UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JIAN RODRIGO CASASOLLA

APLICATIVO ANDROID PARA REVENDEDORAS REALIZAREM PEDIDOS DE PRODUTOS DO SEGMENTO TÊXTIL

TRABALHO DE CONCLUSÃO DE CURSO 2

PATO BRANCO
2018

JIAN RODRIGO CASASOLLA

APLICATIVO ANDROID PARA REVENDEDORAS REALIZAREM PEDIDOS DE PRODUTOS DO SEGMENTO TÊXTIL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão 2 do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco.

Orientador: Prof João Guilherme Brasil Pichetti

PATO BRANCO
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

APLICATIVO ANDROID PARA REVENDEDORAS REALIZAREM PEDIDOS DE PRODUTOS DO SEGMENTO TÊXTIL

POR

JIAN RODRIGO CASASOLLA

Este trabalho de conclusão de curso foi apresentado no dia 05 de dezembro de 2018, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Prof. Esp. João Guilherme Brasil Pichetti
Orientador

Profª MSc Andreia Scariot Beulke

Prof MSc. Robison Cris Brito

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª Drª Beatriz Terezinha Borsoi
Responsável pela Atividade de Trabalho de Conclusão de Curso

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

AGRADECIMENTOS

Ao nosso Deus por ter nos dado a vida, capacidade de enfrentar todos os momentos difíceis, me permitindo viver momentos como este.

A minha família, que sempre me apoiaram, incentivaram e me fizeram acreditar que sou capaz de concluir mais essa etapa na vida.

A Universidade Tecnológica Federal do Paraná, por terem me dado à oportunidade de ingressar no curso, me fornecendo tudo que é necessário para uma boa formação.

Agradeço a todos os professores e amizades que se fizeram presentes nesta jornada, compartilhando seus conhecimentos e dando importância a uma boa formação profissional.

Enfim, obrigado a todas as pessoas, que direta ou indiretamente, contribuíram com o nosso trabalho, formação e sucesso.

Dedico a Deus, que nos criou e foi criado nesta tarefa, e me deu coragem para questionar realidades e propor sempre um novo mundo de possibilidades.

RESUMO

CASASOLLA, Jian Rodrigo. Aplicativo Android para revendedoras realizarem pedidos de produtos do segmento têxtil. 2018. 56f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

A geração Y, que nasceu com a tecnologia, Internet e *smartphones* é o público que mais efetua compras *on-line* correspondendo a 58% do total. Deste percentual, 41% dos pedidos são efetivados por meio de contatos telefônicos. Muitas dessas compras são realizadas por meio da utilização de aplicativos que possuem a finalidade de fornecer produtos com fins de *e-commerce*, que tem o objetivo de proporcionar um serviço mais personalizado e ágil, procurando atender as tendências tecnológicas do mercado atual. O Android é um dos sistemas operacionais para *smartphones*, com possibilidade de instalar aplicativos para efetuar compras *on-line*. Desta forma, as compras da categoria *fashion* são as que mais crescem e, para que o aplicativo seja de uso efetivo, o *design* precisa ser intuitivo. Assim, este trabalho propõe o desenvolvimento de um aplicativo Android, seguindo ao estilo de *underwear*, para efetivação de pedidos das revendedoras de empresas do ramo têxtil, consideradas como clientes. Por meio do aplicativo, o cliente poderá se aproximar ainda mais do seu fornecedor, tendo uma interatividade e comunicação ainda maior. O funcionário é quem fará o controle e gerenciamento dos pedidos. O representante, se responsabilizará pela entrega e cobrança dos pedidos aprovados.

Palavras-chave: Android. Aplicativo Móvel. *E-commerce*.

ABSTRACT

CASASOLLA, Jian Rodrigo. Android app for resellers to order products from the textile segment. 2018. 56f. Monography (Course Completion Work) - Superior Course in Technology in Systems Analysis and Development, Federal Technological University of Paraná, Pato Branco Campus. Pato Branco, 2018.

Generation Y, which was born with technology, Internet and smartphones, is the public that most purchases online, corresponding to 58% from the total. Of this percentage, 41% of requests are made through telephone groups. Many of these purchases are made through applications that have a series of products for e-commerce purposes, which aim to offer a better service to prices, in addition to meeting the technological needs of the current market. Android is one of the operating systems for smartphones, with the ability to install applications for the online shopping. In this way, fashion purchases are the fastest growing and, for the application to be of effective use, the design needs to be intuitive. Thus, this work proposes the development of an Android application, following the style of underwear, for the fulfillment of requests from resellers of textile companies, as customers. Through the application, the customer can be even more important to your supplier, have an even greater interactivity and communication. The employee, who will the control and management of orders. The representative, will take the responsibility for the delivery and collection of approved applications.

Keywords: Android. Mobile Application. *E-commerce*.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

API	*Application Programming Interface*
ERP	*Enterprise Resource Planning*
GPS	*Global Positioning System*
HTTP	*HyperText Transfer Protocol*
REST	*Representational State Transfer*
TI	Tecnologia de Informação
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa de realização do trabalho. Por fim está a apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

Quando se deseja que a empresa esteja à frente de seus concorrentes, as tecnologias são uma boa opção para um funcionamento mais eficiente, desde a parte gerencial até a comercial. Dentre as diversas tecnologias existentes, destacam-se os aplicativos móveis, pois proporcionam mais facilidade de acesso e oferecem recursos para potencializar negócios e melhorar a relação com os clientes. Empresas de pequeno, médio e grande porte, como, por exemplo, salão de beleza, supermercados, *petshops*, academias, dentre outras, têm utilizado essa tendência.

Estima-se de modo geral que aproximadamente 40% das compras que são realizadas, possuem a intervenção de aplicativos *mobile* (CINTRA, 2017) permitindo que empresas lucrem com essa mobilidade.

O Sistema Operacional da Google é o Android e tem seu foco em dispositivos móveis por ser baseado em sistemas de código aberto. O sistema operacional, é um *software* que faz o gerenciamento do sistema. O Android, por exemplo, gerencia um *tablet* ou celular, fornecendo dados em uma interface visual, na qual o usuário pode interagir sem necessitar conhecer como o sistema operacional é composto.

Neste trabalho, foi desenvolvido dois aplicativos móveis, utilizando a plataforma Android, sendo que um desses aplicativos possibilitará ao cliente (revendedor) realizar pedidos antes da visita do seu representante da região. O aplicativo permite identificar por meio da *Application Programming Interface* (API) do Google realizar a autenticação do cliente. O sistema permite que sejam inseridos códigos de cupons de descontos em pedidos, para recompensar os clientes referentes aos produtos e serviços oferecidos pela loja e em caso de efetivação de pedido de compra, informando o código do cupom, o cliente possuirá descontos em sua compra.

1.2 OBJETIVOS

A seguir são apresentados os objetivos pretendidos com a realização deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um aplicativo que utiliza a plataforma Android para realização de pedidos de revendedoras de produtos do segmento têxtil.

1.2.2 Objetivos Específicos

Por meio do aplicativo que foi desenvolvido é possível:

- Permitir a realização de pedidos de compras pelos revendedores.
- Permitir manter um histórico de compras realizadas por revendedores;
- Disponibilizar a visualização dos produtos ofertados pela empresa;
- Facilitar a comunicação de cliente e empresa.

1.3 JUSTIFICATIVA

A essência da organização do segmento de *underwear* é a criação de produtos ou serviços, almejando atender clientes de forma eficiente e eficaz, devido ao alto padrão de tecnologia aplicada no *design*.

Com um número de clientes elevado, subdivididos por representantes e regiões, o grande intervalo de tempo entre uma visita e outra acaba se tornando um incômodo para quem deseja aumentar as vendas. Além disso, ainda há probabilidades de não ter os produtos que o cliente deseja.

Com o aumento da tecnologia, surgiu a oportunidade das empresas

acompanharem e investirem em vendas, por meio de aplicativos móveis. O estudo da empresa GlobalWebIndex que é especializada em estudos relacionados à Internet e tecnologia, realizado em janeiro de 2015, aponta que 80% dos entrevistados possuem *smartphone*, representando, assim, uma oportunidade para uma empresa expor o seu negócio com uma abrangência maior (AMADOR, 2015).

Segundo a pesquisa realizada por Albertin (2016), os gastos e investimentos referentes ao comércio eletrônico e no setor de Tecnologia de Informação (TI) tiveram crescimento, atingindo uma média geral de 2,26% do faturamento líquido das empresas. Com isso, as empresas estão preocupadas em melhorar o relacionamento e o atendimento ao cliente e estão utilizando a tecnologia para melhor atendê-los.

Para efetivar os objetivos, foi desenvolvido para as revendedoras um aplicativo móvel na plataforma Android para realização de pedidos, visando facilitar a interatividade da empresa com o cliente de forma mais ágil.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos. O capítulo 1 apresenta as considerações iniciais, os objetivos e a justificativa. O capítulo 2 apresenta o referencial teórico. No capítulo 3 estão as ferramentas e tecnologias empregadas na modelagem do sistema na implementação subsequente do sistema. O resultado da realização do trabalho é apresentado no capítulo 4. Por fim, no capítulo 5 estão as considerações finais seguidas pelas referências utilizadas no texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico que se baseia na proposta deste trabalho refere a aplicativo em Android para os clientes realizarem seu pedido de compra.

2.1 COMÉRCIO ELETRÔNICO

Em 1995, os Ministérios das Comunicações e da Ciência e Tecnologia disponibilizaram o primeiro provedor de internet, e o acesso à internet no Brasil se difundiu de forma privada. Depois disso, o comércio eletrônico começou a tomar uma direção e sua abrangência ganhou força rumo a Rede Mundial de Consumo. (FERREIRA et al, 2012).

O comércio eletrônico engloba intensamente o mundo das tecnologias de comunicação e de informação, realizando uma cadeia de valores, de processos e de negócios em um ambiente eletrônico, atendendo um objetivo em específico. São processos realizados de forma parcial ou completa, em uma infraestrutura de informação e comunicação, incluindo transações negócio-a-negócio, negócio-a-consumidor e infra organização, de acesso fácil, livre e de baixo custo. (ALBERTIN, 2000).

