina etc), vincular categorias com tipos de compras, consultar total de gastos por categorias, consultar o histórico de preços de determinados produtos, total de gastos por período, entre outras funcionalidades. O aplicativo contará com um banco de dados local, para que algumas informações básicas sejam salvas no dispositivo, sendo o restante das informações disponibilizadas pelo web service.

O aplicativo será leve e com uso intuitivo, facilitando sua utilização por qualquer tipo de usuário. Com a utilização de dados já registradas nas notas fiscais emitidas para o consumidor, não será necessário cadastrar informações manualmente, ou relacionar os gastos realizados através de vínculos com cartões de crédito, o que simplificará o controle de gastos do usuário.

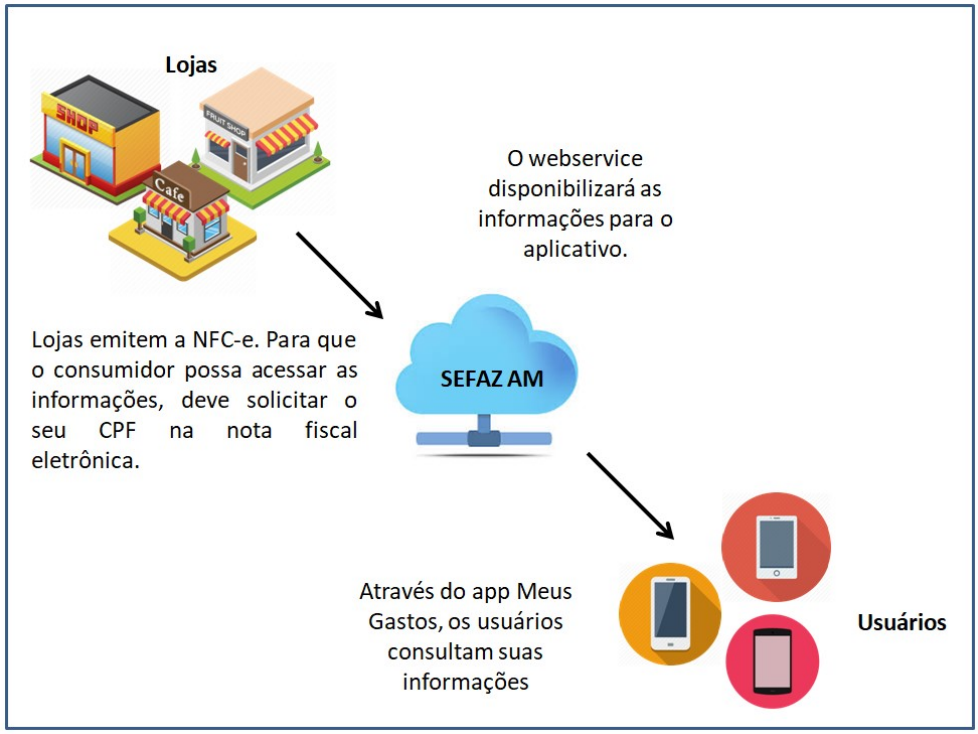
A segunda parte do sistema será um web service desenvolvido para simular um serviço da SEFAZ AM, disponibilizando as informações de acordo com as requisições do usuário. As informações serão disponibilizadas ao aplicativo através do formato JSON (JavaScript Object Notation), tornando a comunicação leve e rápida.

O banco de dados utilizado pelo web service terá as informações de todas as compras realizadas pelo usuário, desde que o mesmo vincule o CPF no momento da emissão das notas fiscais. Serão capturadas as seguintes

informações das notas fiscais: estabelecimento onde foi efetuada a compra, nome do produto, quantidade adquirida, valor do produto e data da compra.

A figura 23 apresenta a estrutura do projeto, onde o web service consultará as informações no banco de dados e disponibilizará os dados através do formato JSON para o aplicativo, que por sua vez exibirá as informações para o usuário, de acordo com a funcionalidade escolhida. O aplicativo também terá acesso às informações salvas no banco de dados do dispositivo.

Figura 23 - Esquema do sistema



Fonte: Autoria própria

3.2 ANÁLISE DO SISTEMA

Neste tópico serão apresentados os principais documentos referentes ao processo de análise do projeto, onde foram definidas todas as características e funcionalidades do sistema. Foram gerados diversos artefatos com o objetivo de determinar o comportamento do aplicativo e do web service, tais como requisitos funcionais e diagrama de caso de uso.

3.2.1 REQUISITOS FUNCIONAIS

Na seção seguinte foram descritos os requisitos funcionais que nortearam o desenvolvimento do aplicativo Meus Gastos. Para facilitar o entendimento dos requisitos, os mesmo foram categorizados por módulos e níveis de prioridade.

Em relação aos módulos, foram definidos 3: acesso, cadastros e consultas. O módulo de acesso agrega as funcionalidades relacionadas ao controle de acesso no aplicativo: cadastro de usuário e login do usuário. O módulo de cadastros se refere às funcionalidades de manutenção dos cadastros realizados no aplicativo, como cadastro e exclusão de categorias, vinculação de gastos por categorias etc. O módulo de consultas engloba as funcionalidades referentes às consultas do sistema, como consulta de gastos por categorias, lojas e produtos.

Em relação aos níveis de prioridade, foram definidos 3: crítica, média e baixa. Os requisitos classificados com prioridade crítica são referentes às funcionalidades bases do sistema, servindo como o esqueleto funcional do projeto, sem as quais o mesmo não funcionará. Requisitos com prioridade média são referentes às funcionalidades importantes para o sistema, mas que não caracterizadas como essenciais. Por último, os requisitos classificados com prioridade baixa, são relacionados à funcionalidades opcionais do sistema, também podendo ser definidos como extras do sistema.

Abaixo estão descritos os requisitos funcionais elaborados para o sistema:

Identificador	RF001		
Nome	Efetuar acesso no aplicativo móvel		
Módulo	Acesso		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	O usuário deverá informar o CPF e senha cadastrados anteriormente para ter acesso ao sistema. As informações serão validadas através do web service. Caso as informações estejam incorretas, a seguinte mensagem será mostrada: "CPF e/ou senha informados são inválidos. Verifique as informações e tente novamente". Caso as informações estejam corretas, o usuário será direcionado para a tela principal do aplicativo.		

Identificador	RF002		
Nome	Cadastrar as informações do usuário		
Módulo	Acesso		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	Caso o usuário não possuir acesso ao sistema, o mesmo poderá cadastrar suas informações diretamente no aplicativo. O usuário irá informar CPF, nome completo, email e senha. O sistema validará o CPF informado. Caso todas as informações sejam válidas, o cadastro é realizado e o usuário é redirecionado para a tela de login do aplicativo.		

Identificador	RF003		
Nome	Cadastrar categorias de gastos		
Módulo	Cadastros		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	O usuário poderá cadastrar tipos de categorias para separar seus gastos de acordo com suas necessidade. Na tela de cadastro de categorias, o usuário deverá informar um nome para a categoria que deseja cadastrar. Caso o cadastro seja efetuada com sucesso, o aplicativo exibirá a mensagem "Categoria cadastrada com sucesso."		

Identificador	RF004		
Nome	Deletar categorias de gastos		
Módulo	Cadastros		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Média
Descrição	O usuário poderá excluir categorias previamente cadastradas. Na tela de categorias, o usuário deverá selecionar a categoria desejada e clicar em excluir. Caso a categoria possua algum registro vinculado, a seguinte mensagem de erro deverá ser exibida: "Exclusão cancelada. Categoria possui gastos vinculados e não pode ser excluída.". Caso a categoria não possua registros vinculados, a mesma será excluída e a seguinte mensagem será exibida: "Categoria excluída com sucesso.".		

