UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOÃO ANTONIO SIMONETTI PUCCI

RUAN FELIPE STARON

APLICATIVO ANDROID PARA CONFERÊNCIA DE VALOR DE COMPRAS E CRIAÇÃO DE LISTAS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2018

JOÃO ANTONIO SIMONETTI PUCCI

RUAN FELIPE STARON

APLICATIVO ANDROID PARA CONFERÊNCIA DE VALOR DE COMPRAS E CRIAÇÃO DE LISTAS

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

Orientador: Profª. Dra. Mônica Hoeldtke Pietruchinski

PONTA GROSSA

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

APLICATIVO ANDROID PARA CONFERÊNCIA DE VALOR DE COMPRA E CRIAÇÃO DE LISTAS

por

JOÃO ANTONIO SIMONETTI PUCCI
RUAN FELIPE STARON

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 22 de Maio de 2018 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Profª. Dra. Mônica Hoeldtke Pietruchinski
Orientador(a)

_____
Prof. Dr. Diego Roberto Antunes
Membro titular

_____
Prof. MSc. Clayton Kossoski
Membro titular

_____
Profª. Dra Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. André Pinz Borges
Coordenador do curso

- A Folha de Aprovação assinada encontra-se na Coordenação do Curso -

RESUMO

PUCCI, João Antonio Simonetti, STARON, Ruan Felipe. **Aplicativo *Android* para conferência de valor de compras e criação de listas**. 2018. Número total de folhas 61. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2018.

Este trabalho apresenta o desenvolvimento do aplicativo para *smartphones* "Mercadinho", cuja principal função é realizar o cálculo do valor total de compras com base em uma lista de produtos criada anteriormente pelo usuário. Grandes supermercados operam utilizando diversos sistemas computacionais a fim de evitar fraudes durante as cobranças dos produtos, sejam estas fraudes cometidas por parte de funcionários, clientes mal-intencionados ou por pura desatenção. Já mercados menores ainda carecem de maneiras de garantir o valor da compra ao final desta, oferecendo maior preocupação ao cliente durante a finalização do processo de compra. Para auxiliar na resolução deste problema, foi desenvolvido um aplicativo móvel e um *website*, bem como uma base de dados *online* compartilhada. Para isso, foram utilizadas diversas tecnologias para diferentes etapas do desenvolvimento, como banco de dados, metodologias de processos ágeis, *web service*, *Java*, PHP, sistema de controle de versão, entre outros. As tecnologias utilizadas são descritas e explanadas ao decorrer dos capítulos subsequentes. Assim, é possível atingir o problema proposto através do uso de tecnologia móvel, que tem seu uso cada vez mais intrínseco à sociedade moderna cada vez mais.

Palavras-chave: *Android*. Computação Móvel. *Web Services*. Compras. Mercado.

ABSTRACT

PUCCI, João Antonio Simonetti, STARON, Ruan Felipe. **Android application for conference of purchase values and shopping lists creation**. 2018. 61p. Undergraduate thesis (Graduation in Systems Analysis and Development) – Federal Technological University of Paraná. Ponta Grossa, 2018.

This final paper describes the development of the smartphone application "Mercadinho", whose main function is to calculate the value to be paid of a purchase, based in a list of products created by the user beforehand. Large supermarkets work using different computer systems in order to avoid frauds. Be these frauds committed by malicious employees, clients, or by legit inattention. In other hand, smaller markets still need to find ways to grant the final value of the products at the final step of the purchase, offering a concern to the client. To help in the solving of this problem a mobile app, a website and a shared database were created. To do this, various technologies were used in different steps of the development process, like database systems, agile methodologies, web services, Java, PHP, version control systems, among others. The Technologies used are described and explained throughout the subsequent chapters. So, we believe that is possible to hit the proposed problem with mobile technology, which is having its use always more inherent to modern society each day.

Palavras-chave: Android. Mobile. Web Services. Shopping. Market.

LISTA DE SIGLAS

API	*Application Programming Interface*
IDE	*Integrated Development Environment*
HTML	*Hypertext Markup Language*
XML	*Extensible Markup Language*
CGI	*Common Gateway Interface*
HTTP	*Hypertext Transfer Protocol*
ASP	*Active Server Pages*
CSS	*Cascade Stylesheets*
SGBD	*Sistema Gerenciador de Banco de Dados*
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
AWS	*Amazon Web Services*
ORM	*Object-Relational Mapping*
SOAP	*Simple Object Access Protocol*
REST	*Representational State Transfer*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da tecnologia móvel, uma onda de novos aplicativos para *smartphones* surgem dia após dia com o intuito de trazer maior comodidade às atividades humanas que antes eram feitas de forma manual, beneficiando os usuários com economia de tempo e energia. O aplicativo para *smartphones* desenvolvido no decorrer deste trabalho de conclusão de curso visa operar neste mesmo sentido, automatizando as operações morosas e cansativas.

Em um cenário hipotético, um cliente que faz compras em um mercado utiliza uma calculadora para gerar o valor final da compra antes de dirigir-se aos caixas, porém se houver divergência entre o valor cobrado e o valor indicado pela calculadora, não há como saber em qual produto ocorreu o erro de cobrança sem refazer todo o cálculo.

Este *software* foi desenvolvido para minimizar esse problema e agilizar o processo de compra permitindo ao usuário adiantar-se ao procedimento de pagamento, dessa forma com uma lista de produtos e os respectivos valores que se encontram na gôndola ficará mais fácil encontrar onde ocorreu o erro.

Além disso, como as informações obtidas durante a compra são disponibilizadas em um ambiente *online* compartilhado por todos os usuários, outros compradores poderão comparar os preços de diferentes mercados, podendo economizar em suas futuras compras.

O projeto então propõe a construção de um aplicativo móvel com o intuito de minimizar o problema. O *software, chamado* “Mercadinho” oferece soluções objetivas e de impacto imediato no dia a dia dos usuários. Com este propósito, o programa foi desenvolvido para plataforma *Android*, visto que pesquisas indicam que cerca de 37.93% dos usuários de computadores no mundo utilizam o sistema operacional da *Open Handset Alliance*, superando até mesmo o *Microsoft Windows*, o SO mais popular para computadores de mesa. (ZURIARRAIN, 2017). Além disso, o custo de hospedagem do aplicativo na *Play Store* é baixo, $25,00 em pagamento único contra $99,00 pagos anualmente na loja da *Apple*.

O escopo do problema foi limitado à realização de compras em mercados, onde usualmente o comprador organiza uma lista de compras para fazer uma previsão do valor a ser pago e ter uma visão geral de suas economias no momento da compra.

Assim, a equipe aborda a concepção do aplicativo de forma que os resultados obtidos com a implementação e distribuição respondam a seguinte pergunta: De que forma é possível desenvolver um aplicativo móvel que gerencie uma lista de compras para otimizar o controle financeiro do usuário?

1.1 OBJETIVO GERAL

O objetivo geral do projeto é implementar um sistema de lista de compras com inclusão e alteração de descrição, quantidade e preço de produtos através do escaneamento do código de barras deste.

Caso um produto ainda não esteja cadastrado, o usuário poderá cadastrá-lo em sua base de dados local escaneando o código de barras do produto e preenchendo os campos referentes à descrição, quantidade e preço. O produto cadastrado pelo cliente ficará disponível na base de dados *online*, para uso por outros usuários, sem a necessidade do escaneamento do produto.

1.2 OBJETIVOS ESPECÍFICOS

- Implementar um cadastro de produtos vendidos em mercados com armazenamento local e em ambiente web compartilhado;
- Desenvolver o aplicativo para o sistema operacional Android;
- Fornecer um mecanismo de compartilhamento de preços e produtos.
- Utilizar uma aplicação para escaneamento e tratamento de código de barras de produtos;

2 REFERENCIAL TEÓRICO

O presente capítulo visa apresentar as diferentes técnicas e teorias referentes ao desenvolvimento e implementação do projeto. As técnicas aqui apresentadas foram analisadas no decorrer do desenvolvimento a fim de decidir quais seriam as diretrizes usadas na execução.

2.1 TECNOLOGIAS PARA IMPLEMENTAÇÃO

Como todo sistema computacional, é necessário que haja uma análise documental com o intuito de verificar quais são as tecnologias utilizar. Tecnologias como bancos de dados, linguagens de programação e sistemas operacionais são apenas alguns dos fatores a serem levados em consideração durante a fase de planejamento e modelagem do *software*.

2.1.1 Programação Móvel

O aplicativo tratado neste trabalho de conclusão de curso, além de ser um sistema computacional, é um sistema especificamente direcionado à dispositivos móveis *(mobile).* Segundo Lecheta (2015, p. 25), atualmente *smartphones* e *tablets* são objetos praticamente inseparáveis das pessoas.

Nesta seção, abordamos as diversas tecnologias que são comumente empregadas no desenvolvimento para aplicações móveis.

2.1.2 Sistema Operacional

Lecheta (2015, p. 25) indica que o sistema operacional *Android*, da *Open Handset Alliance* em conjunto com a *Google*®, além de ser um sucesso e líder de mercado, é indicado para o desenvolvimento móvel, pois gigantes do mercado são responsáveis pelo seu desenvolvimento, como a *Intel*, *Samsung*, *Motorola*, entre outras companhias. Cita ainda que o desenvolvimento para dispositivos móveis requer

uma alta abstração de *hardware* para que o sistema se adapte a vários dos dispositivos disponíveis no mercado.

Por outro lado, o *iOS,* sistema operacional de propriedade da *Apple,* também está em alta no mercado. Dentre as facilidades do desenvolvimento no sistema da multinacional, conforme Simões e Pereira (2014), está a *Itunes University*, que oferece uma gama de vídeos e serviços que auxiliam os desenvolvedores iniciantes na plataforma.

Por fim, recentemente foi lançado o *Windows 10* para celulares, sistema operacional de propriedade da *Microsoft*. A maior vantagem do *software* da *Microsoft* em comparação com os concorrentes é a integração. O sistema da *Microsoft* para celulares será apenas *Windows 10*, o mesmo que está presente em computadores, *tablets*, *Xbox One*, *HoloLens* e Internet das Coisas (SOUZA, 2015). Só é preciso programar o software de forma que ele seja um *Windows app*, um conjunto de aplicativos desenvolvidos para funcionar em qualquer dispositivo, seja ele um *desktop*, *smartphone*, *tablet* ou *Xbox One* (KURTZ, 2015).

Para o desenvolvimento do aplicativo foi utilizado o sistema operacional *Android* como plataforma alvo, pois este está presente na maioria dos aparelhos *smartphones* ao redor do mundo (SANTINO, 2016) e não há nenhum custo envolvido para a programação.

2.1.3 Plataforma de Desenvolvimento

São muitas as opções de plataformas de desenvolvimento e estas variam de acordo com alguns fatores como o sistema operacional alvo do aplicativo e as técnicas de desenvolvimento utilizadas. É possível utilizar linguagens nativas (C/C++ e Java, no caso do *Android*) ou linguagens *web* que se utilizam do navegador do sistema operacional para operar via internet (SILVA & SANTOS, 2014). Uma terceira opção também está disponível: mesclar ambos os tipos de linguagens no que é conhecido como desenvolvimento híbrido. Portanto, é necessário que sejam aplicados diversos parâmetros de validação para que se chegue a uma conclusão de qual é a melhor forma de se desenvolver o projeto.

As duas principais *IDE's* para desenvolvimento *Android* são: *Android Studio* e *Eclipse (LOPES & JUNIOR, 2016)*. Os temas para a seleção da aparência da interface

gráfica, a personalização de teclas de atalho, a função de auto completar código sem pressionar teclas de atalho, a integração nativa com sistemas de controle de versão e a criação de layouts por meio de arrastar e soltar componentes na tela são as principais vantagens do *Android Studio (CARVALHO, 2013)*.

O desenvolvimento nativo para os dispositivos da *Apple* requer a utilização da plataforma de desenvolvimento proprietária, conhecida como *Xcode*, que é mantido pela própria *Apple* e pode ser adquirida por meios oficiais através do *website* da empresa.

Caso seja abordada uma forma de desenvolvimento híbrida ou nativa, visando um sistema multiplataforma, uma opção é utilizar uma IDE de integração, que realiza a conversão do código para que este se adapte às diversas plataformas existentes. Como indicado no *website* oficial da *Xamarin inc* (2017), é possível utilizar a IDE proprietária *Visual Studio* a partir da versão 2015, em conjunto com o *software* de integração *Xamarin*, também proprietário, para desenvolver aplicativos móveis para *IOS*, *Windows 10* e *Android* utilizando uma base de código-fonte única, na linguagem C#.

2.1.3.1Distribuição e venda

A forma de distribuição é outra questão que depende da forma de desenvolvimento utilizada, o que ressalta a importância de uma fase de definições de requisitos e escolha das tecnologias, pois o impacto dos requisitos afeta todo o desenvolvimento do projeto.

No caso de um desenvolvimento que tenha como objetivo a criação de um aplicativo para o sistema operacional *Android*, é necessário que os desenvolvedores utilizem uma conta da *Google*, e tomando ciência dos termos do contrato que podem ser verificados no *website* oficial da empresa, é necessário um pagamento de $25,00. Com isto, a conta de desenvolvedor da *Google* estará habilitada a disponibilizar projetos na *Play Store*, plataforma de vendas e *downloads* da *Google* (*GOOGLE*, 2016).