De acordo com Finkelstein (2011), o comércio eletrônico é uma modalidade de compra a distância, onde são enviadas e recebidas as transmissões de dados por meio eletrônico.

2.2. ANDROID

Neste subcapítulo será apresentada a plataforma Android.

2.2.1 HISTÓRICO DO ANDROID

O Android surgiu em 2003, após seus criadores avaliarem o mercado para criação de um sistema para câmeras digitais e perceberem que o mercado mobile era mais promissor. (MEYER, 2017).

Conhecida inicialmente por Android Inc. e situada na Califórnia, Estados Unidos da América (EUA), foi adquirida em 2005, pela empresa Google e passou a se chamar Google Mobile Division (OGLIARI; BRITO, 2014). Já em 2007, a criação do android teve grande avanço, quando parcerias com empresas de hardware reuniram-se em um consórcio de tecnologia e fundaram a Open Handset Alliance, tendo como objetivo a criação de uma plataforma de código aberta para smartphones. No ano seguinte, a primeira versão estava disponível em um smartphone chamado HTC Dream, desenvolvido pela empresa taiwanesa HTC. (MEYER, 2017).

Segundo dados da pesquisa International Data Corporation, em 2014 o Android estava presente em quase 85% dos *smartphones* em mais de 190 países, sendo considerada uma plataforma que cresce rapidamente e a mais utilizada ao redor do mundo. (DIAS, 2014).

2.2.2 DEFINIÇÃO DO ANDROID

O Android é uma plataforma para desenvolvimento e execução de programas para dispositivos móveis, com um projeto de código aberto e baseado em Linux, possui interface robusta e de fácil utilização e aprendizagem. (OGLIARI; BRITO, 2014).

A linguagem de programação do Android é o Java, e sua interface visual é baseado na utilização de arquivos *eXtensible Markup Language* (XML). Possui aposta em novos aparelhos celulares, que popularmente são conhecidos como *smartphones*, que são celulares com performance de processamento avantajado e que integram diversos recursos, como: alta conectividade com a internet, *Global Positioning System* (GPS), sensores e telas sensíveis ao toque. (OGLIARI; BRITO, 2014).

Os aplicativos desenvolvidos na plataforma Android, podem ser publicados e distribuídos pela loja virtual Play Store ou Google Play, sendo possível o *download* e compra de aplicativos móveis.

2.2.3 ARQUITETURA DO ANDROID

O sistema operacional Android, tem sua infraestrutura baseada em camadas que são: *applications, application framework*, *libraries,* android *runtime* e *kernel* linux, que provêm serviços essenciais (BORDIN, 2012). A Figura 1, demonstra a arquitetura do sistema operacional Android.

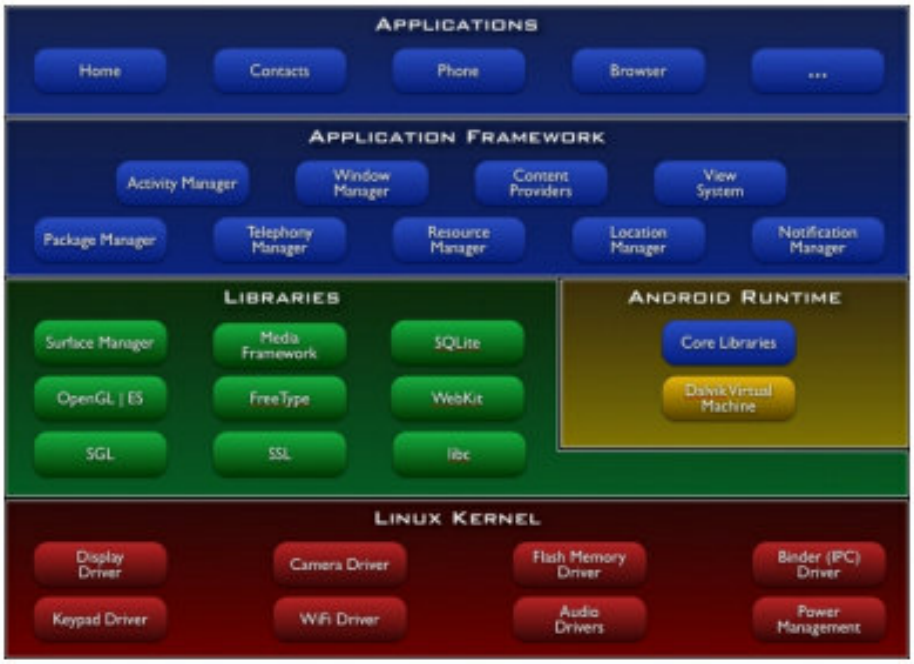

Figura 1 - Camadas de software do Android
Fonte: Platform Versions (BORDIN, 2012)

Os *Application*s são conjuntos de aplicativos que são nativos para serviços como e-mail, navegadores, contatos, envio de *Short Message Service* (SMS) e permite, também, que serviços de terceiros sejam executados. (ANDROID DEVELOPER, 2017).

A *Application Framework* são acessos para desenvolvedores que visam facilitar o reuso de serviços de alto nível essenciais como gerenciamento de janelas, recursos e notificações do dispositivo. (OGLIARI; BRITO, 2014).

As *Libraries* são compostas por bibliotecas C/C++ que são usadas em componentes com funções específicas do Android. Entre as bibliotecas destacam-se o *SQLite* que é uma implementação de banco de dados, o 3D *Libraries* que é utilizado para otimizar a aceleração de um determinado *hardware*, *Media Libraries*

que suporta a gravação e o *playback* de diversos formatos de mídia. (OGLIARI; BRITO, 2014).

O Android *Runtime* é um conjunto das principais bibliotecas básicas que detêm funcionalidades em Java que determinam o ambiente de execução da aplicação. Até a versão 5.0, a máquina virtual Dalvik, era o tempo de execução do Android, para versões posteriores a citada, o respectivo aplicativo executa seus processos com instâncias próprias do Android Runtime. (ANDROID DEVELOPER, 2017).

Já o Linux *Kernel*, faz a comunicação entre o *software* desenvolvido e o *hardware* do dispositivo, sendo responsável por tarefas, como: realizar a segurança, acesso à rede, gerenciamento de processos e memória.

2.2.4 VERSÕES DO ANDROID

Em setembro de 2008, foi lançada a primeira versão comercial. Com exceção da primeira versão (1.0) e da segunda (1.1) que não receberam nomes, as demais todas foram designadas com nomes de sobremesas. Posteriormente, foram lançadas as seguintes versões: Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop e Marshmallow. (MEYER, 2017).

Algumas das versões que não possuem mais suporte são: Cupcake, Donut, Eclair e Froyo. As versões lançadas depois de dezembro de 2010, ainda possuem suporte, que é o caso da versão 2.3.3. O Quadro 1 contém informações das versões que ainda apresenta suporte.

Versão	**Nome da versão**	***Application Program Interface* (API)**	**Distribuição**
2.3.3 – 2.3.7	Gingerbread	10	0.2%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x	Jelly Bean	17	1.5%
4.3	Jelly Bean	18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1	Lollipop	22	14.4%

6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1	Nougat	25	10.1%
8.0	Oreo	26	14.0%
8.1	Oreo	27	7.5%

Quadro 1 - Versões do Android e distribuição
Fonte: Platform Versions (ANDROID DEVELOPER, 2018)

3 MATERIAIS E MÉTODO

A seguir são descritos os materiais e o método utilizados para a modelagem e a implementação do sistema obtido como resultado deste trabalho.

3.1 MATERIAIS

O Quadro 2 apresenta as tecnologias e as ferramentas utilizadas no desenvolvimento do trabalho.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
MySQL Workbench	6.3.9	https://dev.mysql.com/downloads/workbench/	Modelagem do banco de dados.
Java		Linguagem para *back-end*	Linguagem de programação.
Android Studio	2.3	http://developer.android.com/sdk/index.html	Ferramenta em Java para programação do aplicativo mobile.
PostgreSQL	9.6	https://jdbc.postgresql.org/	Banco de dados para armazenamento de informações em geral.
Eclipse IDE	4.7.2	https://www.eclipse.org/	Ferramenta em Java para programação do WebService.
pgAdmin IV	1.2	https://www.pgadmin.org/download/	Administrador de Banco PostgreSQL.
StarUML	2.8.0	http://staruml.io/download	Modelagem dos casos de uso, diagrama de classes.
Retrofit	2.2.0	https://square.github.io/retrofit/	Biblioteca HTTP Client para Android e Java

Quadro 2 - Tecnologias e ferramentas utilizadas

3.2 MÉTODO

O desenvolvimento deste trabalho consiste na realização de atividades do fluxo de processo iterativo, utilizando-se das fases de levantamento de requisitos, análise, projeto, desenvolvimento e testes (PRESSMAN, 2006). A seguir estão definidas as etapas desenvolvidas para a efetivação deste trabalho.

a) Levantamento de requisitos

Esta fase tem por objetivo definir as funcionalidades para o desenvolvimento do sistema. Para isso, foram realizadas conversas informais com os colaboradores pertencentes a uma empresa do ramo têxtil.

Durante o desenvolvimento do trabalho, existiu a necessidade de serem adicionados novos requisitos, como também alterações nos já existentes de modo a melhorar a estrutura e funcionamento do sistema.

b) Análise e projeto

Usando como base os requisitos levantados, foram elaborados os casos de uso do sistema, que foram utilizados para gerar informações para a etapa da definição de banco de dados. Os diagramas da *Unified Modeling Language* (UML) foram criados, utilizando a ferramenta StarUML.

c) Desenvolvimento

Para o desenvolvimento foi utilizada a ferramenta Android Studio para criação dos aplicativos móveis e a ferramenta Eclipse para o WebService. A implementação dos cadastros foi padronizada com operações de inclusão, edição e exclusão.

d) Testes

Os testes foram realizados durante o desenvolvimento de modo informal, visando identificar erros de codificação, verificação e validação dos requisitos definidos.