Identificador	RF005		
Nome	Relacionar lojas com categorias de gastos		
Módulo	Cadastros		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	O usuário poderá vincular registros para categorizar seus gastos. Na tela de consulta de gastos por lojas, o usuário poderá selecionar um registro e selecionar a opção "Vincular categoria". Ele será redirecionado para uma outra tela, onde selecionará qual categoria, previamente cadastrada, deve ser relacionada à loja. Após clicar no botão "Salvar", e a seguinte mensagem será exibida: "Vínculo registrado com sucesso.".		

Identificador	RF006		
Nome	Remover o vínculo entre loja e categoria de gastos		
Módulo	Cadastros		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Média
Descrição	O usuário poderá efetuar a exclusão de vínculos entre lojas e categorias. O usuário irá selecionar um registro na tela de cadastro de categorias, e selecionar "Detalhes". Ele será redirecionado para uma tela, onde todas as lojas vinculadas à categoria serão listadas. Ele selecionará a loja escolhida e clicar na opção "Remover". Após a exclusão do registro, a seguinte mensagem será mostrada "Registro excluído com sucesso.".		

Identificador	RF007		
Nome	Consultar gastos realizados em lojas específicas		
Módulo	Consultas		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	O usuário poderá consultar gastos realizados por lojas. Ele poderá utilizar filtros para efetuar a consulta, como nome da loja e/ou data quando foram efetuadas as compras. Após informar os filtros desejados, irá clicar em "Filtrar". O aplicativo irá enviar a requisição para o web service, que retornará as informações, que serão exibidas para o usuário em forma de lista.		

Identificador	RF008		
Nome	Consultar gastos por produtos		
Módulo	Consultas		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	O usuário poderá consultar gastos por produtos. Ele poderá utilizar filtros para efetuar a consulta, como nome do produto e/ou data quando foram efetuadas as compras. Após informar os filtros desejados, irá clicar em "Filtrar". O aplicativo irá enviar a requisição para o web service, que retornará as informações, que serão exibidas para o usuário em forma de lista.		

Identificador	RF009		
Nome	Consultar gastos por categorias		
Módulo	Consultas		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Crítico
Descrição	O usuário poderá consultar gastos por categorias, de acordo com as categorias previamente criadas. Ele poderá utilizar filtros de data da compra para restringir os dados consultados. Após informar os filtros desejados, irá clicar em "Filtrar". O aplicativo irá enviar a requisição para o web service, que retornará as informações, que serão exibidas para o usuário em forma de gráfico.		

Identificador	RF010		
Nome	Consultar detalhes de compra por loja		
Módulo	Consultas		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Baixa
Descrição	O usuário poderá consultar os detalhes de uma compra por loja. Na tela de consulta de compras por lojas, o usuário poderá selecionar o registro desejado e selecionar a opção "Detalhes". Ele será redirecionado para uma nova tela, onde as informações detalhadas da compra serão mostradas (Nome da loja, CNPJ, valor total, data da compra e endereço do estabelecimento).		

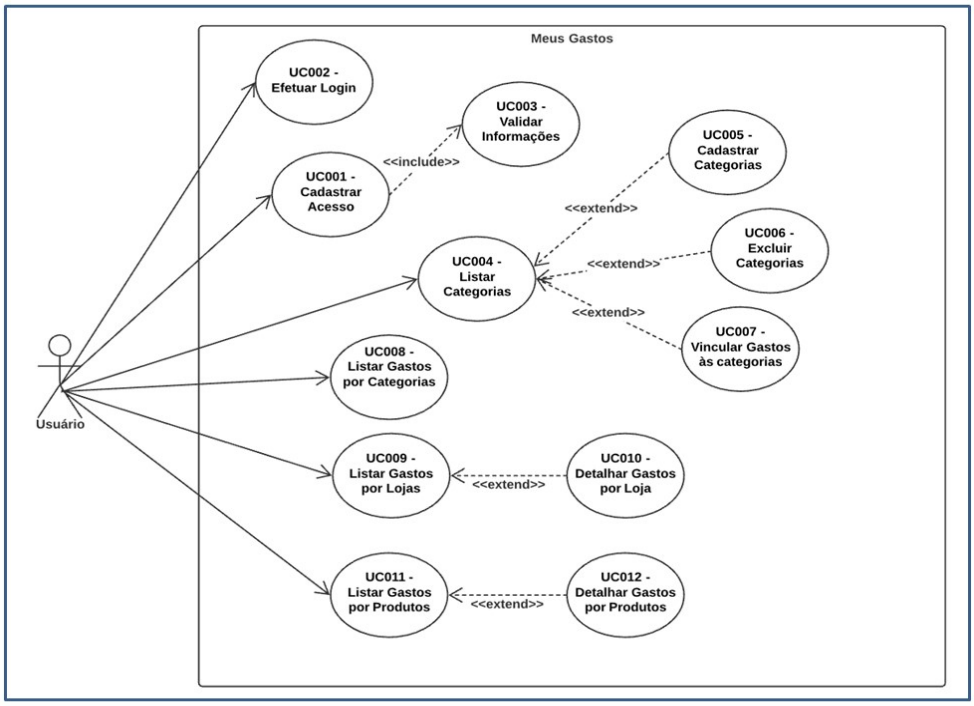
Identificador	RF011		
Nome	Consultar a localização da loja		
Módulo	Consultas		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Baixa
Descrição	O usuário poderá visualizar a localização da loja onde efetuou uma compra. Para isso, na tela de detalhes de compra por loja, ele irá clicar no botão "Ver no mapa". O usuário será redirecionado para o aplicativo Google Maps, que mostrará a localização selecionada.		

Identificador	RF012		
Nome	Consultar detalhes de compra por produto		
Módulo	Consultas		
Data de criação	01/02/2020	**Autor**	Vitor Oliveira
Data da última alteração	N/A	**Autor**	N/A
Versão	1.0	**Prioridade**	Baixa
Descrição	O usuário poderá consultar os detalhes de uma compra por produto. Na tela de consulta de compras por produto, o usuário selecionará o registro desejado e selecionar a opção "Detalhes". Ele será redirecionado para uma nova tela, onde as informações detalhadas do produto serão mostradas (Nome da loja, produto, valor unitário, quantidade, valor total e data da compra)		

3.2.2 DIAGRAMA DE CASO DE USO

Nesta seção será o diagrama de caso de uso, que foi elaborado com o objetivo de facilitar entendimento das funcionalidades e do papel do ator no sistema. Na figura 24 pode ser visualizado o diagrama, que foi gerado a partir dos requisitos funcionais descritos na seção anterior.