A *Apple*, por sua vez, conta com seu próprio meio de distribuição, a *Apple Store*. Como pode ser verificado no *website* oficial da empresa, é necessário que o

desenvolvedor tenha uma conta no programa de desenvolvimento da *Apple*, cuja renovação anual custa $99,00 (*APPLE*, 2016).

A *Microsoft*, utiliza uma loja única, a *Windows Store,* contendo os *Windows apps*, ou seja, se o aplicativo for programado para rodar no sistema operacional *Windows* 10, terá que utilizar essa loja. Existe uma taxa única de $19,00 para pessoa física e $99,00 para pessoa jurídica (*WINDOWS STORE DEVELOPER*, 2017).

Dado que no momento o projeto se dá em fase experimental e que não há a necessidade de lançamento multiplataforma, foi escolhida a plataforma Android para a publicação devido às vantagens citadas anteriormente relativas ao custo de inscrição na loja virtual e a possibilidade do desenvolvimento em uma IDE gratuita.

2.1.4 Leitura de Código de Barras

Todos os produtos vendidos em supermercado possuem um código de barras, segundo norma da GS1 Brasil (ASSOCIAÇÃO BRASILEIRA DE AUTOMAÇÃO, 2016).

Atualmente existem vários tipos de códigos de barras, tais como:

- EAN/UPC, utilizado para leitura no PDV (ponto de venda);
- GS1 DataBar, utilizado no setor de frutas, verduras e legumes;
- GS1-128, utilizado na gestão de logística e de rastreabilidade;
- ITF14, que é um código de barras desenvolvido para codificar apenas GTINs (Chave global) atribuído para a identificação de item comercial/serviço que precisa ser precificado, encomendado, faturado em qualquer ponto da cadeia de suprimentos;
- GS1 DataMatrix, utilizado em hospitais pelo seu tamanho reduzido;
- GS1 QR Code, que é basicamente a evolução do código de barras, pode conter 100 vezes mais informação (GS1 BRASIL, 2016).

Com base nesses padrões de códigos de barras, temos duas opções, por *intent* e por API (*Application Programming Interface* ou, traduzindo para o português, Interface de Programação de Aplicativos). Segundo Lecheta (2015) a *intent* está presente em todos os lugares e representa uma mensagem da aplicação para o sistema operacional, solicitando que algo seja realizado. Desta forma é possível solicitar para que o SO abra uma aplicação especializada em leitura de código de

barras e retorne o resultado da leitura para o uso posterior. No caso de se utilizar uma API, o programa e/ou biblioteca fica integrado a aplicação, assim não é necessário utilizar um *software* terceiro para efetuar a leitura do código de barras.

O *QR Code*, para o nosso projeto, se mostra de extrema importância pois a Nota fiscal de Consumidor Eletrônica faz uso deste recurso para a consulta ao site da receita.

2.1.5 Aplicação *Web*

Visto que o projeto envolve, além de um aplicativo para dispositivos móveis, também um *website*, é necessário analisar os requisitos do desenvolvimento *web*. Para este fim, aborda-se nesta seção as ferramentas e tecnologias disponíveis para este tipo de desenvolvimento.

2.1.5.1Ferramentas e linguagem

Aplicações *web* envolvem *sites web* ou sistemas *web* (CONALLEM, 2000). Para construir um *website* dinâmico existem muitas tecnologias, como o HTML (*Hypertext Markup Language*), uma linguagem de marcação que segundo Rodrigues(2010), é a mais utilizada na estruturação de páginas *web* atualmente. Sua sintaxe é bastante simples e, assim como o XML (*Extensible Markup Language*) é baseada em *tags* que representam os diversos elementos de uma página, como imagens e links.

Para que a aplicação *web* execute comandos é preciso a utilização de CGI's (*Common Gateway Interface*), que segundo Winckler e Pimenta (2002) são programas que podem ser escritos em qualquer linguagem de programação, hospedados no servidor e executados via HTTP, ASP e PHP.

PHP é uma das linguagens de programação mais utilizadas na *web* para a criação de páginas dinâmicas (NIEDERAUER, 2014). Já no ambiente ASP, desenvolvido pela *Microsoft*, é possível combinar várias tecnologias de desenvolvimento *web* de forma combinada para a criação de aplicações profissionais (JUNIOR, 2007).

O CSS (*Cascade Style Sheets*) permite fazer com que a apresentação de páginas *web* seja determinada por um conjunto de especificações de formatação de páginas, preferências tipográficas e outras características do dispositivo do cliente, de forma a garantir a continuidade visual do site (WINCKLER E PIMENTA, 2002).

Para maximizar as possibilidades de programação de uma aplicação *web* o *Javascript é uma opção* bastante completa, visto que pode funcionar em conjunto com o HTML como *script*, ou pode ser usado ainda de forma independente, utilizando *frameworks* como o *Angular 2* (PINHO, 2017). Quando um destes arquivos é carregado, o próprio navegador interpreta o *script* e realiza as operações necessárias. A linguagem *Javascript* (SCHIMID, 2012) é uma linguagem do tipo *Client Side*, ou seja, ela é executada no computador do usuário. Há ainda os *Applets* e os *Plug-ins*, o primeiro são pequenos programas *Java* e o segundo são programas que integram o navegador (WINCKLER & PIMENTA, 2002).

2.1.5.2 Banco e *web service*

Para o armazenamento dos dados existem atualmente várias soluções. O *MySQL*, um dos SGBD's (Sistema Gerenciador de Banco de Dados) mais utilizados no mundo, tornou-se uma solução viável e de missão crítica para a administração de bases de dados. Este SGBD incorpora muitas funções necessárias para outros ambientes, mantendo uma alta velocidade e ainda apresentando configurações de permissões do sistema (GILFILLAN, 2003).

Quando comparado ao seu concorrente *Open Source, o PostgreSQL,* é notável a superioridade do *MySql* em relação à popularidade de uso. A razão por muito tempo pôde ser explicada pelo *MySql* ser mais simples no quesito de usabilidade do que o *PostgreSQL* e pelo fato do *PostgreSQL* ser mais robusto e ter mais funções, o que exigia um conhecimento técnico mais profundo por parte do desenvolvedor.

No entanto, mudanças significativas foram implementadas visando melhorar a usabilidade do *PostgreSQL* e essa diferença em relação ao *MySql* já não é mais tão notável (AMARAL, LIZARDO, SOUZA, 2011). Desta forma, o único requisito que impera na escolha de um banco de dados é que este seja relacional, para uma melhor integração com o banco interno do Android.

Para a criação de *web services* não faltam possibilidades, ainda mais com a emergência destes perante a popularização dos serviços *online*. Uma das possibilidades conhecidas é o SOAP que antigamente significava *Simple Object Access Protocol*, e que passou a ser chamado apenas pelo seu acrônimo, por não usar objetos. Ele é um protocolo de transferência de mensagens em formato XML para uso em ambientes distribuídos. Este padrão funciona como uma espécie de *framework,* permitindo a interoperabilidade entre diversas plataformas com mensagens personalizadas. (SUDA, 2003)

Outra abordagem para a criação de *web services é* o modelo *REST,* outro protocolo de comunicação, baseado no protocolo de comunicação HTTP. Porém ele não impõe restrições ao formato da mensagem, apenas no comportamento dos componentes envolvidos (CARLOS & RAMALHO, 2004).

Para que se possa enviar os dados para a base de dados *online*, através de *web service* próprio, é necessário que os dados informados pelo usuário fiquem armazenados no próprio dispositivo no qual o "Mercadinho" estará instalado, pelo menos temporariamente, até que os registros sejam sincronizados ou excluídos, constituindo assim uma forma de armazenamento intermediária.

Para auxiliar nesta tarefa, o *Android* conta com uma implementação do banco de dados *SQLite* embutida no próprio sistema operacional. Outras formas de persistência estão disponíveis no sistema, como *Shared Preferences* (Preferências compartilhadas), arquivos na memória interna e cartões de memória, e ainda *Content Providers*. No entanto o *SQLite* se adequa ao projeto pois segundo Glauber(2015), o banco de dados *SQLite* oferece facilidades como: suporte a transações; independência do sistema operacional, o que facilita o reuso de código ao portar o aplicativo para outras plataformas; acesso direto ao sistema de arquivos e ainda, o fato de não ter nenhuma configuração necessária por ser embutido no *Android*, por essas razões foi utilizado o *SQLite* para gerenciar o banco no *Android*.

2.1.5.3 Hospedagem

Como o sistema tem suporte à várias funções *online*, faz-se necessário obter um endereço na internet. Para a redução de custos de desenvolvimento, foi utilizado um serviço de hospedagem gratuito. Nesse modelo de trabalho, tem-se um *Host,*

empresa que oferece infraestrutura de servidores e suporte para que possa ser contratado, ficando os contratantes responsáveis apenas pela configuração e manuseio do servidor contratado.

Neste escopo há empresas que atuam apenas como hospedagem de sites e empresas que atuam como provedoras de serviço em nuvem (*cloud computing*). Neste modelo, é possível contratar um ou mais servidores para rodar os mais diversos serviços. Para este projeto foram utilizadas apenas às empresas provedoras de serviços em nuvem, visto que o trabalho pode ser escalonado futuramente, diminuindo o risco de alterações futuras. Duas das empresas mais comumente usadas como provedoras. de serviço são a AWS e o Heroku (ANDRUSHKO, 2016).

Quando comparado ao Heroku, a AWS apresenta uma maior robustez e escalabilidade sendo ótima para grandes projetos. O Heroku por sua vez, tem como maior premissa a facilidade de configuração e *setup* dos serviços, sendo mais indicado para projetos menores (LEICHER, 2016). Como visto no Quadro 1 ambas as opções oferecem suporte à diversas linguagens de programação, ponto relevante a ser considerado no desenvolvimento.

Quadro 1- Comparativo entre os requisitos técnicos da AWS e do Heroku.

	Amazon Web Services	**Heroku**
Dono	Amazon.com	Salesforce.com
Hospedado em	Servidores Proprietários	Data centers da Amazon
Tipo de serviço	IaaS (Amazon EC2) PaaS (Amazon Elastic Beanstalk)	PaaS (Amazon Elastic Beanstalk)
Linguagens	.NET, Ruby, NodeJS, Go, Docker, PHP, Python	Node.js, Java, Ruby, PHP, Python, Go, Scala, Clojure
Regiões geográficas	EUA, Canadá, América do Sul, Europa, Ásia-Pacífico, China	Europa, EUA, Austrália, Japão
Funcionalidades chave	- Várias opções de implantação e habilidade de reverter para versões anteriores; - Controle total sobre a saúde do aplicativo ou mudanças de servidor com notificação por e-mail; - Reinício rápido de todos os servidores de aplicação com um único comando; - Tráfico de carga adaptativo entre instancias automaticamente; - Dimensionamento automático de suas aplicações web baseado em suas necessidades ou em condições definidas;	- Ambiente de tempo de execução totalmente gerenciável com sistema de contêineres inteligentes (dynos); - Dimensionamento manual vertical e horizontal; - Habilidade de reverter sua base de dados ou código rapidamente; - Sistema de monitoramento de aplicativos embutido para metrificação, como tempo de resposta, taxa de transferência, memória, etc. - Integração consistente com o Github;
Categorias de clientes	*Startups*, empresas médias, empresas grandes;	*Freelancers*, *Startups*, empresas médias, empresas grandes (*Heroku Enterprise*);
Usado por	3M Health Information Systems, Grupo BMW, Airbnb, Coursera, Atlassian, Discovery Communication e outros.	Product Hunt, Macy's, Toyota, Citrix, Westfield, Carbon Five, Lutron, Yesware.

Fonte: KOROTYA (2017) (Adaptado)

Foi utilizado a plataforma *cloud Heroku*, a qual disponibiliza tanto domínio como SGBD gratuitamente. O SGBD em questão é o *PostgreSQL*, desta forma não haverá custos envolvendo a hospedagem da aplicação *web*.

2.1.6 Frameworks e API's

Com a baixa dos preços e aumento da popularidade de *smartphones,* muitos *frameworks* foram criados para facilitar a programação para este tipo de dispositivo. Os *frameworks* são basicamente estruturas prontas de código otimizadas com relação ao reuso. Assim é possível realizar uma implementação com nenhum ou pouco código customizado (ZANETTE, 2017). Com isso em mente, foi necessário realizar uma pesquisa por esses *frameworks* a fim de facilitar o processo de codificação do aplicativo proposto.

2.1.6.1Frameworks para banco de dados

A utilização de um framework, minimiza o esforço de desenvolvimento de novas aplicações, pois já contém a definição de arquitetura gerada a partir dele bem como, tem predefinido o fluxo de controle da aplicação (FRANÇA, 2006). Diante disso foram analisadas algumas opções disponíveis: *GreenDAO*, *ORMLite* e *ActiveAndroid*.