4 RESULTADO

Este capítulo apresenta o resultado deste trabalho que é a modelagem e o desenvolvimento de um sistema para revendedores cadastrarem pedidos de produtos do ramo têxtil.

4.1 ESCOPO DO SISTEMA

Ao acessar o sistema, o cliente, considerado um revendedor, pode realizar seu cadastro no aplicativo. Caso já esteja cadastrado, o cliente poderá realizar pedidos de compra informando os itens, a quantidade, a condição de pagamento e, se tiver cupom de desconto para o período da realização do pedido, poderá informar o código do cupom que será descontado do valor total do pedido. Os pedidos serão compostos de produtos pré-cadastrados na ferramenta disponibilizada para o funcionário administrador do sistema. O cliente poderá emitir relatório do seu histórico de pedidos finalizados, como também possuir a localização da empresa via *maps*, formas de contato e informações da empresa.

O administrador do sistema, poderá monitorar os pedidos cadastrados, se no mesmo, por exemplo, conter algum produto que não possua em estoque, o pedido poderá ser rejeitado. Caso o pedido seja rejeitado, o cliente poderá consultar o pedido, no relatório disponível para monitoramento dos *status* dos pedidos.

A cobrança do pedido é realizada pelo representante no ato da entrega, e o pagamento poderá ser em dinheiro, cartão de crédito ou débito, cheque ou nota promissória. A entrega do produto, será realizada pelo representante no ato de sua visita na região. As visitas dos representantes podem variar de acordo com a região de cada cliente, podendo ser semanalmente, mensalmente ou com possibilidades de ser a cada sessenta dias.

O cliente poderá acompanhar o *status* do seu pedido efetuado, por meio da tela principal do sistema. Os *status* estão definidos em: aberto, separação, a caminho, entregue e cancelado.

4.2 MODELAGEM DO SISTEMA

Os Quadros 3 a 10 apresentam os requisitos funcionais e não funcionais identificados para o sistema.

F1 - Manter clientes				
Descrição: O sistema permite o cadastro de clientes contendo nome, endereço, telefone, CPF ou CNPJ, email, data de nascimento, gênero, profissão e as demais informações pessoais. Além disso, é permitido editar o perfil do cadastro do cliente.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF1.1 Validar CPF ou CNPJ	O sistema executa uma função de validação do CPF e CNPJ contenha a quantidade de dígitos necessários.	Regra de negócio	( )	(X)
NF1.2	O sistema não permite que sejam inseridas letras no cadastro do CPF, CNPJ e Telefone	Regra de Implementação	( )	(X)
NF1.3 Validar email	O sistema não permite que sejam realizados mais que um cadastro com o mesmo email.	Regra de Negócio	( )	(X)

Quadro 3 - Requisito manter clientes

F2 - Manter produtos				
Descrição: O sistema permite o cadastro de produtos contendo nome, descrição, tamanho, cor, descrição da imagem e valor unitário. Além disso, deve é permitido editar e excluir cadastros.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF2.1 Excluir	O sistema não permite excluir um item que esteja vinculado a um pedido de compra.	Integridade	( )	(X)

Quadro 4 - Requisito manter produtos

F3 – Cadastrar pedido de compra				
Descrição: O sistema permite o cadastro de pedidos de compra. Além disso, é permitido editar e excluir cadastros de pedidos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF3.1 Excluir	O sistema não permite excluir um pedido que já esteja aprovado.	Integridade	( )	(X)
NF3.2 Informar produtos	O sistema não permite que sejam cadastrados pedidos sem itens.	Regra de implementação	( )	(X)

Quadro 5 - Requisito pedido de compra

F4 – Alterar status pedido de compra				
Descrição: O sistema permite alterar o *status* dos pedidos de compra.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF4.1 Validar saldo em estoque	O sistema não permite que seja aprovado um pedido com itens sem saldo em estoque.	Regra de implementação	( )	(X)
NF4.2 Aprovar pedido	O sistema permite que somente funcionários realizem a aprovação do pedido.	Regra de negócio	( )	(X)

Quadro 6 - Requisito aprovar pedido de compra

F5 – Manter cupom de desconto				
Descrição: O sistema oferece ao usuário administrador a possibilidade de fazer a inclusão, exclusão e alteração de cupons de desconto.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF5.1 Acesso	O sistema permite que somente administradores tenham acesso a essa funcionalidade	Regra de negócio	( )	(X)

Quadro 7 - Requisito manter cupom de desconto

F6 – Emitir relatório de vendas
Descrição: O sistema oferece ao funcionário a possibilidade de emitir um relatório listando suas vendas.

Quadro 8 - Requisito emitir relatório de vendas

F7 – Emitir relatório de compras do cliente
Descrição: O sistema oferece ao cliente a possibilidade de emitir um relatório com suas compras realizadas, filtrando pelo status do pedido.

Quadro 9 - Requisito emitir relatório de compras de cliente

O diagrama de casos de uso apresentado na Figura 1, mostra as principais funcionalidades e quais atores interagem com o sistema.

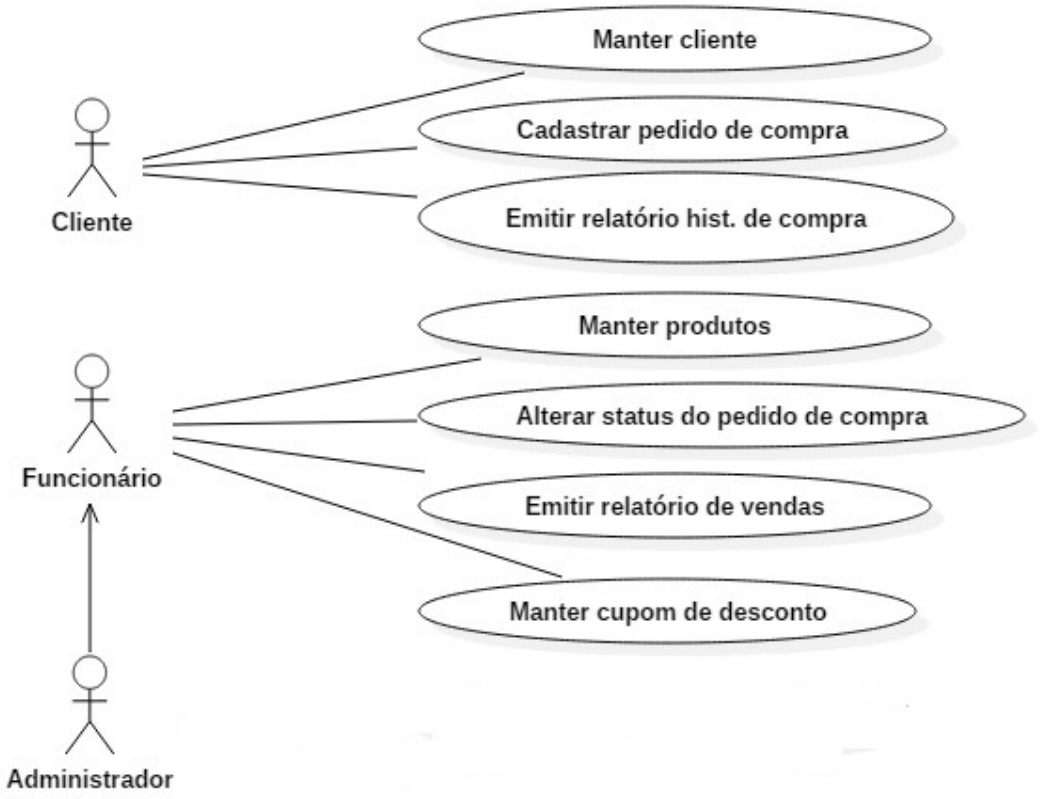

Figura 2 - Diagrama de casos de uso

Os Quadros 10 a 12, apresentam a operação de cadastro. Essa descrição se refere à operação de inclusão de todos os casos de uso identificados como “manter”.

Casos de uso	Incluir (refere-se a todos os casos de uso identificados como “manter”)
Descrição	Incluir dados cadastrais no sistema.
Evento indicador	Acessar o menu de cadastro.
Ator(es)	Cliente, administrador ou funcionário de acordo com as funções definidas no caso de uso.
Pré-condição	Ter o aplicativo instalado em seu dispositivo.
Sequência de eventos	1 – O ator acessa o menu de cadastro.
	2 – O ator informa os dados necessários para o cadastro.
	3 – Os dados são salvos no banco de dados e uma mensagem informativa é mostrada para o usuário.
Pós-condição	Dados do cadastro inseridos no banco.
Fluxo alternativo	Descrição
1 Campos obrigatórios não foram preenchidos	1.1 Antes de salvar os dados, é realizada uma verificação de campos obrigatórios não informado pelo usuário.

	1.2 É exibida uma mensagem informativa para o usuário. 1.3 A opção de cadastro continuará disponível.
2 Inserção de dados inválidos	2.1 Antes de salvar os dados, é realizada uma verificação nos campos com o propósito de identificar campos inválidos. 2.2 É exibida uma mensagem informativa para o usuário. 2.3 A opção de cadastro continuará disponível.

Quadro 10 - Operação de inclusão dos casos de uso de cadastro

Casos de uso	Alterar (refere-se a todos os casos de uso identificados como "manter")
Descrição	Alterar dados cadastrados no sistema.
Evento indicador	Ator solicitar a alteração de algum cadastro.
Ator(es)	Cliente ou administrador de acordo com as funções definidas no caso de uso.
Pré-condição	O registro já estar cadastrado.
Sequência de eventos	1 – O ator acessa a tela para visualizar os registros.
	2 – O ator informa o registro que deseja alterar.
	3 – O ator altera os dados do registro.
	4 – Os dados são alterados no banco de dados e uma mensagem informativa é mostrada para o usuário.
Pós-condição	Dados do cadastro alterados no banco.
Fluxo alternativo	Descrição
1 Campos obrigatórios não foram apagados	1.1 Antes de salvar os dados, é realizada uma verificação de campos obrigatórios não informado pelo usuário. 1.2 É exibida uma mensagem informativa para o usuário. 1.3 A opção de alteração continuará disponível.
2 Inserção de dados inválidos	2.1 Antes de salvar os dados, é realizada uma verificação nos campos com o propósito de identificar campos inválidos. 2.2 É exibida uma mensagem informativa para o usuário. 2.3 A opção de alteração continuará disponível.