Figura 24 - Diagrama de Caso de Uso



Fonte: Autoria própria

3.3 ARQUITETURA DO SISTEMA

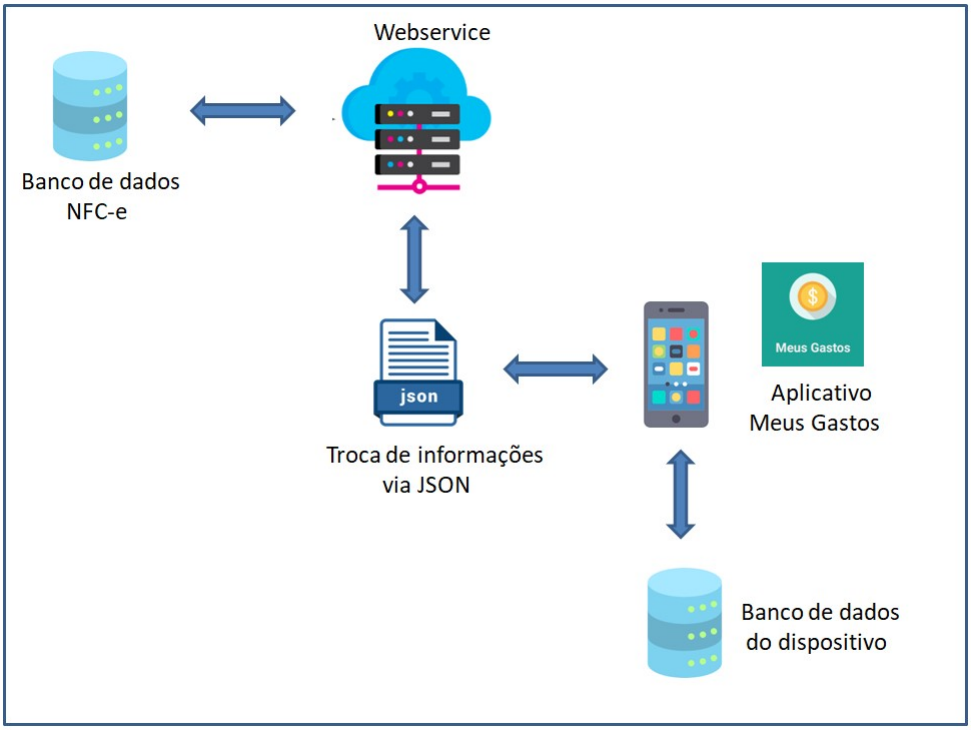
Nesta seção serão descritos os dois módulos que constituem o projeto, o módulo mobile (Aplicativo Meus Gastos) e o módulo web (Web Service NFC-e).

A arquitetura escolhida para o desenvolvimento foi a arquitetura distribuída com 3 camadas (aplicação, web service e banco de dados), onde o web service será responsável por consultar o banco de dados e disponibilizar as informações para o aplicativo móvel, que por sua vez será responsável por efetuar as requisições ao web service de acordo com a funcionalidade

escolhida pelo usuário. Dessa forma, não será necessário o download de todas as informações para o dispositivo, tornando o sistema leve e simples.

A figura 25 descreve a estrutura do sistema como um todo: O banco de dados NFC-e simulará o banco de dados da SEFAZ AM, com os registros de todas as compras realizadas, com as informações das lojas, produtos e consumidores. O web service fará a manutenção dos dados do banco de dados, consultando/incluíndo as informações de acordo com as requisições enviadas pelo aplicativo móvel. A transmissão de dados entre o web service e o aplicativo será realizada no formato JSON, tornando a comunicação rápida e leve. O aplicativo móvel fará requisições para o web service, tanto para consultar as informações das notas fiscais, como para cadastrar as informações do usuário, de acordo com a funcionalidade escolhida. O aplicativo também terá acesso ao banco de dados do dispositivo, para cadastrar/consultar informações básicas do sistema, como categorias de compras e vínculos entre categorias e lojas.

Figura 25 - Estrutura do sistema



Fonte: Autoria própria

3.3.1 MÓDULO WEB SERVICE NFC-e

O módulo web service NFC-e é responsável por mediar a comunicação entre o aplicativo móvel e o banco de dados, permitindo a consulta de informações referentes aos dados das notas fiscais e inserção de informações para criação de usuários do aplicativo.

A figura 26 apresenta um trecho do código da classe ServiceController_NfeDetalhe, utilizada para disponibilizar a lista de todos produtos cadastrados, de acordo com os filtros informados (CPF, descrição do produto, data inicial e final da compra) (Linhas 29 até 32). Podem ser vistas também as anotações utilizadas para mapear os recursos do web service, como @Path (Linhas 16,26), URI onde o recurso que pode ser acessado, @GET que representa o tipo de requisição HTTP (Linha 24) e @Produces (Linha 25), que especifica o tipo de objeto que será produzido, neste caso uma estrutura JSON.

Figura 26 - Utilização de anotações

```
1  package controller;
2
3  import java.sql.SQLException;
14
15
16 @Path("/serviconfe")
17 public class ServiceController_NfeDetalhe {
18
19     private final  NfeDetalheRepository repository = new NfeDetalheRepository();
20
21     /**
22      * Método para listar todos os produtos cadastrados, de acordo com os filtros informados
23      * */
24     @GET
25     @Produces("application/json; charset=UTF-8")
26     @Path("/produtos/")
27
28     public List<NfeDetalhe> TodosProdutos(
29             @QueryParam("cpf") String cpf,
30             @QueryParam("prod") String prod,
31             @QueryParam("dtini") String dtini,
32             @QueryParam("dtfim") String dtfim) throws SQLException, ParseException{
33         return repository.TodosProdutos(cpf,prod,dtini,dtfim);
34     }
35
36 }
```

Fonte: Autoria própria

O método TodosProdutos() retorna um arraylist da classe NfeDetalhe, que registra a lista de produtos vinculados na nota fiscal. A requisição retorna todos os produtos registrados em um determinado período, independente de estarem relacionados na mesma NFC-e.

Através da classe NfeDetalheRepository, que agrupa o repositório de métodos para comunicação com o banco de dados NFC-e, os objetos são preenchidos e convertidos para o formato JSON. Todo o processo de conversão dos objetos Java para a estrutura JSON é abstraido pelo framework JERSEY, utilizando o parser GSON. Na figura 27 pode ser visto um exemplo de uma estrutura resultado de uma requisição enviada para o web service. Nela podem ser identificados os atributos da classe NfeDetalhe (Linhas 4 à 6 e 17,18), com as informações do produto, os atributos da classe NfeCapa (Linhas 8 à 11), com as informações do cabeçalho da nota fiscal e por fim, os atributos da classe Loja (Linhas 12 e 13), com as informações do estabelecimento que efetuou a emissão da NFC-e.

Figura 27 - Trecho de estrutura JSON


```
1
2  {
3    {
4    "codigo": "EA0000123",
5    "descricao": "IMPRESSORA HPDESKJET 2135",
6    "iddetalhe": 15,
7    "nfecapa":       {
8       "cnpj": "02.337.524/0001-06",
9       "cpf": "93225733253",
10      "id": 4,
11      "loja":          {
12         "endereco": "Rua Belo Horizonte, 686 - Aleixo, Manaus-AM CEP: 69.060-601",
13         "razaosocial": "Info Store Computadores da Amazônia Ltda"
14      },
15      "valortotal": 350
16   },
17   "qtd": 1,
18   "valor": 290
19 }
```