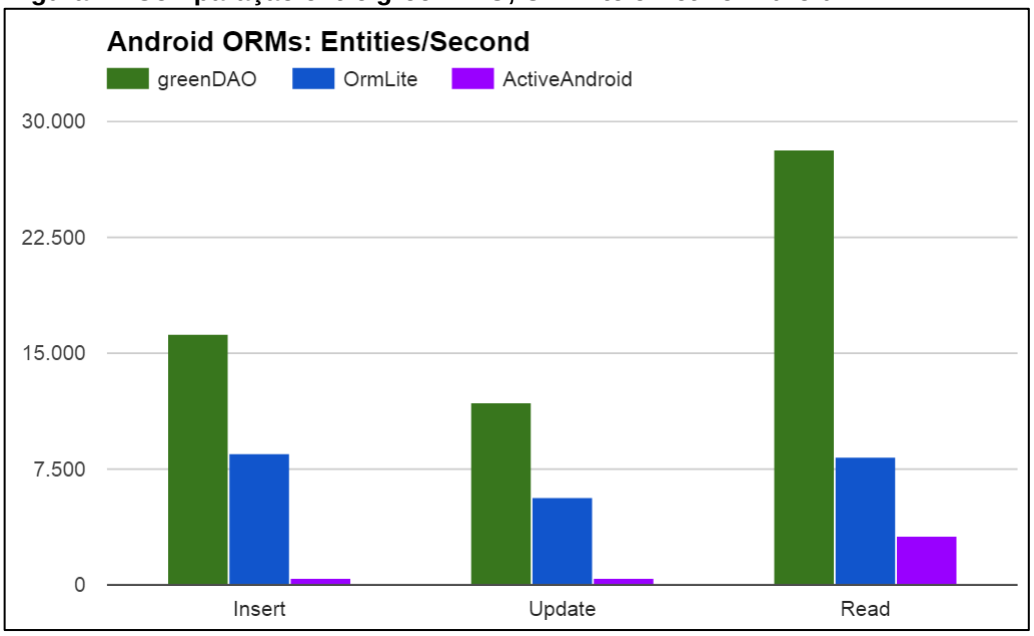
O *GreenDAO* apresenta uma forte característica que é ser extremamente rápido. Em testes apontados pela própria *GreenRobot*, principal contribuidora do *GreenDAO*, cujo desenvolvimento é *OpenSource*, a API se mostrou a mais rápida na realização de transações (GREENROBOT, 2016).

O *ORMLite*, uma alternativa ao *GreenDAO*, se caracteriza pelo alto nível de abstração de transações. Utilizando-o é possível realizar consultas (*queries)* simples e complexas sem que seja necessário que o programador se lembre da sintaxe da SQL. (WATSON, 2014)

A solução *ActiveAndroid*, além de contar com transações simplificadas, permite acesso a módulos alternativos de acesso a memória, como o *ContentProvider* (PARDO, 2010).

Na Figura 1 é apresentado um comparativo entre os *ORMs greenDAO*, *OrmLite* e *ActiveAndroid*, onde foi comparado a quantidade de registros inseridos, alterados e lidos por segundo.

Figura 1 - Comparação entre greenDAO, OrmLite e ActiveAndroid



Fonte: GREENROBOT (2016)

Para o nosso projeto foi utilizado o *framework GreenDao*, devido a sua velocidade e a facilidade em executar os comandos no banco. Além de gerar todo o esquema do banco de dados, é possível fazer transações com somente uma linha de código, o que da forma tradicional demandaria muitas linhas de código e tempo para ser programadas.

2.1.6.2 Frameworks para PHP

A utilização de *frameworks* na programação em PHP é muito bem-vinda no cenário atual, pois acelera o processo de desenvolvimento, ajuda a desenvolver um código bem organizado e utiliza o padrão MVC (*Model - View - Control*). Existem inúmeros *frameworks* no mercado, dentre eles alguns se destacam.

O *Symfony 2 PHP framework* possui vários componentes que são bibliotecas reutilizáveis, dessa forma o desenvolvedor pode utilizar somente o componente que lhe interessa, existem componentes específicos para uma determinada tarefa como

por exemplo: criação de formulários, configuração de objetos, roteamento, autenticação e muito mais (SYMFONY, 2018).

Outro *framework* muito conhecido e utilizado é o *CodeIgniter.* O grande diferencial do *CodeIgniter* é que ele é um *framework* de utilização bastante simples, o que aumenta muito a curva de aprendizado da equipe. Outro ponto, é sua compatibilidade com muitos servidores diferentes (DE LEONE, 2017).

Tem-se também o *framework Laravel*, lançado em 2011, se destaca por agilizar o desenvolvimento, facilitando trabalhos comuns à maioria dos projetos, como, sessão, *cache* e autenticação, conseguindo ainda manter uma arquitetura exclusiva, que permite aos desenvolvedores especificarem sua própria infraestrutura (VERMA, 2016).

O *CodeIgniter* facilita muito a padronização do website com o MVC, embora não seja uma exigência, esse padrão é possível tranquilamente implementá-lo no *CodeIgniter*, além de possuir uma gama muito grande de plugins que facilitam operações específicas do *web site*, por esses motivos foi utilizado como *framework* para *PHP* do nosso projeto.

2.1.6.3 Frameworks para web service REST com PHP

Para a construção da plataforma *online* que manterá os dados enviados pelos usuários armazenados, será necessária a construção de uma interface simples para o envio, recepção e tratamento dos dados, quando receber requisições dos clientes. Há muitas soluções que prometem uma rápida implementação do modelo *REST* como analisado brevemente adiante.

Dentre os mais populares *frameworks* para este tipo de aplicação está o *Fat-Free Framework (*Framework livre de gordura, em tradução literal*)*, que como indicado pelo nome apresenta sua estrutura separada por pequenos módulos, mantendo a implementação compacta. Apresenta boa velocidade, mas sofre com duplicação de código (GAJOTRES, 2015).

Outra opção conhecida é o *Slim Framework*. Este é um dos mais rápidos disponíveis e apresenta uma gama bastante completa de funcionalidades. Um grande ponto positivo desta opção é a grande comunidade de desenvolvedores, que

oferecem, ajuda com implementação, dicas e documentação objetiva. (GAJOTRES, 2015).

Optou-se pelo *Slim framework* pois de forma rápida, fácil e em poucos minutos já é possível disponibilizar os serviços de *GET* e *POST* do *webservice*.

2.2 MÉTODOS DE DESENVOLVIMENTO DE SISTEMAS MÓVEIS E WEB

Ao longo dos anos, vários métodos de desenvolvimento de produtos foram apresentados. Entre eles, existem os chamados métodos ágeis (AMBLER, 2002). Os métodos ágeis têm muita relevância no desenvolvimento de *software* moderno, sendo empregados em empresas de grande porte como *Google* e *Facebook*. Nesta seção são apresentados os métodos Kanban, Extreme Programming e Scrum, concluindo com uma breve comparação entre eles com a intenção de determinar a utilização destes no desenvolvimento deste projeto.

2.2.1 *Kanban* (Melhoria Contínua)

Kanban é o modelo de processo criado e usado por Taiichi Ohno para revolucionar o modelo de processo industrial quando trabalhava como engenheiro industrial para a *Toyota* (SUGIMORI et al., 2007).

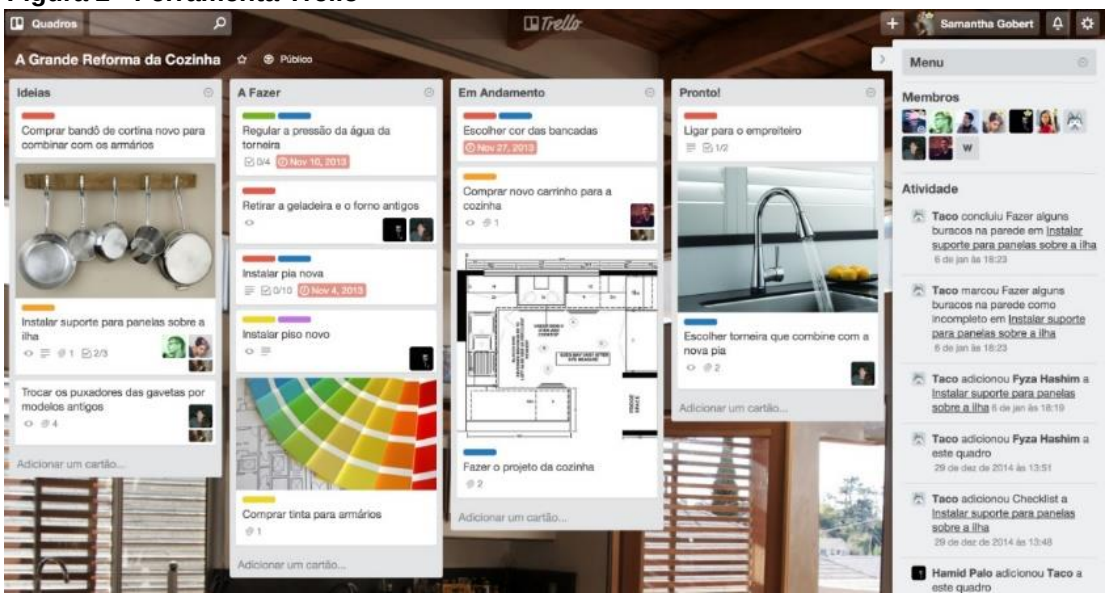
Este modelo é bastante conhecido por sua simplicidade, e costuma ser usado, muitas vezes na forma de quadro, descartando o uso de computadores, tornando a visualização e discussão dos afazeres mais fácil. Em sua forma de aplicação um quadro é dividido em setores que representam partes de um processo de produção, e cartas que representam determinado produto são anexadas à estas seções. Desta forma, é possível notar rapidamente, de forma bastante visual, em que parte do processo de produção uma demanda específica se encontra (ANDERSON, 2013).

A simplicidade fornecida pelo modelo tem vantagens como a redução do custo do processo de informação, rápida e precisa aquisição de fatos e a redução do excesso na demanda de recursos, que costuma atingir níveis altos de desvio conforme o tamanho do escopo do projeto (SUGIMORI et al., 2007).

O Kanban funciona dividindo as etapas do processo em um quadro, criando um modelo visual do fluxo. A ideia é que a informação visual seja processada mais

rapidamente pelo cérebro, que o faz 60 mil vezes mais rápido que informação textual (LEANKIT, 2018). Uma ferramenta *online* e gratuita é o *Trello (www.trello.com)*, é uma maneira fácil e flexível de gerenciar qualquer tipo de projeto, ele organiza vários cartões com as tarefas a fazer, em andamento e concluídas (ATLASSIAN, 2018), na Figura 2 é possível ver um exemplo do gerenciamento que o *Trello* permite.

Figura 2 - Ferramenta Trello



Fonte:TRELLO (2018)

2.2.2 *Extreme Programming* (XP)

É um método ágil focado no desenvolvimento de *softwares* de forma eficiente, de baixo risco, previsível, científico e divertido. Para atingir uma alta qualidade no produto final, o método XP preza o cliente sempre à disposição, reuniões de planejamento, reuniões diárias, integração dos módulos desenvolvidos, entregas de pequenos produtos para o cliente avaliar, entre outros (BECK, 2005).

O método XP é uma disciplina aplicada à programação, pois o XP contempla um conjunto de valores, princípios e práticas que devem ser aplicadas para que se possa dizer que um projeto de desenvolvimento está realmente aplicando o XP (BECK, 2005).

O modelo divide o processo de desenvolvimento em 5 etapas: Planejamento, Gerenciamento, Design, Codificação e Testes. Cada uma destas etapas tem regras que podem ser aplicadas (DON WELLS, 2013)

São estas etapas:

a) Planejamento - Escrita das histórias de usuário, planejamento de *release, releases* frequentes e pequenas, projeto dividido em iterações, planejamento de iterações no começo de cada iteração;

b) Gerência: Espaço de trabalho dedicado ao time, ritmo sustentável, reunião diária em pé, medição da velocidade do projeto, pessoas em movimento, mudar o próprio XP quando este não funcionar adequadamente a projeto.

c) Design: Simplicidade, uso de metáforas em explicações, sem funcionalidades extras no começo do projeto, refatoração do código sempre que possível;

d) Codificação: Ciente sempre disponível, código escrito seguindo padrões pré-estabelecidos, codificar os testes unitários primeiro, código de produção criado em pares;

e) Testes: Todo código deve possuir testes unitários, todo código deve passar em todos os testes unitários criados antes do lançamento, quando um bug é encontrado testes são criados, testes de aceitação são rodados e os resultados são publicados.

2.2.3 Scrum

O *Scrum*, é um *framework* de processo ágil bastante flexível, foi criado por *Ken Schwaber* e *Jeff Sutherland* para ser uma maneira mais confiável, rápida e efetiva de criar software na indústria tecnológica (SUTHERLAND, 2014).

No *Scrum*, desde o início do desenvolvimento já são definidas as funcionalidades do produto. Cada funcionalidade se torna uma *Sprint*, um período limitado de desenvolvimento. Ao término de cada *Sprint* é realizada uma reunião para avaliar o resultado desta. Também é apresentado ao cliente o resultado da *Sprint* para que se tenha ciência se o produto está realmente tomando o rumo desejado e se as especificações e requisitos não mudaram desde o início do desenvolvimento. Assim, o *Scrum tenta* lidar com as incertezas provenientes do ambiente de desenvolvimento de *software*, considerando o objetivo dos *stakeholders* (interessados no projeto), sincronizando-os com os desenvolvedores.