Quadro 11 - Operação de alteração dos casos de uso de cadastro

Casos de uso	Excluir (refere-se a todos os casos de uso identificados como "manter")
Descrição	Excluir dados cadastrados no sistema.
Evento indicador	Ator solicitar a exclusão de algum cadastro.
Ator(es)	Cliente ou administrador de acordo com as funções definidas no caso de uso.
Pré-condição	O registro já estar cadastrado.
Sequência de eventos	1 – O ator acessa a tela para excluir os registros.
	2 – O ator informa o registro que deseja excluir.
	3 – Os dados são excluídos do banco de dados e uma mensagem informativa é mostrada para o usuário.
Pós-condição	Dados do cadastro excluídos do banco de dados

Fluxo alternativo	Descrição
1 Registro possui vinculo no sistema	1.1 Usuário seleciona um registro que possui vinculo no sistema. 1.2 É exibida uma mensagem informativa para o usuário dizendo que não será possível realizar a exclusão do registro.

Quadro 12 - Operação de exclusão dos casos de uso de cadastro

O Quadro 13 representa a operação de cadastro de pedido de compra do ator cliente.

Casos de uso	Cadastrar pedido de compra.
Descrição	O cliente informará os itens.
Evento indicador	Acessar o menu de pedido.
Ator(es)	Cliente.
Pré-condição	Ter o aplicativo instalado em seu dispositivo e estar cadastrado como cliente.
Sequência de eventos	1 – O cliente acessa o menu pedido.
	2 – O cliente informa os itens que deseja comprar.
	3 – O cliente informa a tipo de pagamento.
	4 – O cliente informa o cupom de desconto.
	5 – O cliente verifica se o cupom está ativo.
	6 – Em seguida pressiona o botão enviar pedido.
	7 – Os dados serão salvos e será mostrada uma mensagem informativa para o usuário.
Pós-condição	Dados do pedido inseridos no banco.
Fluxo alternativo	Descrição
3 Inserção do cupom	3.1 Caso o cliente tenha um cupom de desconto informa no campo destinado, caso não tenha segue para o item 6.
3 Validação do cupom de desconto	3.2 Ao informar o cupom de desconto será realizada uma validação para verificar se está ativo. Se estiver ativo, será informado no campo do valor do desconto, o valor atribuído ao pedido. Caso não esteja cadastrado, o sistema informará o valor de R$ 0,00 e não será aplicado desconto ao valor total do pedido.

Quadro 13 - Operação de inclusão de pedido de compra

No Quadro 14 é apresentado a expansão da operação de aprovação do pedido de compra.

Pós-condição	*Status* do pedido será alterado no banco.
Fluxo alternativo	Descrição
Evento indicador	Acessar o menu alterar *status* pedidos.
Ator(es)	Funcionário.
Pré-condição	Ter o aplicativo instalado em seu dispositivo e o cliente ter finalizado um pedido de compra.
Sequência de eventos	1 – O funcionário acessa o menu alterar *status* pedidos.
	2 – O funcionário procura o pedido.
	3 – Em seguida altera o *status* do pedido.
	4 – Na sequência, o funcionário pressiona o botão de atualizar *status*.

Quadro 14 - Operação de aprovar pedido de compra

A Figura 3, representa o diagrama de entidades e relacionamento do banco de dados do sistema.

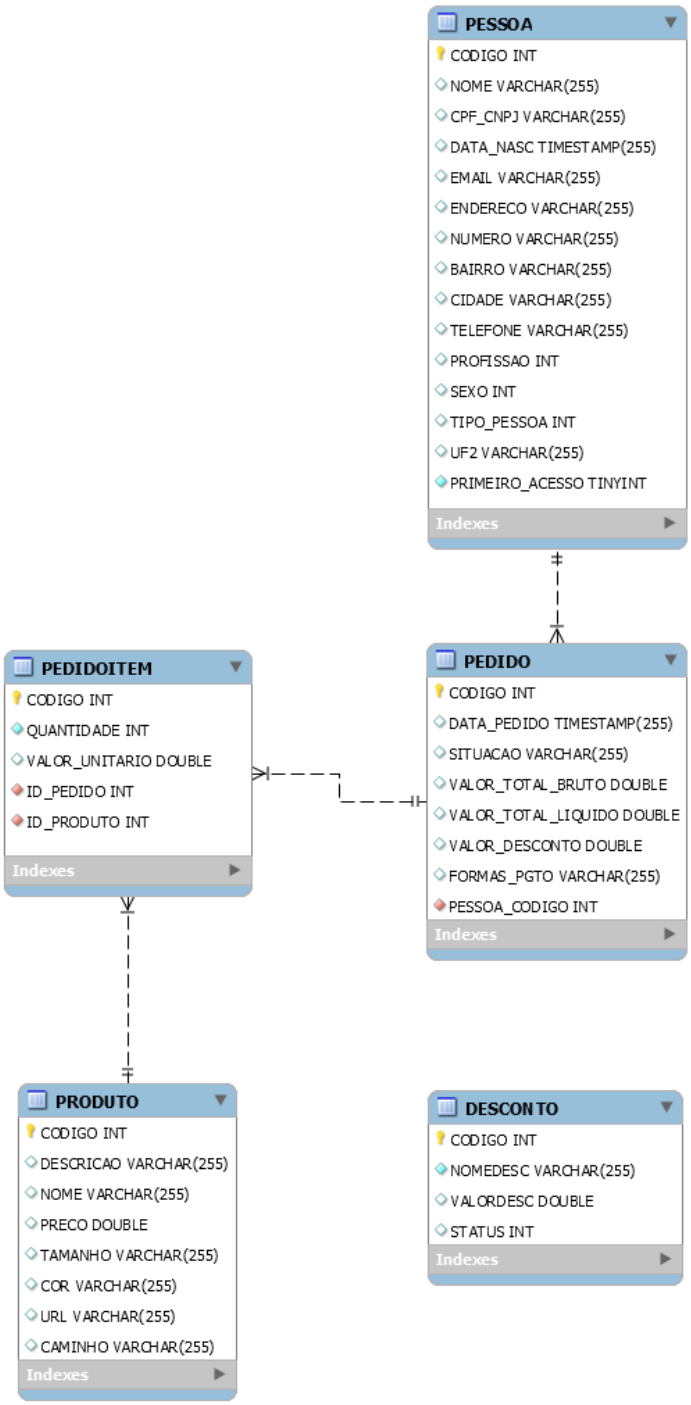

Figura 3 - Diagrama de Entidade-Relacionamento

O Quadro 15, apresenta os campos da tabela cliente, onde são cadastradas as informações de cadastro do cliente.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
CODIGO	Integer	Não	Sim	Não	Chave primária
NOME	Varchar	Sim	Não	Não	Nome do cliente
CPF_CNPJ	Varchar	Sim	Não	Não	CPF ou CNPJ
DATA_NASC	Timestamp	Sim	Não	Não	Data de Nascimento
EMAIL	Varchar	Sim	Não	Não	Endereço eletrônico
ENDERECO	Varchar	Sim	Não	Não	Rua
NUMERO	Varchar	Sim	Não	Não	Número
BAIRRO	Varchar	Sim	Não	Não	Bairro
CIDADE	Varchar	Sim	Não	Não	Cidade
TELEFONE	Varchar	Sim	Não	Não	Número de telefone de contato
PROFISSAO	Integer	Sim	Não	Não	Do enum de profissão do cliente
SEXO	Integer	Sim	Não	Não	Do enum de sexo do cliente
UF2	Varchar	Sim	Não	Não	Unidade da Federação
PRIMEIRO_ACESSO	Tinyint	Não	Não	Não	Primeiro login do cliente

Quadro 15 - Campos da tabela Cliente

Os campos da tabela de pedido são apresentados no Quadro 16. No pedido, o cliente pode inserir cupons de descontos e informar os tipos de pagamentos. Posteriormente, o pedido será vinculado ao funcionário que realizará sua aprovação.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
CODIGO	Integer	Não	Sim	Não	Chave primária
DATA_PEDIDO	Timestamp	Sim	Não	Não	Data de abertura
SITUACAO	Varchar	Sim	Não	Não	Situação do pedido
VALOR_TOTAL_BRUTO	Double	Sim	Não	Não	Valor bruto
VALOR_TOTAL_LIQUIDO	Double	Sim	Não	Não	Valor líquido
VALOR_DESCONTO	Double	Sim	Não	Não	Valor desconto
FORMAS_PGTO	Varchar	Sim	Não	Não	Do enum de Forma de pagamento
PESSOA_CODIGO	Integer	Não	Sim	Sim	Da tabela de pessoa

Quadro 16 - Campos da tabela Pedido

No Quadro 17, é apresentado os campos da tabela de produtos do pedido. Esta tabela foi criada com o intuito de resolver a relação de muitos para muitos: um pedido pode conter vários produtos e um produto pode estar vinculado a muitos pedidos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações

CODIGO	Integer	Não	Sim	Não	Chave primária
QUANTIDADE	Integer	Sim	Não	Não	Cor
VALOR_UNITÁRIO	Double	Sim	Não	Não	Valor unitário
ID_PEDIDO	Integer	Não	Sim	Sim	Da tabela Pedido
ID_PRODUTO	Integer	Não	Sim	Sim	Da tabela Produto

Quadro 17 - Campos da tabela Pedidoitem

O Quadro 18, apresenta os campos da tabela Produto.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
CODIGO	Integer	Não	Sim	Não	Chave primária
DESCRICAO	Varchar	Sim	Não	Não	Descrição
NOME	Varchar	Sim	Não	Não	Nome
PRECO	Double	Sim	Não	Não	Valor unitário
TAMANHO	Varchar	Sim	Não	Não	Tamanho
COR	Varchar	Sim	Não	Não	Cor
URL	Varchar	Sim	Não	Não	Endereçamento da imagem do produto
CAMINHO	Varchar	Sim	Não	Não	Local da imagem

Quadro 18 - Campos da tabela Produto

No Quadro 19, contém os campos da tabela Desconto.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Observações**
CODIGO	Integer	Não	Sim	Não	Chave primária
NOMEDESC	Varchar	Sim	Não	Não	Nome
VALORDESC	Double	Sim	Não	Não	Valor unitário
STATUS	Integer	Sim	Não	Não	Status do cupom

Quadro 19 - Campos da tabela Desconto

4.3 APRESENTAÇÃO DO SISTEMA

Qualquer aplicação para o sistema operacional Android, está se tornando uma referência para usuário final e desenvolvedores, tanto pela sua grande quantidade, mas como, também, pela sua qualidade na utilização dos componentes visuais disponíveis.