Fonte: Autoria própria

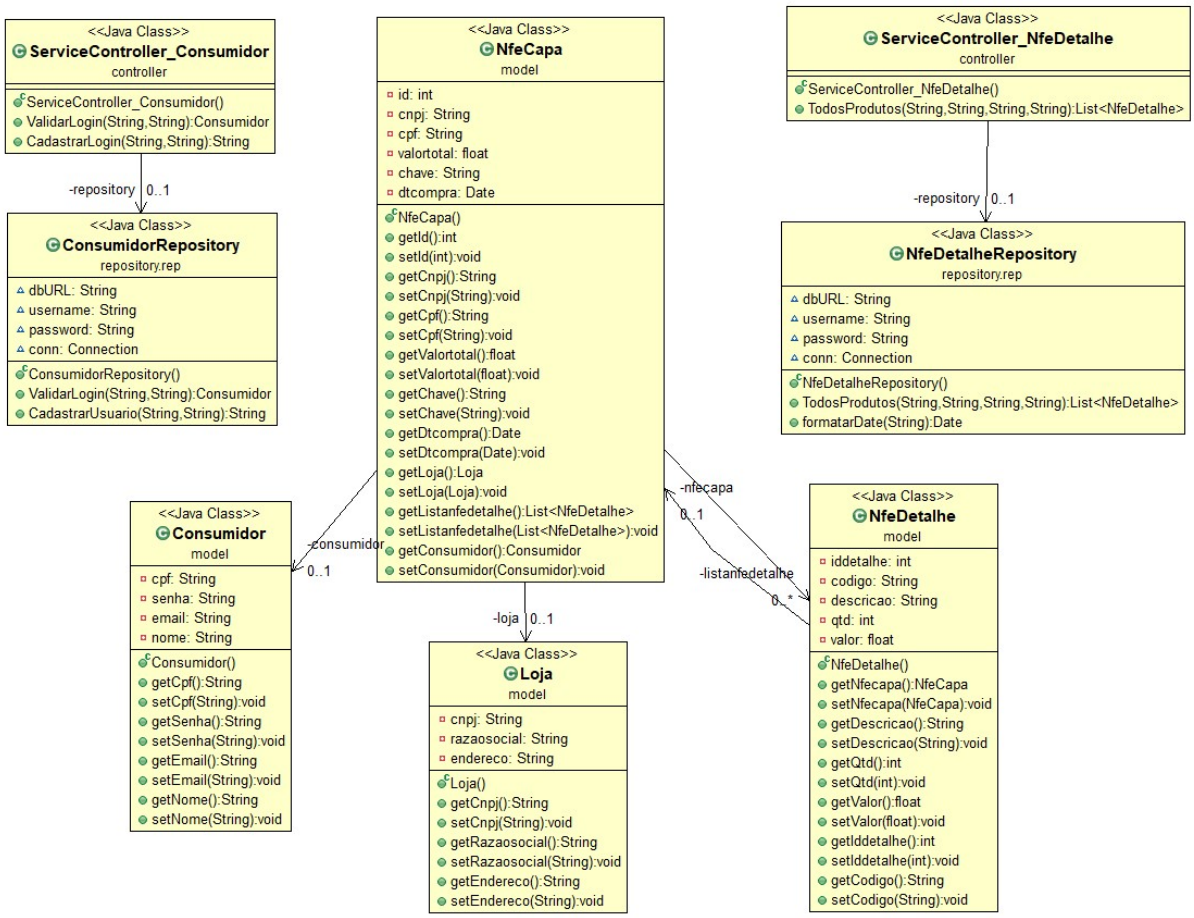
O web service possui 4 classes na camada model: Loja, Consumidor, NfeCapa e NfeDetalhe.

1) Loja: Classe com as informações da loja que efetuou a emissão da nota fiscal;
2) Consumidor: Classe com as informações do consumidor que registrou o CPF na nota fisca emitida;
3) NfeCapa: Classe com as informações da capa da nota fiscal;
4) NfeDetalhe: Classe com as informações dos produtos vinculados na nota fiscal.

As classes do model são utilizadas para que as informações consultadas do banco de dados sejam mapeadas e convertidas em estruturas JSON, para que assim sejam devolvidas ao aplicativo. Na camada controller, para cada classe do model, existem duas outras classes: ServiceController e Repository.

A classe ServiceController é responsável por disponibilizar os recursos através do web service, no padrão REST e a classe Repository contém os métodos responsáveis por efetuar a comunicação com o banco de dados NFC-e. coleção de pares nome/valor, onde o nome é o atributo que foi mapeado na classe Java e o valor o conteúdo desse atributo. A figura 28 apresenta as classes da camada model e o exemplos das classes ServiceController e Repository das classes Consumidor e NfeDetalhe.

Figura 28 - Exemplo de classes do web service



Fonte: Autoria própria

3.3.2 MÓDULO APLICATIVO MÓVEL

O módulo aplicativo móvel (Aplicativo Meus Gastos) é o principal do projeto, sendo por meio dele que o usuário irá utilizar todas as funcionalidades do sistema. O aplicativo foi desenvolvido para ser utilizado em dispositivos que possuam o sistema operacional Android, e tem como proposta permitir que o usuário consulte de forma fácil e rápida todos os gastos em compras realizadas no estado do Amazonas, através das NFC-e emitidas em que o CPF do consumidor for registrado. A figura 29 apresenta a tela inicial do aplicativo, onde o usuário pode efetuar o acesso ao sistema, informando o seu CPF e senha. Caso não possua acesso ao sistema, o usuário poderá efetuar seu cadastro no próprio aplicativo, clicando na opção “Registre-se”.

Figura 29 - Tela de login do aplicativo Meus Gastos

Fonte: Autoria própria

Na tela de cadastro de novo usuário, que pode ser vista na figura 30, serão solicitadas as seguintes informações para criação de um novo acesso: CPF, nome completo, email e senha. O sistema validará as informações fornecidas pelo usuário, e caso não possua um acesso previamente cadastrado, as informações serão inseridas através do web service no banco de dados NFC-e.

Figura 30 - Tela de cadastro de novo usuário

Fonte: Autoria própria

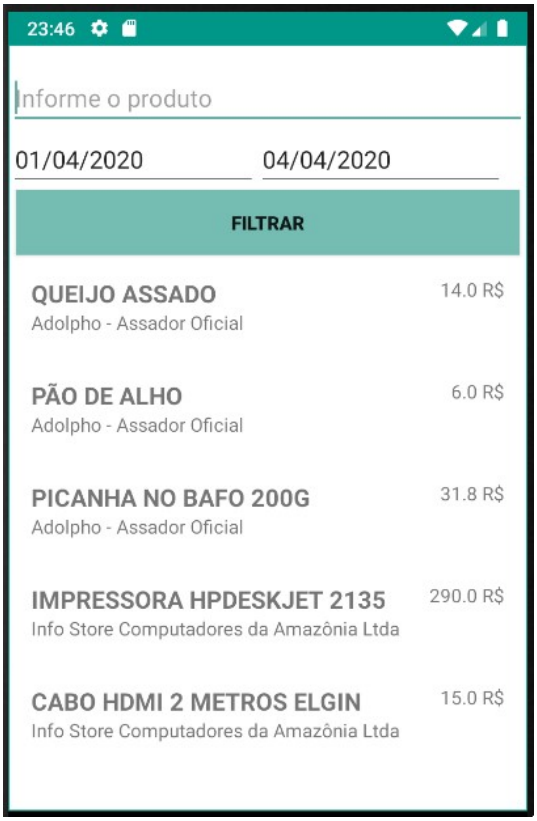
Após efetuar o login no sistema, o usuário será redirecionado para a tela principal do aplicativo, a tela de menu (Figura 31), onde encontrará as principais funcionalidades do sistema: Consulta de gastos por lojas, localizada na opção “Lojas”, consulta de gastos por categorias, localizada na opção “Gastos por Categorias”, consulta de gastos por produtos, na opção “Produtos” e finalmente a função de cadastrar novas categorias, na opção “Categorias”.