2.2.4 Comparativo

O *Kanban* é recomendado para projetos que necessitem de mudanças constantes no *backlog*, limitando a quantidade de tarefas em execução. *Scrum* e *XP* tem um maior nível de sofisticação e são recomendados para times que necessitem de um ganho em comunicação da equipe. Importante também citar que os 3 aderem ao manifesto ágil, e suas regras pré-estabelecidas podem ser alteradas dependendo da demanda do projeto *(BOWES, 2015).*

Como metodologia de desenvolvimento, optou-se por *Scrum* usado juntamente com o *Kanban*, onde ao final de um ciclo é gerada uma nova função ou produto parcial passível de utilização para ser analisado, e ainda se abre espaço para discutir sobre os problemas encontrados e sobre o andamento das metas. Para cada ciclo foram estabelecidos pequenos objetivos que foram divididos entre a dupla desenvolvedora do projeto. Para controle do andamento do ciclo foi utilizada a ferramenta *Trello*, utilizada para controle do *Kanban*.

3 FLUXO DE UTILIZAÇÃO

Neste capítulo é apresentada a concepção do *software* desenvolvido, mostrando as funções do programa e o seu funcionamento.

Os seguintes passos foram considerados na execução do projeto:

a) Implementação da base de dados *online* e *offline*.
b) Implementação do cadastro de produtos incluindo código de barras campo para descrição, quantidade, preço, mercado e data da compra.
c) Pesquisa e implementação da função *scanner* utilizando a câmera do *smartphone*.

3.1 FLUXO DE USO DO WEBSITE

Após realizar acesso ao *website* do Mercadinho através do endereço web “https://mercadinho.herokuapp.com/”, a página inicial apresenta as seguintes funções: Buscar Produto, Cadastrar Novo Usuário e Realizar *Login*.

Caso o usuário não deseje realizar o *login* ele pode acessar a busca de produtos, função que permite que ele pesquise por um produto pelo seu nome ou código de barras e veja qual o valor deste produto em diferentes mercados com base nas últimas informações coletadas pelos outros usuários. Para isso, ainda na página inicial, basta que o usuário utilize a barra de pesquisa posicionada no centro da página para informar a cidade e os dados do produto que deseja buscar, depois clique no botão “Buscar”. O sistema então apresentará uma lista com todos os resultados possíveis baseado nos dados informados.

Em seguida, o usuário realiza um clique no nome do produto que deseja visualizar, como resultado o *site* apresenta uma página com as últimas 10 compras registradas. Nesta tela, o usuário pode filtrar os resultados mudando a quantidade de registros exibidos, mercado e data da compra.

Caso o usuário, na página inicial, deseja fazer o cadastro no sistema, ele fará um clique em “Cadastrar Novo Usuário”. O sistema apresentará uma página para que o usuário informe os seguintes dados: nome, e-mail e senha, ao finalizar basta clicar sobre o botão “Cadastrar”, o usuário é então redirecionado para uma página de agradecimento e informações sobre o funcionamento do sistema.

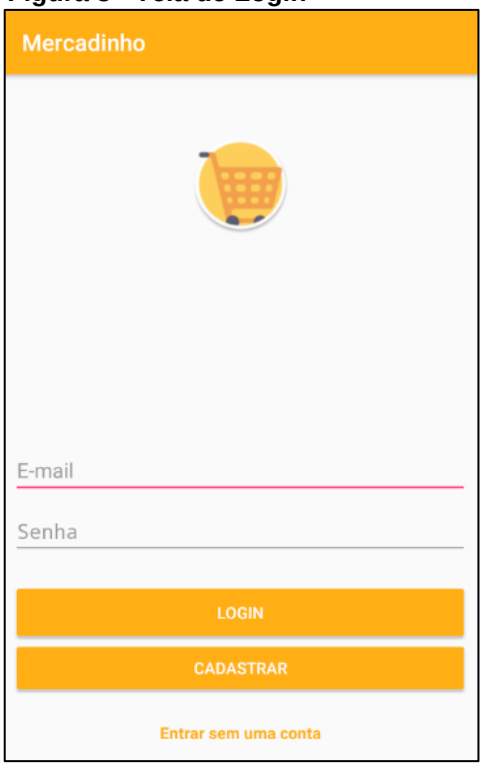
No caso de o usuário já ter um cadastro, ele pode realizar o *login* clicando em "*Login*". O sistema apresenta então uma página requisitando e-mail e senha. Após o usuário informar estes dados e clicar sobre o botão "Login" ele é redirecionado novamente à página inicial, que agora apresenta a opção de lista de compras, representada pelo menu de acesso "Minhas Listas". Ao clicar neste menu, o *website* apresenta uma listagem de todas as listas cadastradas pelo usuário. Nesta página, o usuário *logado* tem a opção de criar, editar, excluir, duplicar ou apenas visualizar uma lista de compras. Para isso, cada registro da listagem apresenta ícones referentes a "Alterar", "Excluir" e "Duplicar".

Para visualizar todos os produtos de uma lista, basta clicar sobre a descrição dela e o sistema fará o redirecionamento para a listagem destes. Como citado, também é possível incluir uma nova lista através do botão com o ícone de "+" e informando a descrição da nova lista. Todas as modificações que ocorrem nas listas pelo *website*, são sincronizadas com *smartphone Android* do usuário no momento que o usuário clica no botão sincronizar listas no aplicativo. Outro ponto importante é que o usuário *logado* no sistema também tem à disposição a função de busca de produtos, detalhada anteriormente.

3.2 FLUXO DE USO DO APLICATIVO MÓVEL

Ao executar o aplicativo é apresentado ao usuário a tela de *login*. Nesta tela o usuário pode escolher realizar o *login* ou cadastrar um usuário, como demonstrado na Figura 3. Para realizar o *login*, basta inserir o nome de usuário e senha previamente cadastrados, caso o usuário deseje se cadastrar, é possível fazer o cadastro através do botão "Cadastrar", onde é solicitado: Usuário, senha e e-mail.

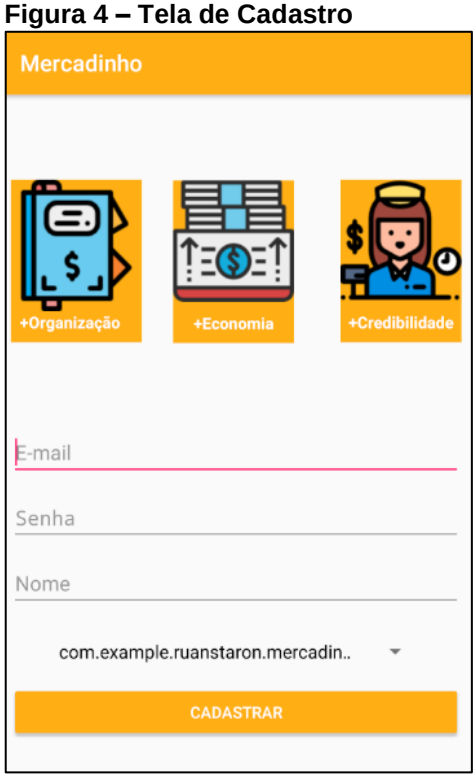
Figura 3 - Tela de Login



Fonte: Autoria Própria

A tela de cadastro, por sua vez, apresenta apenas os campos necessários para as funcionalidades correntes do aplicativo, como observado na Figura 4.

Figura 4 – Tela de Cadastro

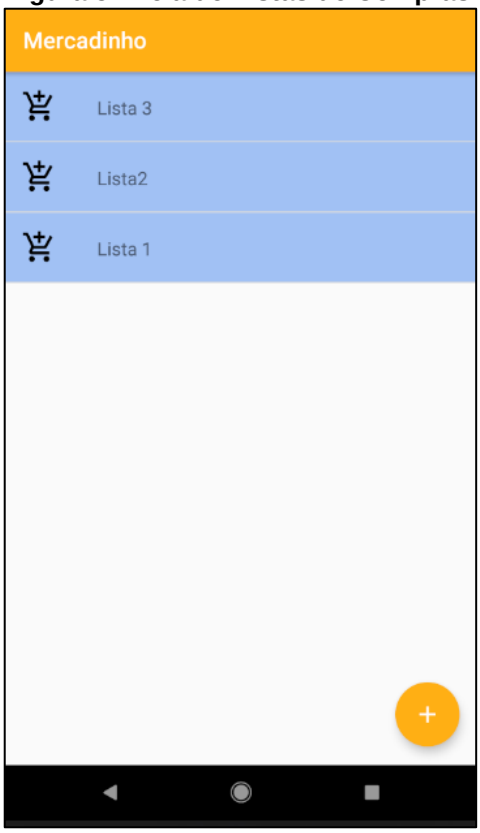


Fonte: Autoria Própria

Na tela de Lista de compras é possível cadastrar, editar, excluir, duplicar, reabrir, conferir e sincronizar as listas de compras que estão no *website*. Para cadastrar uma nova lista de compras basta clicar no botão flutuante com o símbolo de “+”. Para excluir uma lista de compra, basta um toque longo na lista desejada e selecionar o item excluir no menu que é apresentado, para editar a descrição de uma lista de compras basta dar um toque longo e selecionar a opção editar no menu que irá se abrir. É possível duplicar uma lista de compras com um toque longo em cima da lista desejada e clicando na opção duplicar no menu que será exibido.

No canto superior direito existe um botão com símbolo de “Sincronização” que só estará habilitado se o usuário estiver logado. Através desse botão é possível sincronizar suas listas de compras com as que estão no *website*. A Figura 5 apresenta a tela de lista de compras.

Figura 5 - Tela de Listas de Compras



Fonte: Autoria Própria

Para facilitar a visualização da situação de cada lista, utilizamos um esquema de cores, onde se a lista estiver verde significa que aquela lista já foi toda comprada e conferida com o cupom fiscal, vermelha se existe divergência nos valores entre a

lista e o cupom fiscal ou que aquela lista ainda não foi conferida, e azul se a lista está em processo de compra. Caso o usuário queira reabrir uma lista que está verde, basta fazer um toque longo na lista desejada e clicar no item “Reabrir” no menu que aparecerá. No caso de lista vermelha, com um toque longo é possível acessar o item “Corrigir” que é apresentado no menu, se o problema ocorrer por haver valores divergentes entre o cupom fiscal e as compras da lista, são apresentados os produtos que estão com valores divergentes. Se a lista ainda não foi conferida é apresentada uma interface para o escaneamento do cupom fiscal. Para acessar uma lista, basta um toque no registro desejado.

Na tela dos itens da lista de compra o usuário pode adicionar os produtos com quantidades e valores que ele deseja, a fim de montar a sua lista antes de ir ao mercado, para adicionar um produto na lista de compra a única informação obrigatória é a descrição do produto, caso o usuário opte por escanear o código de barras do produto e esse produto esteja cadastrado na base de dados, o campo descrição é preenchido automaticamente, se o código de barras não tiver sido cadastrado basta escrever a descrição do produto na caixa de diálogo que será exibida.

Ao lado de todos os produtos existe uma caixa de seleção que indica que o usuário colocou aquele produto no carrinho (produto comprado). Só é possível marcar esta caixa de seleção com os campos de descrição, quantidade e valor preenchidos, no momento que essa caixa de seleção é marcada, a interface de escaneamento do código de barras é apresentada. É possível excluir ou editar os produtos já adicionados na lista, basta efetuar um toque longo no produto desejado e escolher a opção “excluir” ou “editar” no menu apresentado.

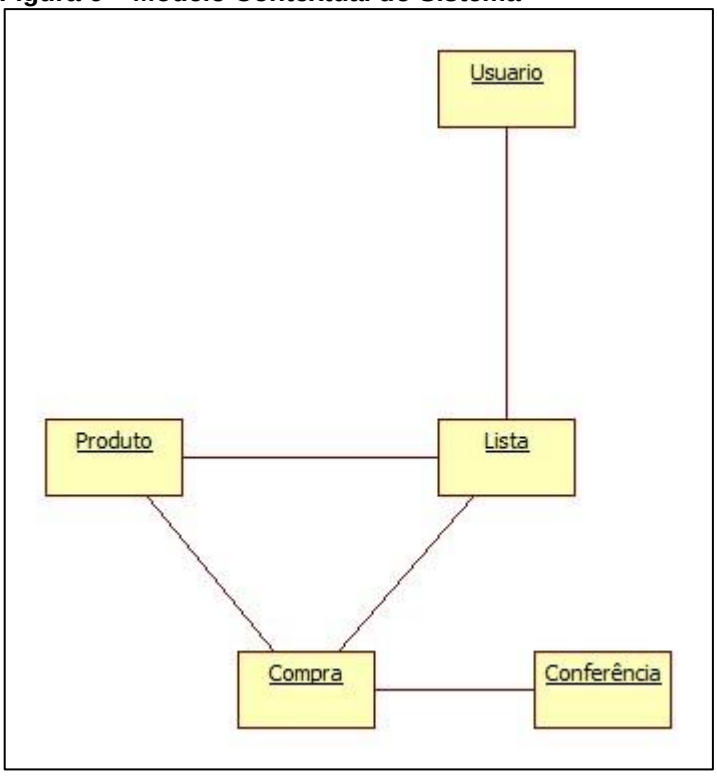
Para melhor visualização os produtos que já foram “comprados” ficam com o fundo verde. No canto direito inferior existe o botão “Finalizar” que verifica se toda a lista foi comprada e questiona o usuário se quer conferir com base no cupom fiscal.