No Android, a interface gráfica é correspondente aos componentes visuais que intermediam a relação com o usuário, sendo então definidos por meio de *tags* escritas no arquivo XML do layout e também possuem classes de códigos, utilizadas na linguagem Java por meio de *activities* que permitem realizar o controle das telas da aplicação. O arquivo de manifest.xml, que também é criado em toda aplicação, tem a função de atribuir permissões, controlar *activities* e outras ações pertinentes ao aplicativo.

Esta seção, contém os recursos tecnológicos do sistema e as suas funcionalidades por meio de telas. A Figura 4, representa a primeira tela do sistema que possui o botão que permite a realização do *login*, por meio de conta no Google.

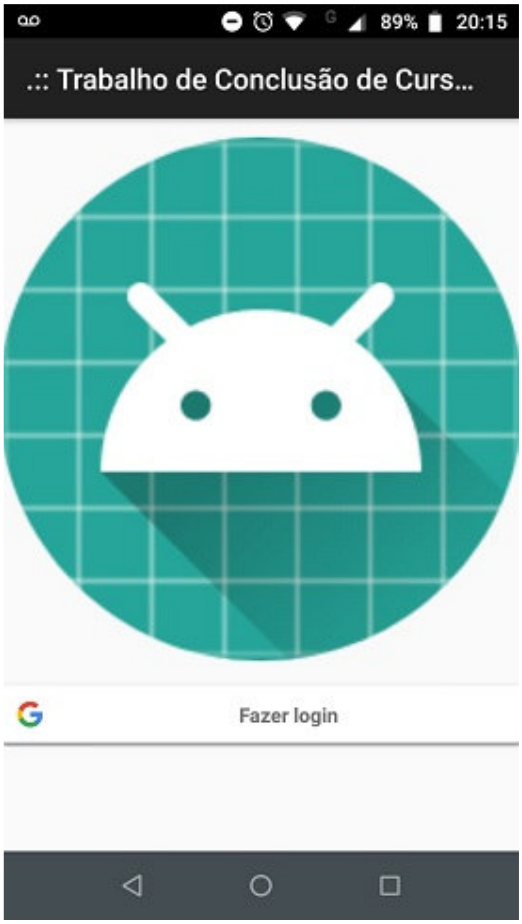

Figura 4 - Primeira tela do aplicativo do cliente

Após autenticar-se no sistema utilizando uma conta do Google, o usuário é direcionado para a tela de menu e seu o nome e o email são exibidos na parte superior da tela, conforme demonstrado na Figura 5.

Figura 5 - Tela de menu

Caso seja o primeiro acesso, ao clicar na opção minha loja virtual o usuário é direcionado para a tela de cadastro de perfil, conforme apresentado na Figura 6. Na opção de voltar conta gmail, o usuário será redirecionado para a tela inicial apresentada na Figura 4 e na opção fazer *logoff*, o aplicativo será desconectado da conta do Google.

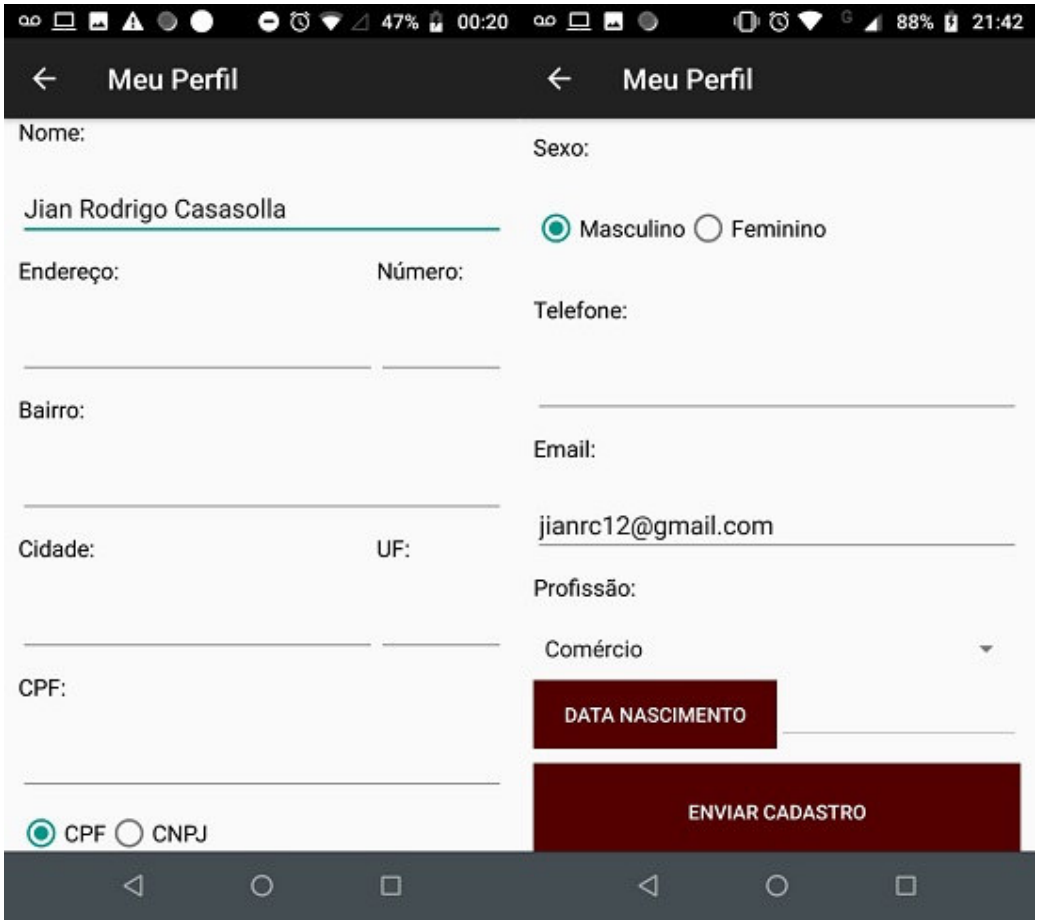

Figura 6 - Tela de cadastro de pessoa

No cadastro de pessoa, o nome e o email não são editáveis, pois são oriundos da conta do Google. Nos campos para os dígitos de CPF ou CNPJ, e telefone, são realizadas validações, verificando se os caracteres inseridos, atendem ao requisito de dígitos mínimos necessários. Os demais campos são de preenchimento obrigatório.

A partir do segundo acesso, quando o usuário for acessar o sistema com a mesma conta de email cadastrado, será direcionado para a tela de *dashboard*, representada na Figura 7.

Figura 7 - Tela de dashboard

No painel de *dashboard*, ao clicar em fazer pedido, serão listados todos os produtos cadastrados (Figura 8), como a sua respectiva descrição, contendo uma imagem do produto, a cor, o tamanho e o preço bruto do produto.

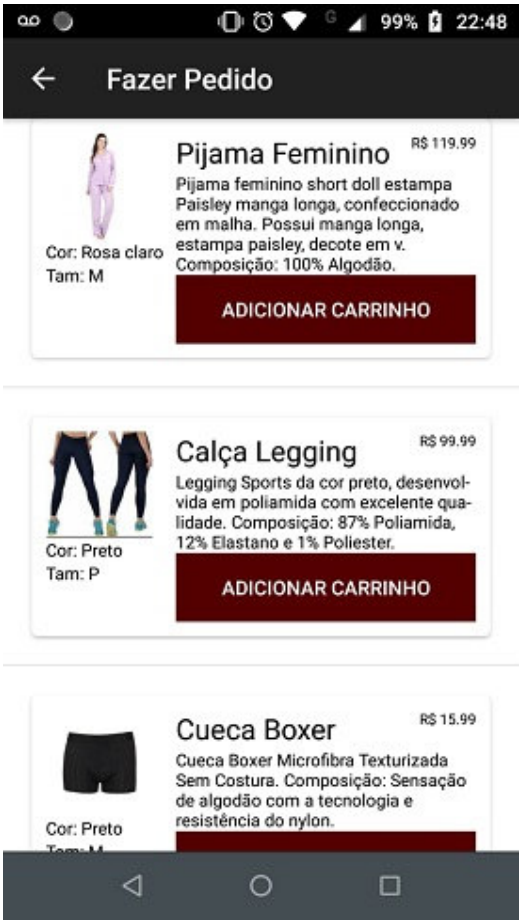

Figura 8 - Tela de Produtos

Ao pressionar o botão adicionar carrinho, o produto é adicionado a um carrinho de compras que será criado. Caso o cliente desejar adquirir uma quantidade superior a um, o botão adicionar carrinho deve ser pressionado conforme a quantidade necessária. Desta forma sua quantidade no carrinho será incrementada.

Após inseridos os produtos no carrinho de compras, o cliente volta ao menu de *dashboard*, utilizando o botão disponível na *toolbar* (Figura 9), em todas as telas do sistema.