Figura 31 - Tela de menu principal

Fonte: Autoria própria

Ao selecionar a opção “Produtos”, o usuário será redirecionado para a tela de consulta de gastos por produtos. Na tela, usuário poderá utilizar filtros com o nome do produto e/ou o período em que foi efetuada a compra. Ao clicar no botão “Filtrar”, o aplicativo efetuará uma requisição para o web service, que retornará uma lista de produtos de acordo com os parâmetros enviados, conforme pode ser visto na figura 32. As informações retornadas do web service, são: Descrição do produto, razão social do vendedor e valor total do item.

Figura 32 - Tela de consulta de gastos por produtos



Fonte: Autoria própria

Na figura 33 é apresentado um trecho de código da classe Activity_Produtc, do método de criação da Activity. Nas linhas 52 e 53 podem ser vistos os parâmetros recebidos da Activity principal, sendo o CPF do usuário logado no aplicativo e o IP do endereço onde o web service está disponível. Entre as linhas 55 e 57 está a declaração da classe Retrofit, onde na linha 56 definimos a URL base onde os recursos do web service serão acessados e na linha 57 configuramos o PARSER GSON para converter as informações da estrutura JSON para o objeto da classe JAVA.

Figura 33 - Trecho do código da classe Activity_Produtc()

```
45      @Override
46      protected void onCreate(Bundle savedInstanceState) {
47          getSupportActionBar().hide();
48          super.onCreate(savedInstanceState);
49          setContentView(R.layout.activity_product);
50
51          Bundle parametros = getIntent().getExtras();
52          parCPF = parametros.getString( key: "parCPF");
53          parIp  = parametros.getString( key: "parIP");
54
55          retrofit = new Retrofit.Builder().
56                  baseUrl(parIp + "/WebServiceRest/rest/").
57                  addConverterFactory(GsonConverterFactory.create()).build();
58
59          viewProdutos = findViewById(R.id.rviewProdutos);
60          editFiltroProduto = findViewById(R.id.editdescProduto);
61
62          editFiltroDtIni = findViewById(R.id.editDataIni);
63          editFiltroDtIni.setOnClickListener((v) → {
```

Fonte: Autoria própria

Na figura 34 é apresentado o método recuperarListaDados() da classe Activity_Produtc(), responsável por consultar o web service com os parâmetros informados pelo usuário (CPF, produto, data inicial e data final) e preencher as informações recebidas na tela do aplicativo. Na linha 143, o método enqueue da classe Call efetua uma requisição assíncrona que será executada por uma Thread separada da principal. Na linha 145 temos a declaração do método onResponse(), responsável por capturar a resposta enviada pelo web service.

O web service retorna uma estrutura JSON, que é carregada em um ArrayList de produtos (Linha 146), que será enviada para o AdapterProdutos, que por sua vez será utilizado para exibir as informações na tela do aplicativo.

Figura 34 - Trecho do código do método recuperarListaDados()

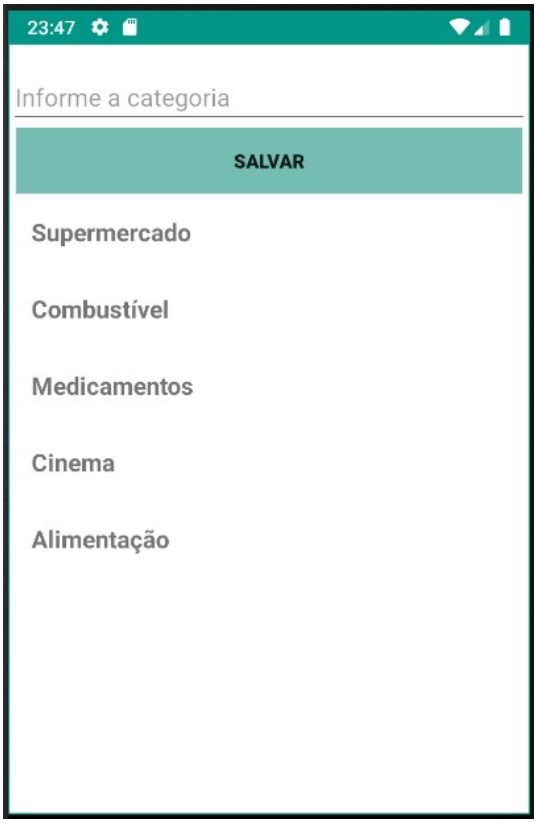
```
138     private void recuperarListaDados(String cpf, String produto, String datini, String datafim)
139     {
140         DadosService dadosService = retrofit.create(DadosService.class);
141         Call<List<NfeDetalhe>> call = dadosService.recuperarListaDados(cpf,produto,datini,datafim);
142
143         call.enqueue(new Callback<List<NfeDetalhe>>() {
144             @Override
145             public void onResponse(Call<List<NfeDetalhe>> call, Response<List<NfeDetalhe>> response) {
146                 listaProdutos = response.body();
147
148                 if (listaProdutos.size() != 0)
149                 {
150                     // Configuração do Adapter
151                     final AdapterProdutos adapter = new AdapterProdutos(listaProdutos);
152
153                     // Configuração do RecyclerView
154                     RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(getApplicationContext());
155                     viewProdutos.setLayoutManager(layoutManager);
156                     viewProdutos.setHasFixedSize(true);
157                     viewProdutos.setAdapter(adapter);
158                 }
159                 else
160                 {
161                     Toast toast = Toast.makeText(getApplicationContext(), text: "Nenhum registro encontrado.",Toast.LENGTH_SHORT);
162                 }
163             }
```

Fonte: Autoria própria

No aplicativo Meus Gastos, uma das funcionalidades mais importantes é a possibilidade do usuário criar categorias de acordo com sua necessidade e vincular seu gastos nessas categorias, para assim poder visualizar de forma rápida e eficiente como estão divididos seus gastos ao longo do tempo.

Na tela de cadastro de categorias, que pode ser vista na figura 21, o usuário pode consultar e cadastrar as categorias de acordo com seu perfil de consumidor, e posteriormente vincular seus gastos por categorias.

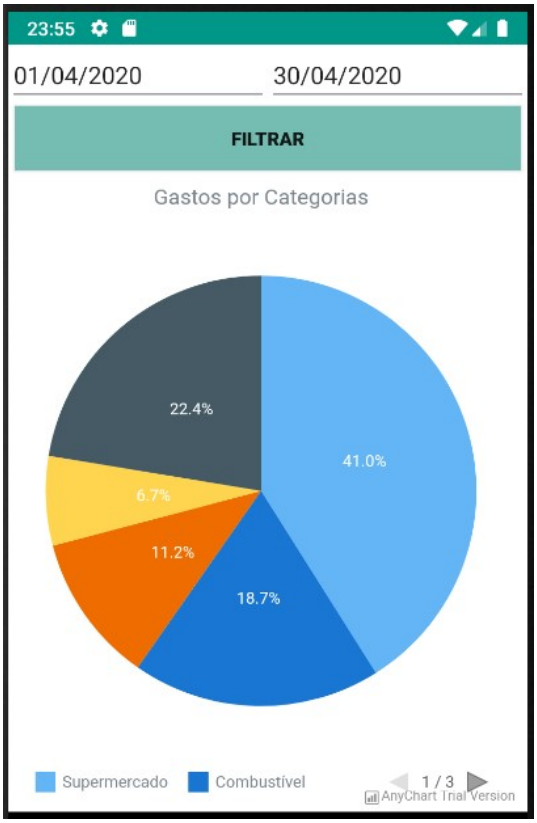
Figura 35 - Tela de consulta/cadastro de categorias



Fonte: Autoria própria

Após vincular os registros de compras em suas respectivas categorias, o usuário poderá consultar, em forma de gráfico, os valores gastos em determinado período de tempo, na tela de consulta de gastos por categorias, figura 36.