Na tela de conferência, primeiramente é apresentada a interface de escaneamento para que o usuário escaneie o QRcode do cupom fiscal, logo em seguida o app faz o cruzamento com as informações de valores obtidas do cupom fiscal com os valores que o usuário cadastrou, é apresentado ao usuário todos os produtos com divergências para que o usuário corrija os problemas se necessário.

3.3 VISÃO GERAL

Para facilitar a usabilidade do sistema, optou-se por automatizar a sincronização dos dados locais do usuário com o *web service*. No momento em que o usuário escaneia o *QRcode* do cupom fiscal, é realizado o *download* os seguintes dados: o mercado e a data da compra, o código de barras, a descrição, quantidade e o valor dos produtos, e então estes dados são armazenados no banco local. No momento que o aplicativo é aberto, essas informações são enviadas para o *web service* e, em seguida, a base local é atualizada, assim o usuário não precisa se preocupar em atualizar a base local e enviar as informações para o *web service*. A Figura 6 apresenta um diagrama de classes adaptado para apresentar um modelo contextual do sistema, apresentando as conexões das principais partes envolvidas.

Figura 6 – Modelo Contextual do Sistema

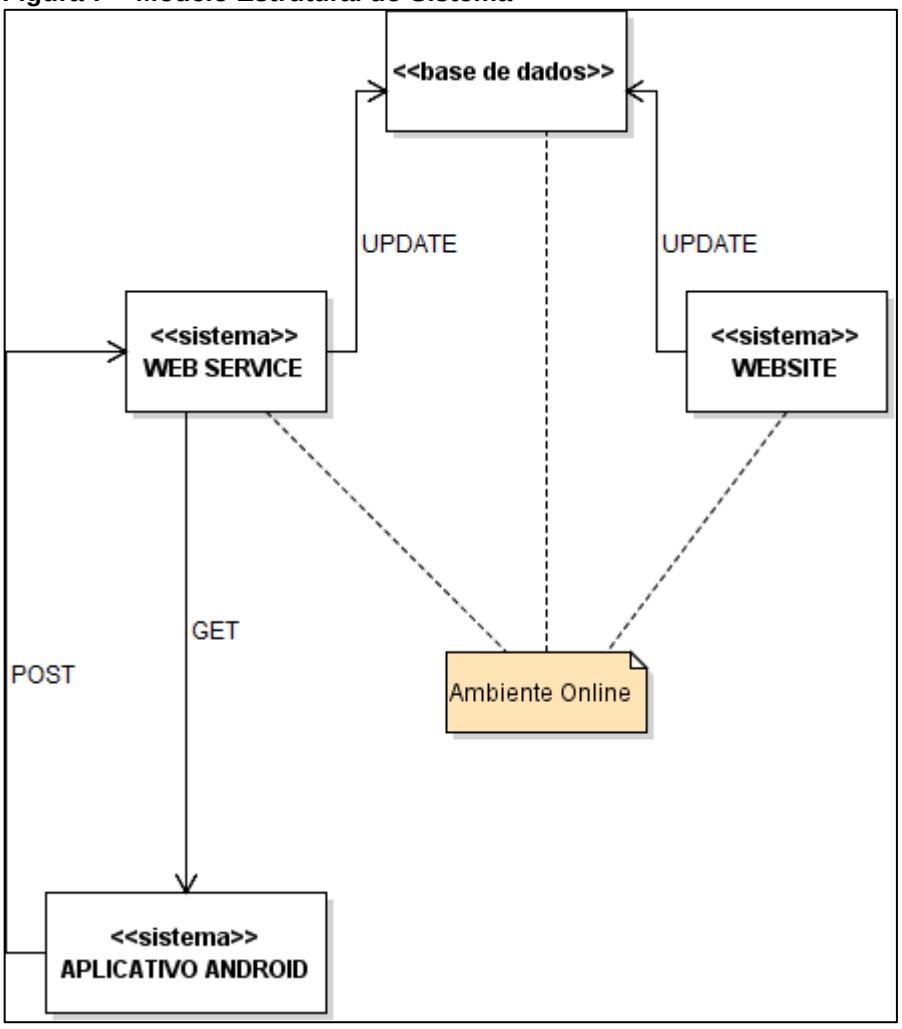


Fonte: Autoria Própria

4 MODELO ESTRUTURAL DO SISTEMA

Como dito anteriormente, para a completude do sistema, é de vital importância a existência de 3 partes fundamentais: o aplicativo móvel, o *website* e o *web service*, como é indicado na Figura 7, representados por um diagrama de classes adaptado para representar o modelo estrutural do sistema.

Figura 7 – Modelo Estrutural do Sistema



Fonte: Autoria Própria

4.1 CASOS DE USO DO APLICATIVO ANDROID

O aplicativo implementa os seguintes casos de uso:

a) Criar conta - cria uma nova conta de usuário.
b) Realizar *Login* - inicia uma sessão e carrega os dados de usuário do sistema.
c) Realizar *Logout* - encerra uma sessão, impossibilitando o acesso aos dados a não ser que seja realizado um novo login.
d) Criar Lista - cria lista de compras.
e) Editar Lista - edita os dados de uma lista de compras.
f) Excluir Lista - exclui uma lista de compras.
g) Duplicar Lista - duplica uma lista de compras.
h) Conferir Lista - realiza a conferência dos valores dos produtos de uma lista através do escaneamento de um cupom fiscal eletrônico. Encerrando a lista de compras.
i) Reabrir Lista - exclui a conferência e as compras de uma lista para permitir que esses processos sejam realizados utilizando-se da mesma lista.
j) Adicionar Produto - cadastra um novo produto em uma lista selecionada.
k) Editar Produto - altera os dados de um produto em uma lista selecionada.
l) Excluir Produto - exclui um produto em uma lista selecionada.
m) Comprar produto - define que aquele produto foi comprado.
n) Sincronizar Listas - sincroniza os dados de listas da conta de usuário com o *web service*.
o) Validar Login - *web service* recebe os dados de *login* e verifica se estão corretos, retornando o resultado da verificação para o cliente.
p) Enviar Listas - *web Service* envia as listas que estão em ambiente online para a base local do aplicativo do usuário.
q) Receber Dados - *web Service* recebe dados enviados pelo aplicativo e os salva no servidor online.

Também foram elaborados os modelos descritivos dos casos de uso (APÊNDICE A).

4.2 CASOS DE USO DO WEBSITE

O *website* implementa os seguintes casos de uso:

a) Criar conta que cria uma nova conta de usuário;
b) Realizar *Login* que inicia uma sessão e carrega os dados de usuário do sistema;
c) Realizar *Logout* que encerra uma sessão, impossibilitando o acesso aos dados a não ser que seja realizado um novo login;
d) Criar Lista cria uma nova lista de compras;
e) Editar Lista que edita os dados de uma lista de compras;
f) Excluir Lista que exclui uma lista de compras;
g) Duplicar Lista que duplica uma lista de compras;
h) Adicionar Produto que cadastra um novo produto em uma lista selecionada;
i) Editar Produto que altera os dados de um produto em uma lista selecionada;
j) Excluir Produto que exclui um produto em uma lista selecionada.

Assim como os casos de uso do aplicativo *android*, também foram criadas as descrições dos casos de uso do *website* (APÊNDICE B).

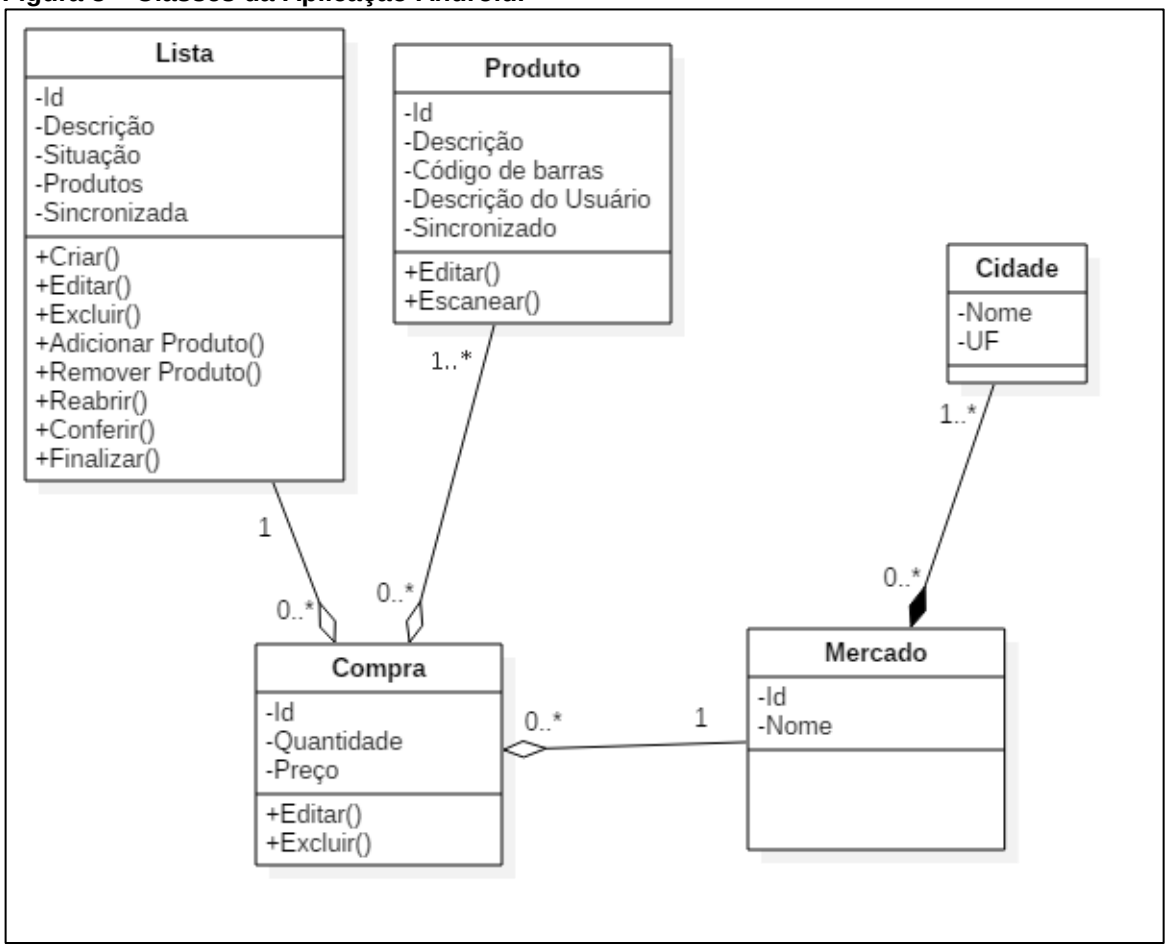
5 MODELO ESTRUTURAL DO SISTEMA

Com a reunião das informações necessárias para a concepção do aplicativo, *web service* e *website* e com os casos de uso já elaborados e consolidados, foi possível desenvolver o diagrama de classes, ainda como modelo estrutural para que auxilie na visualização dos requisitos levantados.

5.1 DIAGRAMA DE CLASSES DA APLICAÇÃO *ANDROID*

Na Figura 8 tem-se representado o modelo estrutural do aplicativo, utilizando-se do diagrama de classes da UML como modelo representativo.

Figura 8 – Classes da Aplicação Android.

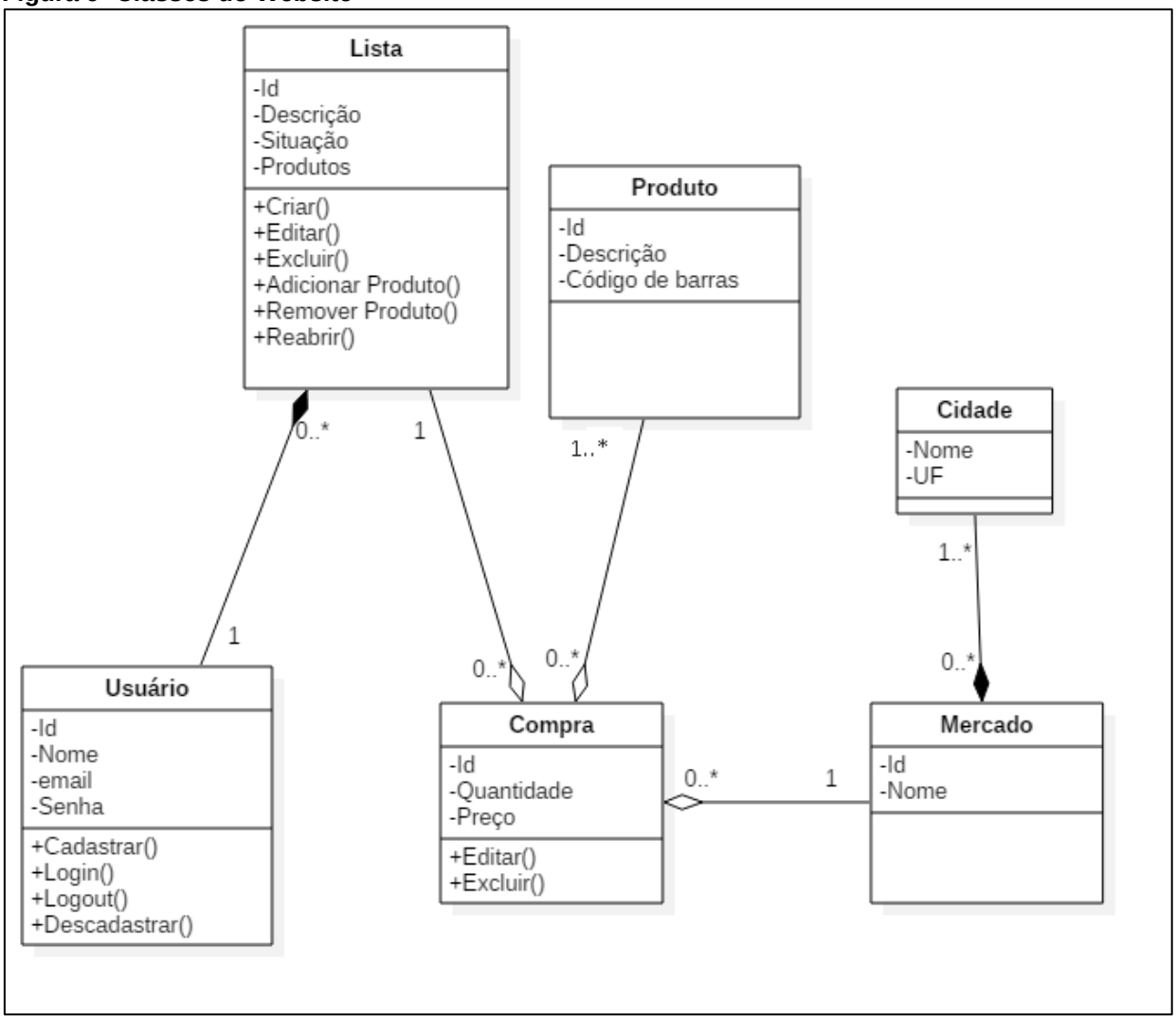


Fonte: Autoria Própria

5.2 DIAGRAMA DE CLASSES DO WEBSITE

Na Figura 9 tem-se representado o modelo estrutural do *website*, utilizando-se do diagrama de classes da UML como modelo representativo.