Figura 9 - Botão de voltar activity

Para acessar o carrinho de compras e visualizar o resumo de sua compra, é necessário clicar no botão disponível na tela de *dashboard*. A Figura 10 apresenta a tela do carrinho de compras.

Figura 10 - Tela do carrinho de compras

Na tela apresentada na Figura 10, os campos valor total e quantidade, são atualizados conforme o cliente for alocando ou removendo itens do carrinho. Para remover todos os itens inseridos, basta clicar na opção esvaziar e uma mensagem de confirmação será apresentada na tela, solicitando se o usuário tem certeza da operação que está sendo realizada.

Para remover apenas um item do carrinho, o cliente deve pressionar por 1 ou 2 segundos em cima do produto desejado até ser apresentada uma mensagem de confirmação da operação de exclusão, conforme Figura 11.

Figura 11 - Confirmação de exclusão de item

No canto inferior direito da tela da Figura 11, do carrinho de compras, há um botão flutuante que, ao ser clicado, será aberta a tela de produtos, na qual o usuário poderá adicionar itens conforme a necessidade.

Ao clicar no botão confirmar no rodapé da tela de carrinho de compras, o cliente será redirecionado para a tela de finalização do pedido (Figura 12). Nesta tela, o cliente informará se a forma de pagamento será em dinheiro, cartão de crédito ou débito, cheque ou nota promissória.

Centralizado na tela, está o campo para inserção do código do cupom de desconto. Após inserir o código, o cliente deve clicar em verificar validade para verificar se o cupom está ativo e, caso estiver o valor do desconto será apresentado logo abaixo ao código e o valor líquido do pedido será atualizado, caso contrário, será atribuído o valor 0,00 para o campo.

Figura 12 - Tela de Finalizar Pedido

Após inseridas as informações, o cliente clica em enviar o pedido e a mensagem de confirmação para gerar o pedido aparece na tela confirmar o pedido. Após a confirmação será retornada outra mensagem informando o código do pedido gerado (Figura 13), caso contrário, o aplicativo retorna para a tela da Figura 12.

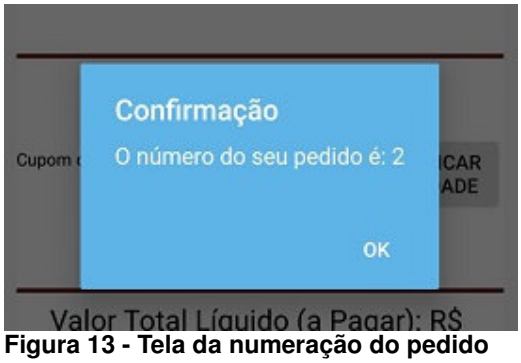

Figura 13 - Tela da numeração do pedido

Após clicar no botão ok, o usuário é direcionado para a tela de *dashboard* (Figura 7). Nesta tela há a opção para que o cliente acompanhe o *status* dos seus pedidos. O administrador do sistema será responsável por efetuar a modificação de

status do pedido. Os pedidos podem estar com *status* aberto, em separação, a caminho, entregue e cancelado. A Figura 14 apresenta pedidos de um determinado cliente, com o *status* em aberto.

Figura 14 - Tela de Meus Pedidos

Ainda na tela de *dashboard*, há a tela de contato, com informações referentes a empresa e a tela de localização da empresa via API Google Maps.

No aplicativo para uso do funcionário administrador do sistema, há algumas funcionalidades para uso e controle de determinadas tarefas no sistema. Ao abrir o sistema, as opções são listadas em forma de lista, conforme apresentadas na Figura 15.

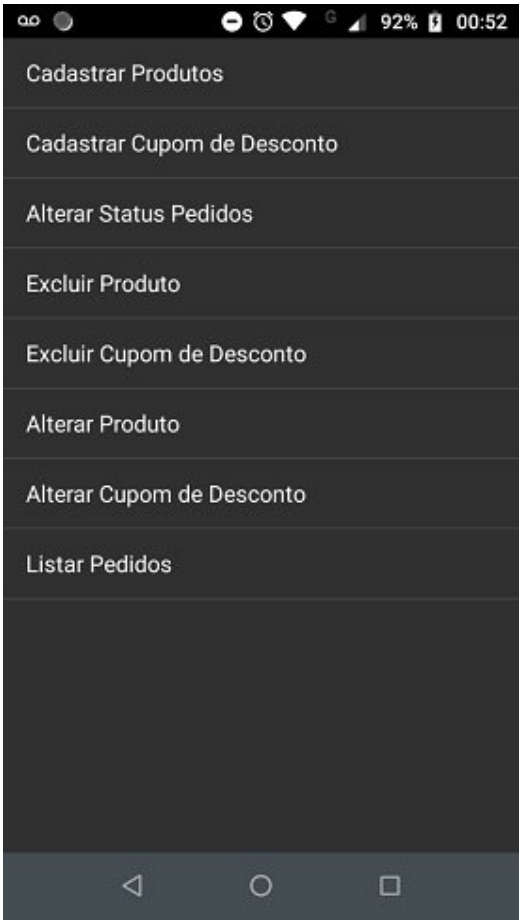

Figura 15 - Tela principal do funcionário administrador

Nas opções de cadastrar, seja cadastro de produto ou cupom de desconto, o funcionário administrador insere informações necessárias para que as informações sejam listadas no aplicativo do cliente, os campos estão representados na Figura 16.

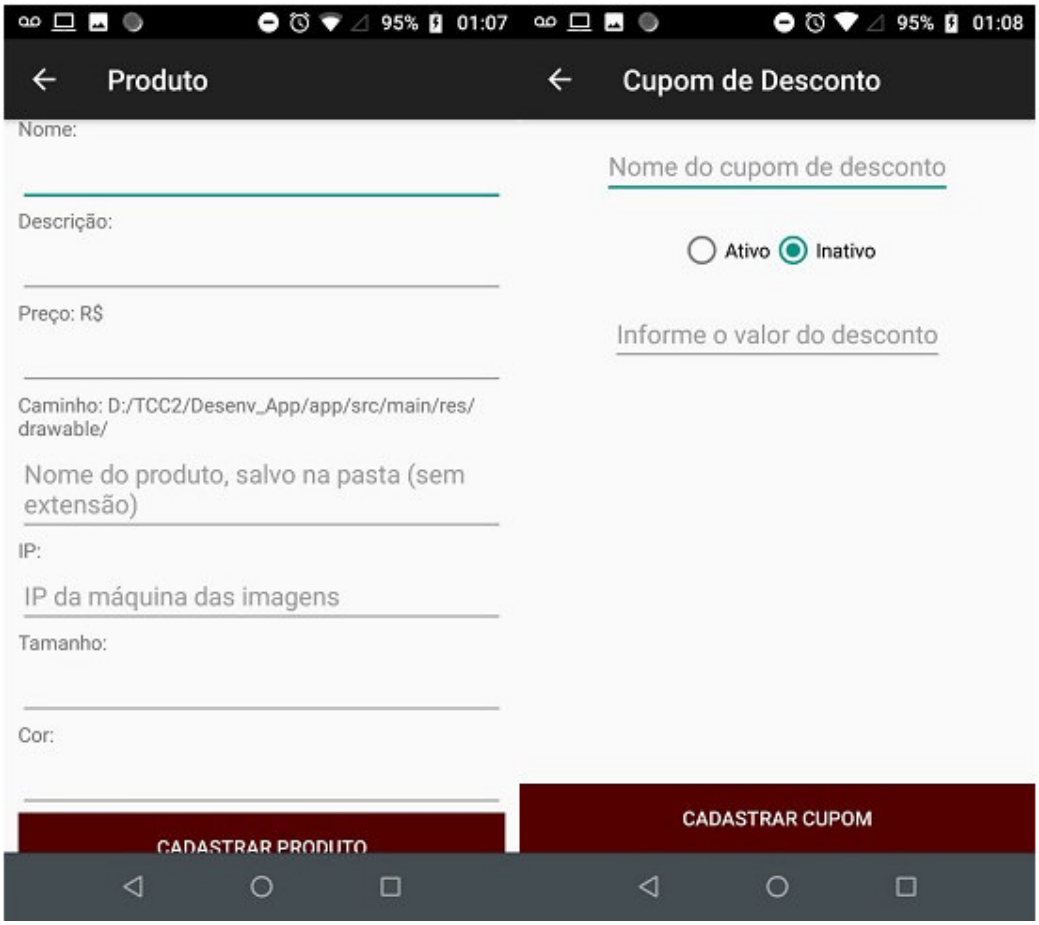

Figura 16 - Tela de cadastro de produto e cupom desconto

Para alterar o *status* do pedido, o funcionário acessa a opção de alterar *status* pedidos na tela principal, seleciona o pedido, e abrirá uma tela para escolher o novo *status* para o pedido.

Para excluir o produto ou cupom de desconto, o funcionário acessa as opções referentes na tela principal e procura a opção a ser excluída, e clica – se no botão excluir em cada registro.

A opção de alterar produto e cupom de desconto ao acessar essas opções, o usuário segura clicado por aproximadamente 1 a 2 segundos na opção desejada e altera as informações conforme desejar.

Na opção listar pedidos, será listado para o funcionário administrador todos os pedidos referentes a todos os cadastros de pessoas, separados pelo status correspondente.

4.4 IMPLEMENTAÇÃO DO SISTEMA

A seguir são apresentados alguns trechos de códigos do aplicativo do cliente, com a finalidade de mostrar como o mesmo foi organizado e implementado.

Os arquivos de projeto do Android, por padrão são exibidos no modo de visualização Android, podendo ser alterado pelo desenvolvedor para os modos *packages, project*, entre outros, conforme achar necessário. O padrão escolhido não afetará a estrutura de arquivos no local de armazenamento do disco, porém é subdividido por módulos e tipos de arquivos com intuito de facilitar a navegação. (Figura 17).