Figura 36 - Tela de consulta de gastos por categorias



Fonte: Autoria própria

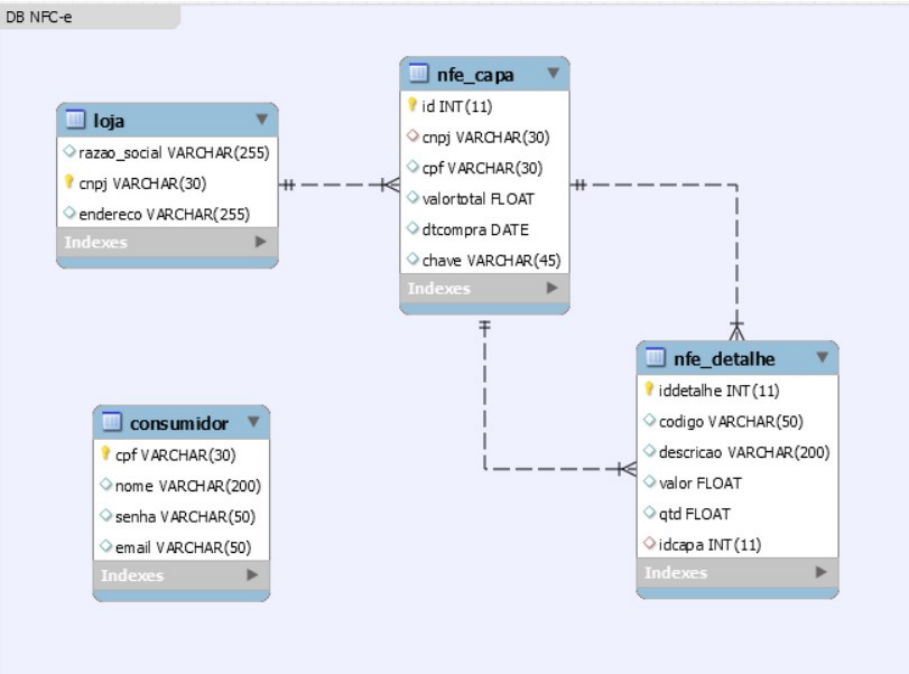
3.3.3 MODELAGEM DO BANCO DE DADOS

A modelagem do banco de dados desse projeto consiste em duas estruturas separadas, sendo uma utilizada pelo módulo web service e outra utilizada pelo aplicativo móvel.

O módulo web service NFC-e conta com uma base de dados modelada e configurada no SGBD MySQL, banco de dados relacional, que possui portabilidade, excelente desempenho e estabilidade, sendo uma escolha segura para sistemas voltados para WEB. O banco de dados possui 4 tabelas consumidor, loja, nfe_capa e nfe_detalhe. Na figura 37 oferecemos o relacionamento entre as tabelas. A tabela consumidor contém as informações referentes ao usuário, que estará vinculado à NFC-e através do CPF. A tabela loja possui as informações dos estabelecimentos que efetuaram a emissão das notas fiscais.

As tabelas nfe_capa e nfe_detalhe são respectivamente, as informações do cabeçalho da nota e os produtos e valores listados na nota fiscal emitida. Uma particularidade da estrutura de banco de dados é a ausência de uma chave estrangeira entre as tabelas consumidor e nfe_capa. O motivo disso é a possibilidade de o usuário não possuir cadastro no sistema, mas solicitar a emissão de notas fiscais com seu CPF, causando um erro durante a inclusão das informações no banco de dados.

Figura 37 - Modelagem do banco de dados MySQL



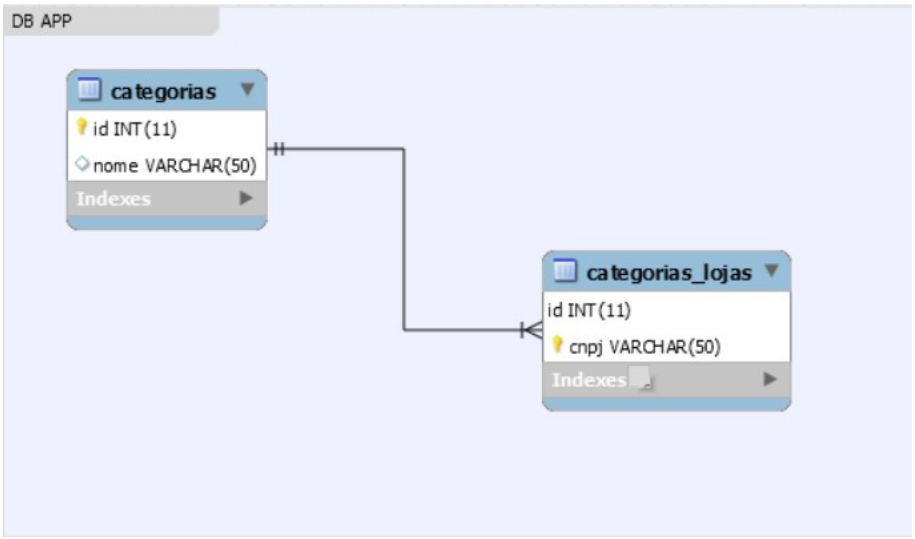
Fonte: Autoria própria

O módulo aplicativo móvel por sua vez, conta com uma base de dados bem mais simples, modelada no SGBD nativo do android, o SQLite, um banco de dados leve e eficiente para utilização em aplicações desenvolvidas para dispositivos móveis. Quase que a totalidade das informações utilizadas pelo aplicativo serão provenientes do web service, logo a estrutura do banco de dados é simplificada, contendo apenas duas tabelas: categorias e categorias_lojas. A tabela de categorias contém as informações cadastradas pelo usuário no aplicativo, com o id e nome da categoria escolhida. A outra tabela é a categorias_lojas que conterá o relacionamento entre as categorias

cadastradas e o CNPJ dos estabelecimentos em que o usuário efetuou compras.

Esse relacionamento serve para que o usuário possa, de forma dinâmica e totalmente configurável, controlar seu gastos de acordo com categorias criadas para atender suas necessidades. A figura 38 exibe o esquema do banco de dados SQLite, com seus respectivos relacionamentos.

Figura 38 - Modelagem do banco de dados SQLite



Fonte: Autoria própria

3.3.4 SIMULAÇÃO DO SITE DA SEFAZ

Para efetuar o desenvolvimento do web service que simulará a aplicação da SEFAZ AM, foi necessário gerar uma base de dados com informações de notas fiscais emitidas no ambiente de produção. As informações utilizadas para gerar o banco de dados foram consultadas no Portal Estadual da NFC-e, onde o consumidor que informar o CPF no momento da emissão da nota fiscal, pode consultá-las através da página http://portalnfce.sefaz.am.gov.br/ (Figura 39).