Figura 9 -Classes do Website



Fonte: Autoria Própria

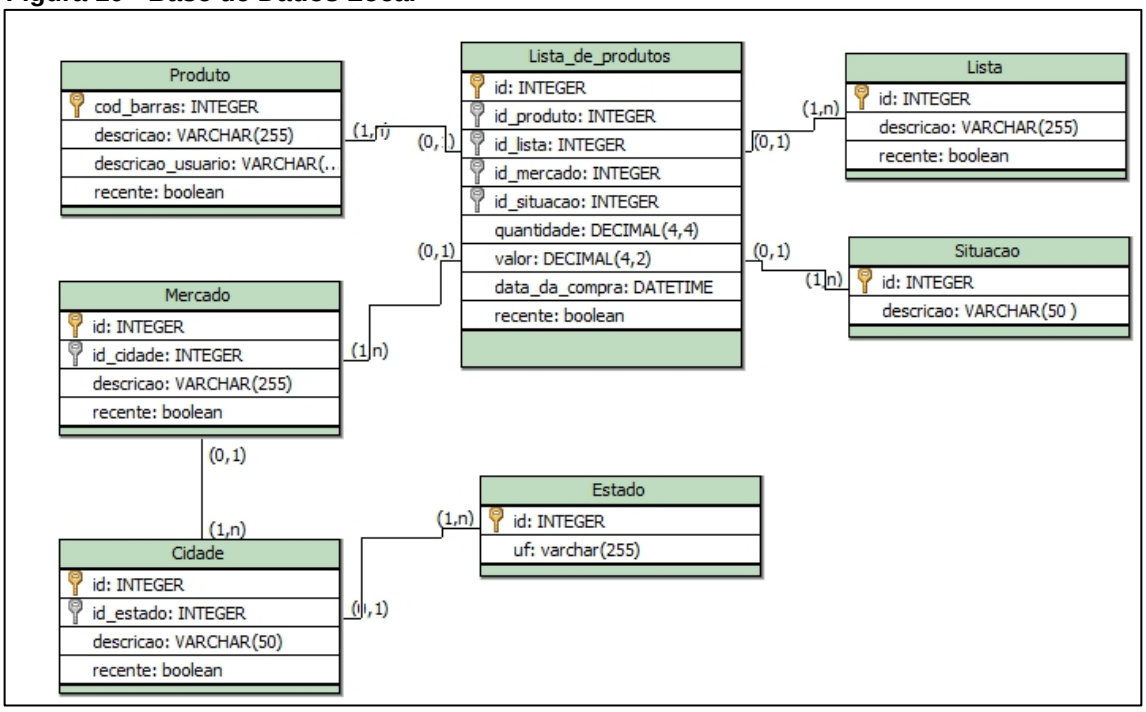
6 PROJETO DO BANCO DE DADOS

Neste capítulo será apresentado como foram projetados os bancos de dados *online* e *offline*.

6.1 MODELO DE ENTIDADE / RELACIONAMENTO

A base de dados local é representada através do modelo de entidade e relacionamento apresento na Figura 10.

Figura 10 - Base de Dados Local



Fonte: Autoria Própria

Para manter a normalização do banco, foi optado por centralizar os dados na tabela Lista_de_compras, o qual se relaciona com todas as outras tabelas. Visando manter o banco sempre atualizado, durante a sincronização de dados, a aplicação envia para o servidor, todos os produtos, listas de compras e mercados que estão com o atributo recente marcado, logo em seguida a aplicação apaga essas tabelas e recebe do *web service* todos os dados atualizados.

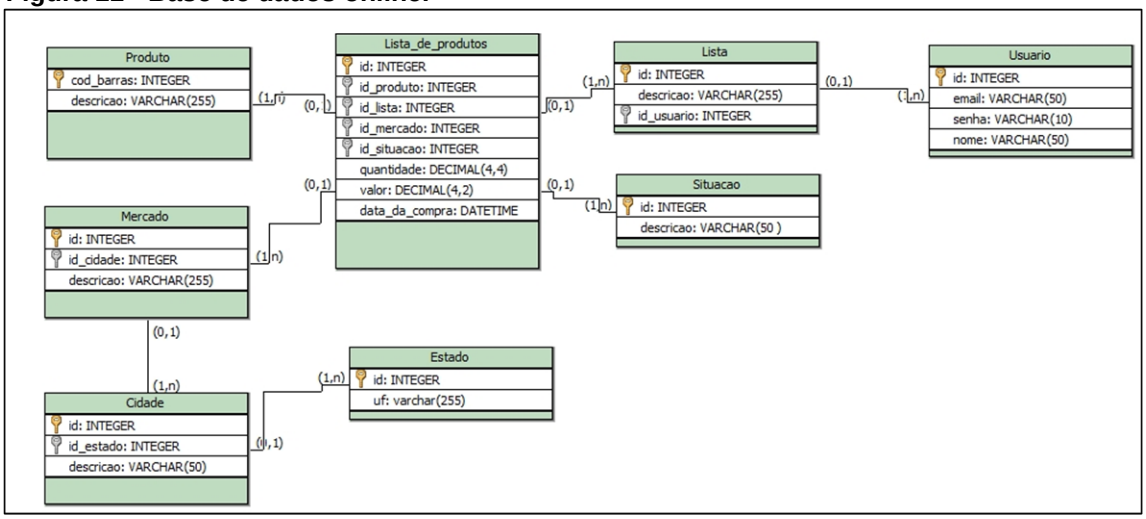
Existe uma particularidade na tabela *produtos* que é o atributo descricao_usuario, para garantir que um produto possua uma descrição condizente

com a realidade, a informação que o usuário digita é armazenada nesse atributo, mas a informação que é enviada para o *web service* vem do atributo descrição que é preenchido com base na descrição que está no cupom fiscal.

6.2 DIAGRAMA DO BANCO DE DADOS DO *WEB SERVICE*

A base de dados *online* está representada através do modelo de entidade e relacionamento apresentado na Figura 11.

Figura 11 - Base de dados online.



Fonte: Autoria Própria

O banco do *web service* é muito parecido com o banco local, com a diferença que não há necessidade de um atributo descricao_usuario, foi incluído a tabela usuário, utilizada para armazenar as informações de *login* e senha.

As tabelas produtos, lista_de_produtos, lista, mercado e cidade, servem como uma espécie de tabelas mães, pois o sistema foi projetado para que essas tabelas armazenem somente os dados mais atuais, pois é a partir delas que todas as bases locais são atualizadas.

7 WEB SERVICE

O *web service* é parte fundamental da projeto, uma vez que ele tem a função de conectar os sistemas *web* e *mobile* com a base de dados *online*.

As funções do *web service* podem ser divididas em:

a) cadastro de conta de usuário, onde durante uma requisição de cadastro no sistema são enviados para o *web service* o nome, e-mail, senha e cidade do usuário. O *web service* recebe os dados, atribui um identificador único e os grava no banco de dados *online*;
b) validação de *login*, onde durante uma tentativa de *login*, são enviados o e-mail e senha do usuário para o *web service,* este verifica se os dados informados existem e são correspondentes, retornando uma autorização de *login* em caso positivo;;
c) sincronização de listas, onde durante uma requisição do Aplicativo. Todas as Listas do usuário são enviadas para o aplicativo, que adicionará as novas e atualizará as existentes. Na sequência, o aplicativo enviará as listas que ainda não estão no ambiente online. O *web service* realizará a inclusão destas listas na base de dados para que o acesso fique disponível via *websit*e;
d) sincronização de produtos, onde durante a inicialização do aplicativo, uma requisição é feita ao *web service*, neste momento, todos os produtos de listas conferidas são enviadas ao *web service*, que os grava no banco *online*. Na sequência o *web service* envia para o aplicativo todos os produtos presentes na base *online*. Então, ao recebê-los, o aplicativo adicionará os produtos não existentes e atualizará os já presentes.

8 CONCLUSÃO

O presente trabalho foi projetado desde o início para automatizar um trabalho manual bastante recorrente nos dias de hoje. Suas principais funções são: facilitar a organização de listas de compras, auxiliar o usuário a manter controle de seus gastos e conferir o valor das compras realizadas em mercados. O objetivo pôde ser concluído utilizando-se das ferramentas de desenvolvimento *web* e desenvolvimento *mobile,* após um longo período de programação acreditamos no potencial da aplicação pois foi possível utilizar na prática o software o qual se mostrou um excelente aliado para que o consumidor só pague o valor que está na gondola.

Dentre as tecnologias utilizadas o *Android Studio SDK* foi de grande importância no desenvolvimento do aplicativo já que apresentou uma interface facilitadora para a criação das telas do aplicativo e por ter uma integração rápida com a documentação oficial do sistema operacional *Android*.

Uma dificuldade encontrada, no desenvolvimento do aplicativo *mobile,* foi manter a comunicação com o *web service* sincronizada, de forma que o banco de dados do aplicativo não entrasse em conflito com o banco de dados *online*, causando irregularidades nos registros, um exemplo disso foi constatado quando dos usuários diferentes fizessem compras em um mesmo estabelecimento, pois os dois usuários iriam enviar para o *webservice* que compraram produtos em um determinado supermercado, causando registros duplicados, tal dificuldade foi solucionada com o normalização da tabela de mercados usando o CNPJ como chave primária, o mesmo ocorreu com os produtos, o qual foi solucionado com a utilização do código de barras dos mesmos, tal solução só foi possível pois tanto o CNPJ quanto o código de barras são únicos.

Outra dificuldade foi obter as informações contidas no cupom fiscal através do site da Receita Federal. Diferente de uma NF-e (Nota fiscal eletrônica) que possui um arquivo XML (*Extensible Markup Language*) o NFC-e (Nota fiscal de consumidor eletrônica) não possui um arquivo para baixar, caso seja escaneado o QRCode contido na NFC-e, será direcionado para um site da Receita contendo as informações contidas naquele NFC-e. O grande desafio se deu ao tentar ler as informações contidas nesse HTML, a primeira dificuldade foi obter o *captcha* anti robôs para poder ter acesso as informações detalhadas da NFC-e, esse problema foi resolvido pois no

momento que é escaneado o QRCode é gerado um *captcha* válido no site da receita e passado pelo método *Get* para o webservice, dessa forma é possível capturar esse *captcha* através do endereço contido no QRCode. A segunda dificuldade aconteceu após alguns meses da implementação desse método, como as informações estão em formato HTML, a forma que optamos para ler esse HTML seria ler as *tags* e capturar os valores contidos nelas, mas de tempos em tempos a Receita altera essas *tags*, a solução que encontramos foi de tempos em tempos quando a Receita altera essas tags, lançarmos uma atualização corrigindo esse problema.

No desenvolvimento do *website,* uma dificuldade foi encontrada ainda na fase de elicitação de requisitos. O sistema realiza parte das operações que são feitas pelo aplicativo. Porém, foi necessário atenção para que as funções deste não aumentassem a necessidade de sincronizações entre o aplicativo e a base *online*. Com isso em mente, o *website,* teve suas funções reduzidas àquelas que realmente tirariam proveito de um sistema web, tais como: Criação de usuário, criação de listas de compras, visualização dos produtos com seus devidos valores e estabelecimentos comprados.

Não foi possível efetuar testes com outros usuários, para detectarmos problemas de usabilidade e possíveis *bugs* que outros usuários podem encontrar.

Pode-se dizer que o projeto alcançou todos os seus objetivos e mostrou também de forma categórica a viabilidade de aplicação de tecnologia móvel na resolução de problemas rotineiros da vida moderna.