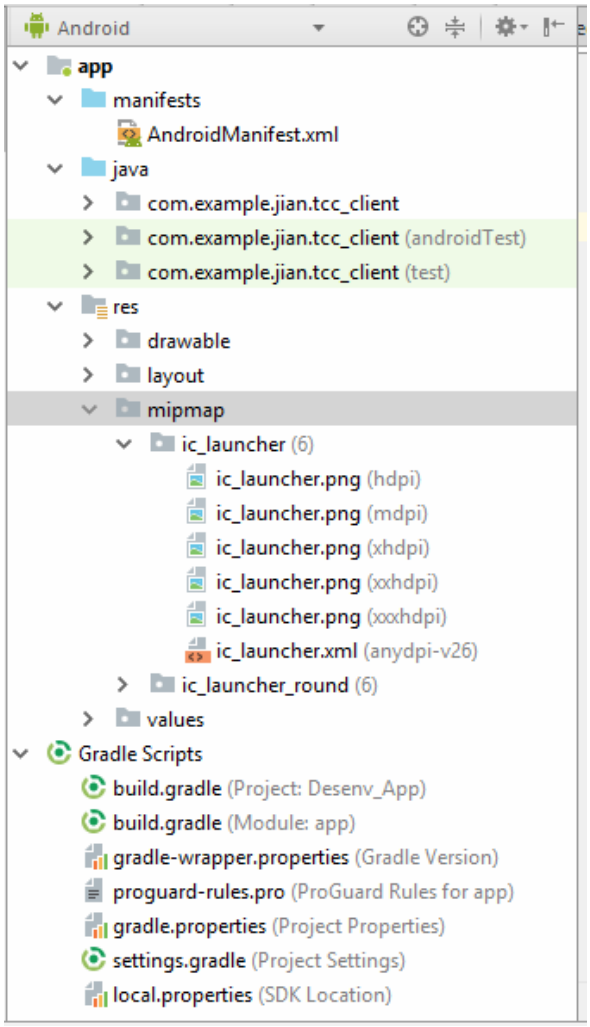

Figura 17 - Arquivos de projeto

No diretório *manifests*, é localizado o arquivo responsável por designar permissões e configurações para o aplicativo, sendo essencial para seu

funcionamento. A estrutura Java, é onde encontra-se a estrutura lógica para o funcionamento do aplicativo, contendo arquivos no formato Java, nesses arquivos são codificadas classes e regras de implementação, também conhecidas como *back-end*. Na pasta *res* na Figura 17, estão localizadas as imagens (*drawable*), as telas da estrutura XML (*layout*), também conhecidas como *front-end*, os ícones (*mipmap*) e valores estáticos para aplicação (*values*). Na estrutura de Gradle Scripts estão os arquivos utilizados na geração do aplicativo.

A Listagem 1, representa a codificação da função "*handleSignInResult*", da classe MainActivity, onde após autenticar-se com a conta do Google e obter sucesso, será criado um registro no banco de dados com as informações do nome e email.

```
public void handleSignInResult(GoogleSignInResult result) {
    if (result.isSuccess()) {
        GoogleSignInAccount account = result.getSignInAccount();
        tvNome.setText(account.getDisplayName());
        tvEmail.setText(account.getEmail());
        Pessoa pessoa = new Pessoa();
        pessoa.setEmail(account.getEmail());
        pessoa.setNome(account.getDisplayName());
        PessoaService service =
ServiceGenerator.createService(PessoaService.class);
        Call<Pessoa> call = service.validarUsuario(pessoa);
        call.enqueue(new Callback<Pessoa>() {
            @Override
            public void onResponse(Call<Pessoa> call, Response<Pessoa>
response) {
                if (response.isSuccessful()) {
                    Pessoa pessoa = response.body();
                    Util.gravarInteger(MainActivity.this,
Constatantes.PESSOA_ID, pessoa.getIdPessoa());
                    novoRegistro = pessoa.isPrimeiroAcesso();
                }
            }

            @Override
            public void onFailure(Call<Pessoa> call, Throwable t) {
                t.printStackTrace();
            }
        });

        email = account.getEmail();

        SharedPreferences spEmail =
PreferenceManager.getDefaultSharedPreferences(this);
        spEmail.edit().putString("email", email).apply();

    } else {
        goLogInScreen();
    }
}
```

Listagem 1 - Método handleSignInResult ao logar com a conta do google

A Listagem 2, refere-se à codificação do botão minha loja virtual, da classe MainActivity.

```
public void btnMenuTCCOnClick(View view) {
    if (novoRegistro) {
        Intent i = new Intent(this, PessoaActivity.class);
        startActivity(i);
        finish();
    } else {
        Intent i = new Intent(this, DashboardActivity.class);
        startActivity(i);
        finish();
    }
}
```

Listagem 2 - Codificação do botão para acessar a loja virtual

A classe ServiceGenerator, que usa a biblioteca *HyperText Transfer Protocol* (HTTP) Cliente para android e java, o Retrofit tem o intuito de criar um serviço *Representational State Transfer* (REST) com o Host API fornecido. A Listagem 3, refere a classe ServiceGenerator.

```
public class ServiceGenerator {
    private static final String API_HOST = "http://192.168.0.105:8080/";

    private static final OkHttpClient httpClient = new
OkHttpClient.Builder()
            .readTimeout(60, TimeUnit.SECONDS)
            .connectTimeout(60, TimeUnit.SECONDS)
            .writeTimeout(60, TimeUnit.SECONDS)
            .build();

    private static final Retrofit.Builder builder = new Retrofit.Builder()
            .baseUrl(API_HOST).client(httpClient)
            .addConverterFactory(GsonConverterFactory.create(new
GsonBuilder()
                    .setDateFormat("yyyy-MM-dd'T'HH:mm:ss").create()));

    private static final Retrofit retrofit = builder.build();

    public static <S> S createService(Class<S> serviceClass) {
        return retrofit.create(serviceClass);
    }
}
```

Listagem 3 - Codificação da classe ServiceGenerator

A Listagem 4, se refere ao serviço REST da interface de pedido.

```
public interface PedidoService {
    @GET("/rest/pedido/")
    Call<List<Pedido>> getAll();

    @GET("/rest/pedido/pessoa/{codigoPessoa}")
    Call<List<Pedido>> getByPessoa(@Path("codigoPessoa") Integer
codigoPessoa);

    @GET("/rest/pedido/{id}")
```

```
    Call<Pedido> getById(@Path("id") Long id);

    @POST("/rest/pedido/")
    Call<Pedido> save(@Body Pedido entity);

    @PUT("/rest/pedido/")
    Call<Pedido> update(@Body Pedido entity);

    @DELETE("/rest/pedido/{id}")
    Call<Void> delete(@Path("id") Long id);
}
```

Listagem 4 - Serviço REST da interface de pedido

A Listagem 5, é o método utilizado para listar os produtos, da classe ListarProdutoActivity.

```
public void listarProdutos(){
    try {
        final ProgressDialog pDialog = new ProgressDialog(this);
        pDialog.setCancelable(false);
        pDialog.setCanceledOnTouchOutside(false);
        pDialog.setTitle(R.string.aguarde);
        pDialog.show();

        // Cria o Serviço
        ProdutoService service =
ServiceGenerator.createService(ProdutoService.class);
        Call<List<Produto>> call = service.getAll();
        // Adicionar método de retorno ("callback")
        call.enqueue(new Callback<List<Produto>>() {
            @Override
            public void onResponse(Call<List<Produto>> call,
Response<List<Produto>> response) {
                pDialog.dismiss();
                // Resposta do servidor
                if (response.isSuccessful()) {
                    // Resposta com sucesso
                    adapter.atualizarLista(response.body());
                } else {
                    // Resposta com erro
                    Util.showMsgToast(ListarProdutoActivity.this, "Favor
verificar sua conexão");
                }
            }

            @Override
            public void onFailure(Call<List<Produto>> call, Throwable t) {
                // Erro no dispositivo
                Util.showMsgToast(ListarProdutoActivity.this, "Favor
verificar sua conexão");
            }
        });
    } catch (Exception e) {
        Toast.makeText(this, "Falha ao listar os produtos!" +
e.getMessage(),
                Toast.LENGTH_SHORT).show();
        e.printStackTrace();
    }
}
```

Listagem 5 - Método de listar produtos

E na Listagem 6, está apresentado a parte do adapter, que se refere a lista de produtos disponíveis no banco de dados.

```
public class ProdutosAdapter extends BaseAdapter {
    private Context c;
    private List<Produto> produtos = new ArrayList<>();

    public ProdutosAdapter(Context c) {
        this.c = c;
    }

    public void atualizarLista(List<Produto> produtos) {
        this.produtos.clear();
        this.produtos.addAll(produtos);
        this.notifyDataSetChanged();
    }

    @Override
    public int getCount() {
        return produtos.size();
    }

    @Override
    public Produto getItem(int position) {
        return produtos.get(position);
    }

    @Override
    public long getItemId(int position) {
        return produtos.get(position).getCodigo();
    }

    @Override
    public View getView(final int position, final View view, ViewGroup
viewGroup) {
        LayoutInflater inflater = (LayoutInflater)
c.getSystemService(c.LAYOUT_INFLATER_SERVICE);
        View elemento_lista = inflater.inflate(R.layout.elementos_lista,
null);

        //declara a criação dos componentes
        TextView tvNomeLista = (TextView)
elemento_lista.findViewById(R.id.tvNomeLista);
        TextView tvDescricaoLista = (TextView)
elemento_lista.findViewById(R.id.tvDescricaoLista);
        ImageView imgProduto = (ImageView)
elemento_lista.findViewById(R.id.imgProduto);
        TextView tvPreco = (TextView)
elemento_lista.findViewById(R.id.tvPreco);
        TextView tvTamanho = (TextView)
elemento_lista.findViewById(R.id.tvTamanho);
        TextView tvCor = (TextView)
elemento_lista.findViewById(R.id.tvCor);
        Button btnAddCarrinho = (Button)
elemento_lista.findViewById(R.id.btnAddCarrinho);

        //cursor percorre os componentes
        final Produto produto = getItem(position);
        Picasso.Builder builder = new Picasso.Builder(c);
```

```
        builder.listener(new Picasso.Listener() {
                 @Override
                 public void onImageLoadFailed(Picasso picasso, Uri uri,
Exception exception) {
                     exception.printStackTrace();
                     Log.e("PRODUTOS_ADAPTER", exception.getMessage());
                 }
             }
        );
        //seta nome e endereço nos componentes correspondentes
        tvNomeLista.setText(produto.getNome());
        tvDescricaoLista.setText(produto.getDescricao());
        builder.build().load(produto.getUrl()).into(imgProduto);
        tvPreco.setText("R$ " + produto.getPreco().toString());
        tvCor.setText("Cor: " + produto.getCor().toString());
        tvTamanho.setText("Tam: " + produto.getTamanho().toString());
        btnAddCarrinho.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
             final AlertDialog.Builder builder = new
AlertDialog.Builder(c);
             builder.setTitle("Atenção!");
             builder.setMessage("Deseja inserir esse item no carrinho?")
                     .setPositiveButton("Sim",  new
DialogInterface.OnClickListener() {
                     @Override
                     public void onClick(DialogInterface dialog, int id) {
                         Carrinho.getInstancia().addItem(produto);
                         Toast.makeText(c, "Produto inserido no carrinho de
compras",Toast.LENGTH_SHORT).show();
                     }
             })
                .setNegativeButton("Não", new
DialogInterface.OnClickListener() {
                     @Override
                     public void onClick(DialogInterface dialog,int id) {
                         dialog.cancel();
                     }
                 })
                 .show();
             }
        });
        return elemento_lista;
    }
}
```