Figura 39 - Portal Estadual da NFC-e

Fonte: Portal Estadual da NFC-e, 2020

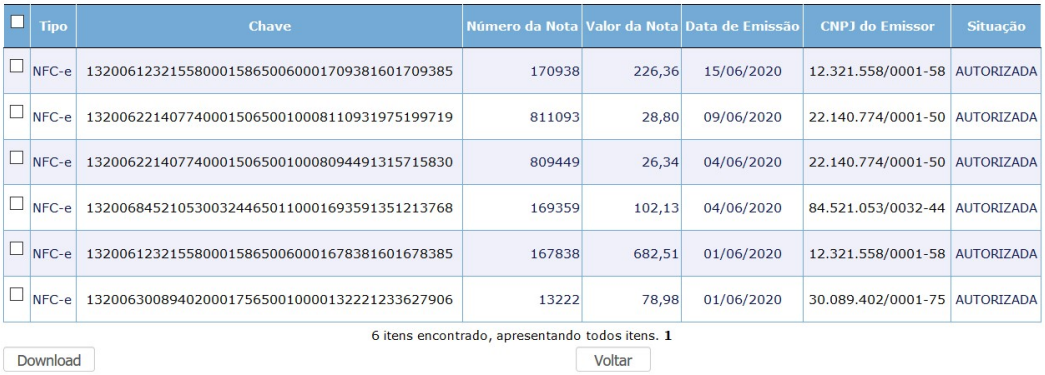
Para efetuar a consulta das notas fiscais emitidas com seu CPF vinculado, o usuário deve cadastrar suas informações para ter acesso ao portal. Após a finalização do cadastro, o consumidor poderá acessar todas as notas atráves do CPF registrado (Figura 40).

Figura 40 - Consulta de Nota Fiscal por CPF

Fonte: Portal Estadual da NFC-e, 2020

As notas fiscais podem ser consultadas por período de emissão e/ou CNPJ do emissor, onde o usuário pode visualizar a NFC-e como foi impressa originalmente ou baixar o arquivo XML referente à nota fiscal. Para efetuar o download dos arquivos desejados, o usuário deve selecionar as notas e clicar em Download (Figura 41).

Figura 41 - Arquivos de Notas Fiscais em XML



	Tipo	Chave	Número da Nota	Valor da Nota	Data de Emissão	CNPJ do Emissor	Situação
☐	NFC-e	13200612321558000158650060001709381601709385	170938	226,36	15/06/2020	12.321.558/0001-58	AUTORIZADA
☐	NFC-e	13200622140774000150650010008110931975199719	811093	28,80	09/06/2020	22.140.774/0001-50	AUTORIZADA
☐	NFC-e	13200622140774000150650010008094491315715830	809449	26,34	04/06/2020	22.140.774/0001-50	AUTORIZADA
☐	NFC-e	13200684521053003244650110001693591351213768	169359	102,13	04/06/2020	84.521.053/0032-44	AUTORIZADA
☐	NFC-e	13200612321558000158650060001678381601678385	167838	682,51	01/06/2020	12.321.558/0001-58	AUTORIZADA
☐	NFC-e	13200630089402000175650010000132221233627906	13222	78,98	01/06/2020	30.089.402/0001-75	AUTORIZADA

6 itens encontrado, apresentando todos itens. **1**

Download Voltar

Fonte: Portal Estadual da NFC-e, 2020

De posse dos arquivos no formato XML referentes às notas fiscais emitidas, foi necessário desenvolver um programa para consultar as informações em cada arquivo e salvar no banco de dados relacional descrito no tópico anterior. Para isso, foi elaborado um sistema para ler os arquivos baixados, que percorre as TAGS do XML com as principais informações e registra os dados no banco de dados. Foram coletadas informações do produto (Código, descrição, valor unitário e quantidade), dados do emissor da nota fiscal (CPNJ, razão social e endereço) e informações do cabeçalho da nota fiscal (Data de emissão, chave da nota, valor total).

4 CONSIDERAÇÕES FINAIS

Este trabalho contém a descrição do processo de desenvolvimento de duas aplicações, uma desenvolvida para aplicativos móveis e outra voltada para a web, na forma de um web service. O objetivo geral do trabalho foi alcançado, sendo que para isso fosse possível, muitos conhecimentos e técnicas novas foram aprendidos ao longo do tempo que esse projeto foi desenvolvido.

4.1 DIFICULDADES ENCONTRADAS

A primeira grande dificuldade encontrada neste projeto foi a impossibilidade de integração dos sistemas com as informações reais contidas nos servidores da SEFAZ AM, por motivos burocráticos que não puderam ser resolvidos. Com isso em mente, foi necessário não desenvolver um, mas dois sistemas completamente diferentes, que precisam se comunicar, e possuem características técnicas e necessidades específicas.

Como não possuo domínio total nas linguagens de programação necessárias para o desenvolvimento dos módulo do projeto, foram necessárias diversas horas de pesquisa e testes para que os requisitos levantados no início do projeto pudessem ganhar forma no aplicativo e web service propostos.

Todo esse processo serviu de grande aprendizado, contribuindo para o desenvolvimento de habilidades necessárias que serão utilizadas no ambiente de trabalho.

4.2 TRABALHOS FUTUROS

Como trabalhos futuros, foram definidas 2 prioridades: efetuar uma nova tentativa de parceria com a SEFAZ Amazonas para que o aplicativo possa acessar os dados de produção e melhorar a interface do aplicativo como um todo, adicionando novas funcionalidades.

4.3 LIÇÕES APRENDIDAS

Devido à complexidade de desenvolver duas aplicações únicas, com arquiteturas complementares (cliente/servidor), recursos e funcionalidades diferenciadas, foi necessário aprofundar conhecimentos adquiridos durante o todo o curso de graduação, desde as etapas de análise e obtenção de requisitos, elaboração da arquitetura do sistema, modelagem do banco de dados e finalmente o desenvolvimento das aplicações que fazem parte do projeto. Foram necessários horas de estudo para a compreensão de diversas ferramentas, APIs e frameworks para desenvolvimento de aplicativos móveis, comunicação entre aplicativo e web services através de JSON e desenvolvimento de web services no padrão RESTFul.

Todo esse processo de estudo e desenvolvimento levou à uma grande obtenção de conhecimento nas áreas citadas anteriormente, elevando os conhecimentos adquiridos durante o curso de graduação

5 REFERÊNCIAS BIBLIOGRÁFICAS

ANICHE, Maurício; **Testes Automatizados de Software - Um guia prático.** São Paulo: Casa do Código - Série CAELUM.

ANY CHARTS; **AnyCharts Android Charts**. Disponível em < https://www.anychart.com/pt/technical-integrations/samples/android-charts>. Acesso em 01 Mar. 2020.

BHADE, Jaydevi; YADAV, Prof.himanshu. **Evaluation of Android Networking Libraries**. Radharaman Institute of Technology & science, Bhopal, Índia, 2019. Disponível em <https://ijsret.com/wp-content/uploads/2019/07/IJSRET_V5_issue4_393.pdf>. Acesso em 01 Mar. 2020.

DIMARZIO, J.F.; **Android Programming with Android Studio.** 4º ed. Indianapolis: WROX, 2017.

DUARTE, Roberto Dias. **Sped - Sistema Público de Escrituração Digital.** 2º ed. Rio de Janeiro: Confederação Nacional do Comércio de Bens, Serviços e Turismo, 2012.

ENCAT; **Estatísticas NFC-e.** Disponível em < http://nfce.encat.org/estatisticas>. Acesso em 20 Mar. 2019.

FARIA, Fernanda B. et al. **Evolução e Principais Características do IDE Eclipse**. 2010. Disponível em: <https://www.enacomp.com.br/2010/anais/artigos/completos/enacomp2010_23.pdf>. Acesso em: 01 Fev. 2020.