8.1 TRABALHOS FUTUROS

O projeto apresenta um prospecto de evolução sobre IoT (*Internet of Things*), demonstrada pela facilidade de aquisição e transmissão de dados provenientes do usuário, que podem ser utilizadas para a tomada de decisão por meio de análises estatísticas, como por exemplo, criar listas de forma automática tendo como entradas as datas e compras dos últimos meses para determinado usuário, ou até mesmo realizar as compras de forma automática com a devida autorização, substituindo totalmente o processo de “levantamento de dados” hoje realizado manualmente por parte do usuário.

O sistema carece de uma forma mais eficaz de divulgação dos melhores preços encontrados, onde seja possível visualizar em qual mercado está o produto mais barato, e a possibilidade de acrescentar produtos de mercados diferentes em uma mesma lista de compras, dessa forma maximizando a economia.

Para complementar o software uma excelente oportunidade de melhoria seria a adição de um outro módulo financeiro, para controlar os gastos diários, não somente com compras, mas com todos os tipos de gastos que o usuário teria.

Outra contribuição futura seria estudar e aplicar conceitos de usabilidade e acessibilidade, para tornar o aplicativo mais popular e operável a qualquer pessoa. Nesse quesito seria possível a aplicação de questionários a usuários *testers,* implementação de comandos de voz, leitor de voz, padronização conforme o *Material Design,* entre outras possibilidades. Inclusive melhorar a usabilidade da principal funcionalidade do aplicativo que é o escaneamento de código de barras, já que dependendo do dispositivo, esta pode ser uma atividade bastante complicada e até mesmo impossível.

REFERÊNCIAS

ANDERSON, David J. **KANBAN - Successful Evolutionary Change for Your Technology Business.** Sequim, Washington, 2010.

ANDRUSHKO, Sviatoslav. **Framework x Bibliotexa x API. Entenda as Diferenças.** Disponível em :<https://becode.com.br/framework-biblioteca-api-entenda-as-diferencas>. Acesso em: 29 abr. 2018.

AMARAL, Hugo Richard; LIZARDO, Luis Eduardo Oliveira; DE SOUZA, Arthur Camara Vieira. **PostgreSQL: uma alternativa para sistemas gerenciadores de banco de dados de código aberto.** Anais do Congresso Nacional Universidade, EAD e Software Livre. Vol. 2. No. 2. 2011.

AMBLER, S. **Agile modeling**. New York: Wiley Computer. Publishing, 2002.

APPLE, Inc. **Apple Developer Program**. Disponível em: <http://developer.apple.com /programs>. Acesso em: 1° dez. 2016.

APPLE, Inc. **XCODE.** Disponível em: <https://developer.apple.com/xcode>. Acesso em: 1° dez. 2016.

ATLASSIAN. **Trello.** Disponível em: < https://trello.com/home>. Acesso em: 28 mai. 2018.

BECK, Kent. **Extreme Programming Explained: Embrace Change**. Estados Unidos - Boston: Addison-Wesley Professional, 2005.

BOWES, Jim. **Kanban vs Scrum vs XP – Na Agile comparison.** Disponível em: <https://manifesto.co.uk/kanban-vs-scrum-vs-xp-an-agile-comparison>. Acesso em: 11 mai. 2018.

CARLOS, J. Feijó Lopes; RAMALHO, José Carlos. **Web services: metodologias de desenvolvimento.** Fevereiro 2004. Disponível em: <http://repositorium.sdum.uminho .pt/bitstream/1822/559/1/LR04.pdf>. Acesso em: 30 nov. 2016.

CARVALHO, Suelen. **Android Studio: vantagens e desvantagens com relação ao Eclipse.** Disponível em: <http://imasters.com.br/mobile/android/android-studiovantagens-e-desvantagens-com-relacao-ao-eclipse/>. Acesso em: 6 out. 2015.

CONALLEM, J.: **"Building Web Applications with UML".** Addison Wesley, 2000.

DE LEONE, Leonello. **6 Frameworks PHP que você precisa conhecer!.** Disponível em: <https://becode.com.br/frameworks-php-que-voce-precisa-conhecer>. Acesso em: 29 abr. 2018.

FRANÇA, L.; STAA, A. **Geradores de Artefatos: Implementação e Instanciação de frameworks.** Simpósio Brasileiro de Engenharia de Software, 15. SBC, 2001. PP 302-315.

GAJOTRES. **12 Best PHP RESTful Micro Frameworks (Pro/Con).** Disponível em: <https://www.gajotres.net/best-available-php-restful-micro-frameworks/4/>. Acesso em: 22 out. 2017.

GILFILLAN, Ian. **Bíblia de MySQL.** Espanha – Madrid: Anaya Multimedia, 2003.

GLAUBER, Nelson. **Dominando o Android.** Novatec, São Paulo, 2015.

GOOGLE. **Google Play Developer Console.** Disponível em: <http://play.google .com/apps/publish/signup>. Acesso em: 1° dez. 2016.

GREENROBOT. **greenDao the best way to access SQLite.** 2016. Disponível em: <http://greenrobot.org/greendao/>. Acesso em: 29 abr. 2018.

GS1, Brasil. **Código de barras.** Disponível em: <https:// www.gs1br.org/codigos-e-padroes/codigo-de-barras>. Acesso em: 30 nov. 2016.

KOROTYA, Eugeniya. **AWS vs Heroku: Cloud Platform Comparison for 2017.** Disponível em: <https://hackernoon.com/aws-vs-heroku-cloud-platform-comparison-for-2017-5f2194c0673e>. Acesso em: 11 mai. 2018.

KURTZ, João. **Aplicativos universais da Microsoft agora se chamam apenas Windows Apps.** Disponível em: <http://www.techtudo.com.br/noticias/noticia/2015/03/aplicativos-universais-da-microsoft-agora-se-chamam-apenas-windows-apps.html>. Acesso em: 05 set. 2017.

LEANKIT. **What is Kanban?** Disponível em: < https://leankit.com/learn/kanban/what-is-kanban>. Acesso em: 11 mai. 2018.

LECHETA, R, R. **Google Android: Aprenda a criar aplicações para dispositivos Móveis com Android SDK.** 4. Edição. São Paulo: Novatec, 2015. LEICHER, Andreas. **Should I use Heroku or AWS?.** Disponível em: <https://hackernoon.com/should-i-use-heroku-or-aws-3bfcd4706a36>. Acesso em: 29 abr. 2018.

LOPES, A, S; JUNIOR, A, M, S. **Desenvolvimento de Uma Aplicação Android Para Pesquisa de Preços em Supermercados, Acessível à Deficientes Visuais.** Campos dos Goytacazes. Mai. 2016.

MICROSOFT. **Build native iOS, Android and Windows apps in Visual Studio.** Disponível em: <https://www.xamarin.com/visual-studio#designers>. Acesso em: 22 out. 2017.

MICROSOFT. **Windows Store Developer**. Disponível em: <https://developer.microsoft.com/pt-br/store/register>. Acesso em: 05 set. 2017.

NIEDERAUER, Juliano. **PHP para quem conhece php: Recursos avançados para a criação de Websites dinâmicos.** São Paulo. 2017.

PARDO, Michael. **ActiveAndroid**. Disponível em: <http://www.activeandroid.com/>. Acesso em: 22 out. 2017.

PINHO, Diego. **O ECMAScript 6 e o Futuro do Javascript.** Disponível em: <https://imasters.com.br/front-end/javascript/o-ecmascript-6-e-o-futuro-do-javascript/?trace=1519021197&source=single>. Acesso em: 29 abr. 2018.

RODRIGUES, J. HTML básico. **DEVMEDIA**. 2010. Disponível em: <https://www.devmedia.com.br/html-basico-codigos-html/16596>. Acesso em: 30 nov. 2016.

SANTINO, Renato, **Quase 9 em cada 10 celulares no mundo usam Android.** Disponível em: <https://olhardigital.com.br/noticia/quase-9-em-cada-10-celulares-no-mundo-usa-android/63634>. Acesso em: 7 set. 2017.

SCHIMID, J. Introdução ao Javascript**. DEVMEDIA**. 2012. Disponível em: <https://www.devmedia.com .br/introducao-ao-javascript/25548>. Acesso em: 30 nov. 2016.

SILVA, M. M; SANTOS, M, T. **Os Paradigmas de Desenvolvimento de Aplicativos Para Aparelhos Celulares.** TIS, São Carlos. Mai./ago. 2014.

SIMÕES, D. D; PEREIRA, J. C. **Sistemas Operacionais Móveis – Android x ios.** Paranavaí. 2014.

SISTEMA PÚBLICO DE ESCRITURAÇÃO DIGITAL. **Nota Fiscal de Consumidor Eletrônica.** 2010. DISPONÍVEL EM: <http://www.sped.fazenda.pr.gov.br/modules /conteudo/conteudo.php?conteudo=92>. Acesso em: 29 abr. 2018.

SOUZA, Elson. **O que esperar da versão final do Windows 10 para celulares?** Disponível em: <http://www.techtudo.com.br/noticias/noticia/2015/05/o-que-esperar-da-versao-final-do-windows-10-para-celulares.html>. Acesso em: 5 set. 2017.

SUDA, B. **SOAP Web Services.** Edinburgh, Escócia. Mai. 2003.

SUGIMORI, Y., KUSUNOKI, K., CHO, F., UCHIKAWA, S., **Toyota production system and Kanban system Materialization of just-in-time and respectfor-human system, International Journal of Production Research.** Disponível em: <http://www.tandfonline.com/doi/pdf/10.1080/00207547708943149?needAccess=true&>. Acesso em: 5 set. 2017.

SUTHERLAND, Jeff. **Scrum: The Art of Doing Twice the Work in Half of the Time**. Estados Unidos - Boston: Crown Business, 2014.

SYMFONY. **What is Symfony?.** Disponível em: <https://symfony.com/what-is-symfony>. Acesso em: 29 abr. 2018.

TRELLO. **Trello lets you work more collaboratively and get more done.** Disponível em: < https://trello.com/home>. Acesso em: 10 mai. 2018.

VERMA, Vijay. **Why Laravel is Best PHP Framework.** Disponível em: <https://blog.vanila.io/why-laravel-is-best-php-framework-98a2784d76dc>. Acesso em: 10 mai. 2018.

WATSON, Gray. **OrmLite - Lightweight Object Relational Mapping (ORM) Java Package.** Disponível em: <http://ormlite.com/>. Acesso em: 22 out. 2017.

WELLS, Don. **Extreme Programming: A gentle introduction.** Disponível em: <http://www.extremeprogramming.org>. Acesso em: 11 mai. 2018.

WINCKLER, Marco; PIMENTA, Marcelo Soares. **Avaliação de usabilidade de sites web.** Porto Alegre: Sociedade Brasileira de Computação (SBC), 2002.

ZANETTE, Alysson. **Framework x Bibliotexa x API. Entenda as Diferenças.** Disponível em: <https://becode.com.br/framework-biblioteca-api-entenda-as-diferencas>. Acesso em: 29 abr. 2018.

ZURIARRAIN, J. M. Disponível em: <https://brasil.elpais.com/brasil/2017/04/04/tecnologia/1491296467_396232.html>. Acesso em: 22 out. 2017.

APÊNDICE A – Modelo Descritivo dos Casos de Uso do Aplicativo Android

ID UC001	
Nome do Caso de Uso	Criar conta
Ator(es)	Principal: Usuário Suporte: *Web Service*
Descrição	Este Caso de uso tem por objetivo permitir a criação de uma conta de usuário.
Pré-condições	Conexão com a internet
Pós-condições	Caso de uso “UC017 - Receber Dados” é executado.
Cenário Principal	1 - Usuário clica no botão de *Sign In*. 2 - Usuário insere informações de *logon*. 3 - Aplicativo conecta-se com o *web service*. 4 - As informações são enviadas ao *web service*.
Cenário Alternativo	Não há.

ID UC002	
Nome do Caso de Uso	Realizar *login*
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir que o usuário tenha acesso ao aplicativo.
Pré-condições	Não há
Pós-condições	Não há
Cenário Principal	O usuário clica no botão de *login*. O usuário entra com suas credenciais. O caso de uso “UC015 - Validar *login*” é executado. Usuário recebe acesso ao aplicativo.
Cenário Alternativo	Não há.

ID UC003	
Nome do Caso de Uso	Realizar *logout*
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo encerrar o acesso ao aplicativo.
Pré-condições	Usuário estar com uma sessão ativa.
Pós-condições	Não há
Cenário Principal	Usuário acessa a tela de configurações. Usuário clica no botão de *logout.* Aplicativo dispara mensagem de confirmação de *logout*. O acesso ao aplicativo é encerrado.
Cenário Alternativo	Não há.

ID UC004	
Nome do Caso de Uso	Criar lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a criação de uma lista de compras.
Pré-condições	Não há
Pós-condições	Não há
Cenário Principal	Usuário toca no botão '+' na tela de listas de compras. O sistema dispara uma tela de diálogo para que o usuário informe o nome da lista de compras. Usuário informa o nome da lista de compras que será criada. Sistema valida o nome informado. Uma nova lista de compras é criada.
Cenário Alternativo	Não há.