FONSECA, Rúben; SIMÕES, Alberto. **Alternativas ao XML: YAML e JSON**. 2007. 14 f . Disponível em <https://repositorium.sdum.uminho.pt/bitstream/1822/6230/1/xmlyamljson07.pdf>. Acesso em: 12 mar. 2020.

FRIESEN, Jeff; **JAVA XML and JSON**. Dauphin : APRESS, 2016.

GARGENTA, Marko; **Learning Android.** Estados Unidos: O'REILLY, 2011.

GONÇALVES, Eduardo. **JSON Tutorial**. 2012. Disponível em: https://www.devmedia.com.br/json-tutorial/25275. Acesso em: 15 abr. 2020.

JERSEY; **Eclipse Jersey is a REST framework that provides a JAX-RS (JSR-370) implementation and more.** Disponível em <https://eclipse-ee4j.github.io/jersey/>. Acesso em 03 Mar. 2020.

JSON. **Introducing JSON**. Disponível em: <https://www.json.org/json-en.html>. Acesso em: 15 Fev 2020.

KALIN, Martin; **Java Web Services - Up and Running.** 2º ed. Estados Unidos: O'REILLY, 2013.

MEDINA, Alex; **Diferencias entre SOAP x REST**. Disponível em: < https://www.alexmedina.net/diferencias-entre-soap-vs-rest/>. Acesso em 04 Abr. 2020.

NURSEITOV, Nurzhan *et al*. **Comparison of JSON and XML Data Interchange Formats: A Case Study**. 2009. 6 f. Department Of Computer Science, Montana State University, Bozeman, 2009. Disponível em: <https://www.cs.montana.edu/izurieta/pubs/caine2009.pdf>. Acesso em 20 Abr. 2020.

ORACLE; **The Java EE 6 Tutorial**. Disponível em <https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>. Acesso em 01 Mar. 2020.

ORGANIZZE; **Sistema para controle financeiro Organizze**. Disponível em:<https://www.organizze.com.br/>. Acesso em: 09 Mar. 2019.

PORTAL ESTADUAL DA NFC-e; **Nota Fiscal do Consumidor Eletrônica**. Disponível em: <http://portalnfce.sefaz.am.gov.br/>. Acesso em 27 Abr. 2020.

PORTAL DA FAZENDA - SP; **Sobre a nota fiscal paulistana.** Disponível em: <https://portal.fazenda.sp.gov.br/servicos/nfp/Paginas/Sobre.aspx>. Acesso em: 20 Mar. 2019.

PORTAL ANDROID DEVELOPERS; **Conheça o Android Studio.** Disponível em: <https://developer.android.com/studio/intro>. Acesso em: 17 Mar. 2020.

RECEITA FEDERAL; **NFC-e: Manual de Padrões Técnicos DANFE NFC-e e QR-CODE**. Disponível em < http://www.nfe.fazenda.gov.br/portal/listaConteudo.aspx?tipoConteudo=33ol5hhSYZk=>. Acesso em: 20 Mar. 2019.

RETROFIT; **A type-safe HTTP client for Android and Java**. Disponível em: <https://square.github.io/retrofit/>. Acesso em: 06 Jun. 2019.

SAUDATE, Alexandre; **REST - Construa API's inteligentes de maneira simples**. São Paulo: Caso do código, 2014.

SANDOVAL, Jose. **RESTful Java Web Services - Master core REST concepts and create RESTful web services in Java**. Birmingham, UK: Packt Publishing Ltd, 2009.

SPC BRASIL; **Cresce para 63% o número de consumidores que controlam suas finanças, revelam CNDL/SPC Brasil e Banco Central**. Disponível <https://www.spcbrasil.org.br/pesquisas/pesquisa/5873>. Acesso em: 09 Mar. 2019.

ZARELLI, Guilherme Biff. **Como funciona o SOAP – Protocolo Simples de Acesso a Objetos**. 2012. Disponível em: <https://helpdev.com.br/2012/03/22/como-funciona-o-soap-protocolo-simples-de-acesso-a-objetos/>. Acesso em: 01 mar. 2019.

6 ANEXO 1 - PRINCIPAIS CASOS DE USO

Código	UC001
Nome	Cadastrar acesso
Objetivo	Cadastrar o acesso do usuário no sistema
Ator	Usuário
Pré-condição	N/A
Fluxo normal	1. Informar e-email
	2. Informar senha e confirmação de senha
	3. Informar CPF
	4. Informar nome completo
	5. Clicar no botão "Salvar"
	6. Sistema exibe a mensagem "Cadastro realizado com sucesso."
Fluxo alternativo	1. Em caso de divergência nas informações, exibir uma mensagem de erro para o usuário
	2. Caso o CPF informado já esteja cadastrado, exibir uma mensagem para o usuário
Pós-condição:	O usuário terá um acesso cadastrado no sistema

Código	UC007
Nome	Vincular Gastos às Categorias
Objetivo	Vincular os registros de compras por categorias de gastos
Ator	Usuário
Pré-condição	1. Usuário com acesso ao sistema
	2. Categoria cadastrada
Fluxo normal	1. Na tela de gastos por lojas, selecionar um registro de gasto
	2. Escolher uma categoria na tela de seleção exibida
	3. Clicar no botão "Salvar"
	4. Sistema exibe a mensagem: "Gasto categorizado com sucesso.
Fluxo alternativo	1. Caso o tipo de gasto já tenha sido vinculado à uma mensagem de erro para o usuário: "Gasto já vinculado à uma categoria."
Pós-condição:	-

Código	UC008
Nome	Listar Gastos por Categorias
Objetivo	Consultar os gastos vinculados às categorias cadastradas no aplicativo, em determinado intervalo de datas
Ator	Usuário
Pré-condição	1. Usuário com acesso ao sistema
	2. Gastos vinculados às categorias
Fluxo normal	1. Acessar a tela de consulta de gastos por categorias
	2. Informar o intervalo de datas no filtro de pesquisa
	3. Clicar no botão "Filtrar"
	4. O Sistema exibe o gráfico de acordo com as informações consultadas
Fluxo alternativo	1. Caso nenhum registro seja encontrado, o sistema exibe a seguinte mensagem: "Nenhum registro encontrado."
Pós-condição:	-

Código	UC009
Nome	Listar Gastos por Lojas
Objetivo	Consultar os registros de gastos por lojas
Ator	Usuário
Pré-condição	1. Usuário com acesso ao sistema
Fluxo normal	1. Acessar a tela de gastos por lojas
	2. Informar o intervalo de datas no filtro de pesquisa
	3. Clicar no botão "Filtrar"
	4. O sistema exibe os registros de acordo com o filtro informado
Fluxo alternativo	1. Caso nenhum registro seja encontrado, o sistema exibe a seguinte mensagem: "Nenhum registro encontrado."
Pós-condição:	-

Código	UC011
Nome	Listar Gastos por Produtos
Objetivo	Consultar os registros de gastos por produtos
Ator	Usuário
Pré-condição	1. Usuário com acesso ao sistema
Fluxo normal	1. Acessar a tela de gastos por lojas
	2. Informar o intervalo de datas no filtro de pesquisa
	3. Clicar no botão "Filtrar"
	4. O sistema exibe os registros de acordo com o filtro informado
Fluxo alternativo	1. Caso nenhum registro seja encontrado, o sistema exibe a seguinte mensagem: "Nenhum registro encontrado."
Pós-condição:	-