ID UC005	
Nome do Caso de Uso	Editar lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a edição do nome de uma lista de compras.
Pré-condições	Existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há
Cenário Principal	Na tela de listas de compras, o usuário realiza um toque longo sobre a lista que deseja alterar. A *ActionBar* do sistema passa a exibir opções de contexto. O usuário toca sobre a opção referente à edição de listas. O sistema dispara um diálogo para que o usuário informe o novo nome da lista. O usuário informa o novo nome e confirma. O sistema valida o novo nome e grava as informações.
Cenário Alternativo	Não há.

ID UC006	
Nome do Caso de Uso	Excluir lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a exclusão de uma lista de compras.
Pré-condições	Existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há
Cenário Principal	Na tela de listas, o usuário realiza um toque longo sobre a lista que deseja excluir. A *ActionBar* do aplicativo passa a exibir opções de contexto O usuário toca sobre a opção referente à exclusão de lista. O aplicativo apresenta um diálogo requisitando confirmação para a exclusão. O usuário confirma a exclusão. A lista é excluída do sistema.
Cenário Alternativo	Não há.

ID UC007	
Nome do Caso de Uso	Duplicar lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a duplicação de uma lista de compras.
Pré-condições	Existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há
Cenário Principal	Na tela de listas, o usuário realiza um toque longo sobre a lista que deseja duplicar. A *ActionBar* do aplicativo passa a exibir opções de contexto. O usuário toca sobre a opção referente à duplicação de lista. O aplicativo apresenta um diálogo requisitando confirmação para a duplicação. O usuário confirma a duplicação. O aplicativo apresenta um diálogo requisitando um novo nome para a lista duplicada. O usuário digita o novo nome e confirma. A lista é duplicada no sistema.
Cenário Alternativo	Não há.

ID UC008	
Nome do Caso de Uso	Conferir lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a conferência dos valores entre uma lista de compras e o cupom fiscal.
Pré-condições	A existência de pelo menos uma lista de compras com produtos comprados.
Pós-condições	Não há
Cenário Principal	Na tela de listas, o usuário realiza um toque longo sobre a lista que deseja conferir. A *ActionBar* do aplicativo passa a exibir opções de contexto O usuário toca sobre a opção referente à conferência de lista. Um diálogo é apresentado confirmando se o usuário deseja escanear um cupom fiscal. O usuário clica no botão “OK” para confirmar. O usuário escaneia o *QRcode* do cupom fiscal. O sistema marca as compras como checadas e apresenta as irregularidades, se houverem.
Cenário Alternativo	Não há.

ID UC009	
Nome do Caso de Uso	Reabrir lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a reabertura de uma lista de compras já conferida.
Pré-condições	A existência de pelo menos uma lista de compras conferida.
Pós-condições	Não há
Cenário Principal	Na tela de listas, o usuário realiza um toque longo sobre a lista que deseja reabrir. A *ActionBar* do aplicativo passa a exibir as opções de contexto. O usuário toca sobre a opção referente à Reabertura de listas. O aplicativo apresenta um diálogo requisitando confirmação para a reabertura da lista. O usuário confirma a reabertura. A lista volta para o status inicial, somente com a descrição dos produtos preenchida.
Cenário Alternativo	Não há.

ID UC010	
Nome do Caso de Uso	Adicionar produto
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a inserção de um produto em uma lista de compras.
Pré-condições	Ao menos uma lista de compras cadastrada no sistema.
Pós-condições	Não há
Cenário Principal	Na tela de listas, o usuário toca sobre a lista que deseja vincular produtos. O aplicativo apresenta a tela de compras da lista. O usuário digita o nome do produto que deseja comprar. O usuário insere a quantidade do produto que deseja comprar. O usuário toca sobre o botão ‘OK’ para confirmar a inclusão do produto na lista de compra
Cenário Alternativo	Na tela de listas, o usuário toca sobre a lista que deseja vincular produtos. O aplicativo apresenta a tela de compras da lista. O usuário clica no botão “escanear”. O usuário escaneia o código de barras do produto. O aplicativo preenche a descrição do produto. O usuário insere a quantidade do produto que deseja comprar. O usuário toca sobre o botão ‘OK’ para confirmar a inclusão do produto na lista de compra
Cenário Alternativo	Na tela de listas, o usuário toca sobre a lista que deseja vincular produtos. O aplicativo apresenta a tela de compras da lista. O usuário clica no botão “escanear”. O usuário escaneia o código de barras do produto. O aplicativo solicita a descrição do produto escaneado. O usuário preenche os campos e clica no botão “OK”. O aplicativo preenche a descrição do produto que deseja comprar. O usuário insere a quantidade do produto que deseja comprar. O usuário toca sobre o botão ‘OK’ para confirmar a inclusão do produto na lista de compra

ID UC011	
Nome do Caso de Uso	Editar produto
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a edição da descrição, quantidade ou valor de um produto.
Pré-condições	Ao menos um produto adicionado na lista de compras.
Pós-condições	Não há
Cenário Principal	Na tela de lista de compras, o usuário realiza um toque longo sobre a compra que deseja alterar. A *ActionBar* do sistema passa a exibir opções de contexto. O usuário toca sobre a opção referente à edição de produto. O usuário escolhe qual atributo do produto deseja editar. O usuário confirma a edição do produto. O sistema aplica e grava as alterações realizadas.
Cenário Alternativo	Não há.

ID UC012	
Nome do Caso de Uso	Excluir produto
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a exclusão de um produto em uma lista de compras.
Pré-condições	Ao menos um produto adicionado na lista de compras.
Pós-condições	Não há
Cenário Principal	Na tela de lista de compras, o usuário realiza um toque longo sobre uma compra que deseja excluir. A *ActionBar* passa a exibir as opções de contexto. O usuário realiza um toque sobre a opção de exclusão de produto. Um diálogo é apresentado perguntando se o usuário realmente deseja excluir o produto. O usuário toca em ‘OK’ para confirmar a exclusão. O produto é excluído da lista de compra.
Cenário Alternativo	Não há.

ID UC013	
Nome do Caso de Uso	Comprar produto
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a exclusão de um produto em uma lista de compras.
Pré-condições	Ao menos um produto adicionado na lista de compras.
Pós-condições	Não há
Cenário Principal	Na tela de lista de compras, o usuário realiza um toque na caixa de seleção “Comprado” ao lado da descrição do produto. A linha que indica o registro do produto fica com a cor de fundo na cor verde. O sistema marca o produto como comprado.
Cenário Alternativo	Na tela de lista de compras, o usuário realiza um toque na caixa de seleção “Comprado” ao lado da descrição do produto. O sistema solicita para escanear o código de barras do produto. O sistema solicita para preencher o valor do produto. A linha que indica o registro do produto fica com a cor de fundo na cor verde. O sistema marca o produto como comprado.

ID UC014	
Nome do Caso de Uso	Sincronizar listas
Ator(es)	Principal: Usuário Suporte: *Web Service*
Descrição	Este Caso de uso tem por objetivo enviar as listas para o *web service* e atualiza-las.
Pré-condições	Conexão com a internet
Pós-condições	Não há
Cenário Principal	O usuário toca sobre a opção referente à sincronização de dados. O sistema se comunica com o web service. O caso de uso “UC016 - Enviar listas” é executado. Os dados de listas cadastrados na base local são enviados para o web service. O caso de uso “UC017 - Recebe dados” é executado. Uma mensagem de que a sincronização foi realizada com sucesso é exibida ao usuário.
Cenário Alternativo	Não há.

ID UC015	
Nome do Caso de Uso	Validar *login*
Ator(es)	Principal: *Web Service*
Descrição	Este Caso de uso tem por objetivo verificar se as informações de login existem no *web service*.
Pré-condições	Caso de uso “UC002 - Realizar Login” foi executado.
Pós-condições	Não há
Cenário Principal	Os dados de login são recebidos pelo *web service*. A correção dos dados de login é verificada. A confirmação de login é enviada ao aplicativo cliente.
Cenário Alternativo	Não há.

ID UC016	
Nome do Caso de Uso	Enviar listas
Ator(es)	Principal: *Web Service*
Descrição	Este Caso de uso tem por objetivo enviar as listas de compras para o aplicativo.
Pré-condições	O caso de uso “UC014 - Sincronizar Listas” foi executado.
Pós-condições	Não há
Cenário Principal	O *web service* seleciona as listas do usuário que requisitou acesso. As listas são enviadas para o aplicativo cliente.
Cenário Alternativo	Não há.

ID UC017	
Nome do Caso de Uso	Receber dados
Ator(es)	Principal: *Web Service*
Descrição	Este Caso de uso tem por objetivo receber dados enviados pelo aplicativo.
Pré-condições	O casos de usos “UC014 - Sincronizar Listas” ou “UC018 - Criar Conta” foram executados.
Pós-condições	Não há
Cenário Principal	Os dados enviados pelo aplicativo cliente são recebidos. Os dados são gravados na base de dados online.
Cenário Alternativo	Não há.

APÊNDICE B – Modelo Descritivo dos Casos de Uso do Website

ID UC018	
Nome do Caso de Uso	Criar conta
Ator(es)	Principal: Usuário Suporte: *Web service*
Descrição	Este Caso de uso tem por objetivo cadastrar um novo usuário.
Pré-condições	Não há.
Pós-condições	Caso de uso “UC017 - Receber Dados” é executado.
Cenário Principal	Usuário clica no botão de *Sign In*. Usuário insere informações de *logon*. *Website* conecta-se com o *web service*. As informações são enviadas ao *web service*.
Cenário Alternativo	Não há.

ID UC019	
Nome do Caso de Uso	Realizar *login*
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir o acesso a área restrita do site.
Pré-condições	Não há.
Pós-condições	Não há.
Cenário Principal	O usuário clica no botão de l*ogin*. O usuário entra com suas credenciais. *Website* valida os dados. Usuário recebe acesso a área restrita do *website*.
Cenário Alternativo	Não há.

ID UC020	
Nome do Caso de Uso	Realizar *logout*
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso encerra o acesso a área restrita do *website*.
Pré-condições	Usuário estar com uma sessão ativa no sistema.
Pós-condições	Não há.
Cenário Principal	Usuário clica no botão de *logout*. *Website* dispara mensagem de confirmação de *logout*. O acesso a área restrita do site é encerrado.
Cenário Alternativo	Não há.

ID UC021	
Nome do Caso de Uso	Criar lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a criação de uma nova lista de compras.
Pré-condições	Usuário estar com uma sessão ativa no sistema.
Pós-condições	Não há.
Cenário Principal	Usuário acessa a área de lista de compras. Usuário toca no botão '+' na tela de Listas. O sistema dispara uma tela de diálogo para que o usuário informe o nome da lista. Usuário informa o nome da lista que será criada. Sistema valida o nome informado. Uma nova lista de produtos é criada.
Cenário Alternativo	Não há.

ID UC022	
Nome do Caso de Uso	Editar lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a edição do nome de uma lista de compras.
Pré-condições	Usuário estar com uma sessão ativa no sistema e existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há.
Cenário Principal	Na tela de listas, o usuário clica no botão que possui um ícone de um lápis. O sistema dispara um diálogo para que o usuário informe o novo nome da lista. O usuário informa o novo nome e confirma. O sistema valida o novo nome e grava as informações.
Cenário Alternativo	Não há.

ID UC023	
Nome do Caso de Uso	Excluir lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a exclusão de uma lista de compras.
Pré-condições	Usuário estar com uma sessão ativa no sistema e existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há.
Cenário Principal	Na tela de listas, o usuário clica no botão que possui um ícone de lixeira. O *website* apresenta um diálogo requisitando confirmação para a exclusão. O usuário confirma a exclusão. A lista é excluída do sistema.
Cenário Alternativo	Não há.

ID UC024	
Nome do Caso de Uso	Duplicar lista
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a duplicação de uma lista de compra.
Pré-condições	Usuário estar com uma sessão ativa no sistema e existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há.
Cenário Principal	Na tela de listas, o usuário clica no botão com um ícone de duplicação. O *website* apresenta um diálogo requisitando confirmação para a duplicação. O usuário confirma a duplicação. O aplicativo apresenta um diálogo requisitando um novo nome para a lista duplicada. O usuário digita o novo nome e confirma. A lista é duplicada no sistema.
Cenário Alternativo	Não há.

ID UC025	
Nome do Caso de Uso	Adicionar produto
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a inserção de um produto em uma lista de compra.
Pré-condições	Usuário estar com uma sessão ativa no sistema e existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há.
Cenário Principal	Na tela de listas, o usuário clica sobre a lista que deseja vincular produtos. O *website* apresenta a tela de compras da lista. O usuário digita o nome do produto que deseja comprar. O usuário clica sobre o botão 'OK' para confirmar a inclusão do produto na lista de compra
Cenário Alternativo	Não há.

ID UC026	
Nome do Caso de Uso	Editar produto
Ator(es)	Principal: Usuário
Descrição	Este Caso de uso tem por objetivo permitir a edição da descrição, quantidade e valor de um produto em uma lista de compra.
Pré-condições	Usuário estar com uma sessão ativa no sistema e existir ao menos uma lista de compras cadastrada.
Pós-condições	Não há.
Cenário Principal	Na tela de lista de compras, o usuário clica no botão com um ícone de lápis. O sistema habilita que os campos de descrição, quantidade e valor possam ser alterados. O usuário confirma a edição do produto. O sistema aplica e grava as alterações realizadas. A tela é atualizada para refletir as últimas mudanças.
Cenário Alternativo	Não há.