Listagem 6 - Adapter de produtos

Na Listagem 7, temos o método para esvaziar carrinho da classe CarrinhoActivity.

```
public void esvaziarCarrinho(View view) {
    final AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Atenção!");
    builder.setMessage("Deseja esvaziar o carrinho?")
            .setPositiveButton("Sim",  new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int id) {
                    Carrinho.getInstancia().esvaziarCarrinho();
                    mostrarLista();
```

```
                    atualizarVlrTotal();
                    atualizarVolume();
                    Toast.makeText(CarrinhoActivity.this, "Carrinho de
compras foi esvaziado com sucesso!", Toast.LENGTH_SHORT).show();
                }
            })
            .setNegativeButton("Não", new DialogInterface.OnClickListener()
{
                @Override
                public void onClick(DialogInterface dialog,int id) {
                    dialog.cancel();
                }
            })
            .show();
}
```

Listagem 7 - Método para esvaziar o carrinho

A Listagem 8, refere-se ao método de excluir o item do carrinho de compras da classe CarrinhoActivity.

```
private AdapterView.OnItemLongClickListener longClickListener = new
AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, final
int position, long codigo) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(CarrinhoActivity.this);
        builder.setTitle("Atenção!");
        builder.setMessage("Deseja Excluir o item do carrinho?")
                .setPositiveButton("Sim",  new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        Carrinho.getInstancia().removerItem(position);
                        mostrarLista();
                        atualizarVlrTotal();
                        atualizarVolume();
                        Toast.makeText(CarrinhoActivity.this, "O item foi
removido do carrinho com sucesso!",Toast.LENGTH_SHORT).show();
                    }
                })
                .setNegativeButton("Não", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int id) {
                        dialog.cancel();
                    }
                })
                .show();
        return false;
    }

};
```

Listagem 8 - Método de excluir item do carrinho

Na Listagem 9, temos a classe Carrinho, onde temos os métodos para adicionar itens ao carrinho, contar o número de itens inseridos, realiza seus devidos

cálculos de valor total, desconto e retornar para ser utilizado em outras telas.

```
public class Carrinho{
    private static Carrinho instancia;
    private final List<PedidoItem> itens = new ArrayList<>();
    private Double valorDesconto = 0.0;
    private Pedido pedido;
    private Carrinho() {}

    public static Carrinho getInstancia() {
        if (instancia == null) {
            instancia = new Carrinho();
        }
        return instancia;
    }

    public void addItem(Produto produto) {
        PedidoItem pedidoItem = null;
        for (PedidoItem pi : itens) {
            if (pi.getProduto().equals(produto)) {
                pedidoItem = pi;
                break;
            }
        }
        if (pedidoItem == null) {
            pedidoItem = new PedidoItem();
            pedidoItem.setValorUnitario(produto.getPreco());
            pedidoItem.setQuantidade(1);
            pedidoItem.setProduto(produto);
            itens.add(pedidoItem);
        } else {
            pedidoItem.setQuantidade(pedidoItem.getQuantidade() + 1);
        }
    }

    public void esvaziarCarrinho() {
        itens.clear();
        valorDesconto = 0.0;
    }

    public List<PedidoItem> getItens() {
        return itens;
    }

    public Integer getNumeroItens() {
        Integer total = 0;
        for (PedidoItem pedidoItem : itens) {
            total += pedidoItem.getQuantidade();
        }
        return total;
    }

    public Double getValorDesconto(){
        return valorDesconto;
    }

    public void setValorDesconto(Double valorDesconto) {
        this.valorDesconto = valorDesconto;
    }

    public Double getTotal() {
```

```
        Double total = 0.0;
        for (PedidoItem pedidoItem : itens) {
            if (pedidoItem.getValorUnitario() != null) {
                total += pedidoItem.getValorUnitario() *
pedidoItem.getQuantidade();
            }
        }
        return total;
    }

    public Double getTotalLiquido() {
        return getTotal() - valorDesconto;
    }

    public String getValorDescontoFormatado() {
        Double valorDesconto = getValorDesconto();
        return new DecimalFormat("#,##0.00").format(valorDesconto);
    }

    public String getTotalFormatado() {
        Double total = getTotal();
        return new DecimalFormat("#,##0.00").format(total);
    }

    public String getTotalLiquidoFormatado() {
        Double totalLiquido = getTotalLiquido();
        return new DecimalFormat("#,##0.00").format(totalLiquido);
    }

    public void removerItem(int indice) {
        itens.remove(indice);
    }
}
```

Listagem 9 - Classe Carrinho

Na Listagem 10, temos uma lista personalizada através do código XML, para mostrar todos os produtos, organizados em CardViews, também foi usado TextView para componentes de texto, ImageView para as imagens e Button para chamar a funcionalidade a ser executada.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp">

    <android.support.v7.widget.CardView
        android:id="@+id/card_view"
        android:layout_gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:elevation="4dp"
        app:cardCornerRadius="4dp"
        android:layout_margin="10dp">
```

```
<RelativeLayout
    android:padding="10dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_height="40pt"
        android:layout_width="40pt"
        android:layout_marginRight="16dp"
        android:contentDescription="Imagem"
        android:id="@+id/imgProduto" />

    <TextView
        android:layout_toRightOf="@id/imgProduto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvNomeLista"
        android:textColor="@android:color/black"
        android:textSize="20sp" />


    <TextView
        android:textColor="@android:color/black"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:gravity="bottom"
        android:id="@+id/tvPreco"
        android:textSize="10sp" />

        <TextView
            android:layout_toRightOf="@id/imgProduto"
            android:layout_below="@id/tvNomeLista"
            android:textColor="@android:color/black"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/tvDescricaoLista"
            android:textSize="12sp" />

    <TextView
        android:layout_marginRight="5dp"
        android:layout_below="@id/imgProduto"
        android:textColor="@android:color/black"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvCor"
        android:textSize="12sp" />

    <TextView
        android:layout_marginRight="5dp"
        android:layout_below="@id/tvCor"
        android:textColor="@android:color/black"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvTamanho"
        android:textSize="12sp" />

    <Button
        android:layout_toRightOf="@id/imgProduto"
        android:layout_below="@id/tvDescricaoLista"
```

```
            android:id="@+id/btnAddCarrinho"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:textColor="#FFF"
            android:text="Adicionar Carrinho"
            android:onClick="addCarrinho"/>
    </RelativeLayout>
    </android.support.v7.widget.CardView>
</LinearLayout>
```

Listagem 10 - Codificação em XML

As demais classes da aplicação por parte do cliente e do usuário administrador, seguem o mesmo padrão de codificação dos códigos das listagens 01 a 10.

5 CONCLUSÃO

Este trabalho teve como objetivo propor uma solução para dispositivos móveis Android, para que revendedores, possam antecipar seus pedidos, antes da visita de seu representante na região, visando promover o cadastro de novos clientes e, para a empresa que receberá estes pedidos, garantir a entrega do produto de interesse do revendedor, bem como organizando-se melhor em seus processos internos, enquanto empresa.

As tecnologias utilizadas para consumir web service com retrofit, teve facilidade de integração, velocidade e eficiência na transmissão de dados. As ferramentas utilizadas garantiram na implementação o desenvolvimento, tendo em vista, sendo uma das ferramentas mais utilizadas no mercado de trabalho.

Como sugestão para trabalho futuros, destaca-se a integração com *Enterprise Resource Planning* (ERP) utilizado pela empresa que se propôs a colaborar com o processo de desenvolvimento e fazer a integração de grade do produto, composto por cor e tamanho. Aperfeiçoar a usabilidade da ferramenta para o funcionário administrador, inserindo novas funcionalidade e relatórios.

REFERÊNCIAS

ALBERTIN, A. L. **Pesquisa FGV-EAESP de Comércio Eletrônico no Mercado Brasileiro**, 18º edição, 2016, livraria da FGV-EAESP, (55 11) 3799-7790.

AMADOR, João Gabriel. **Estudo mostra crescimento no uso de dispositivos móveis e domínio Android.** Disponível em: < http://www.correiobraziliense.com.br/app/noticia/tecnologia/2015/01/17/interna_tecnologia,466691/estudo-mostra-crescimento-no-uso-de-dispositivos-moveis-e-dominio-andr.shtml>

ANDROID DEVELOPER. **Arquitetura da plataforma**. 2017. Disponível em: < https://developer.android.com/guide/platform/index.html>. Acesso em: 14 mai. 2017.

BORDIN. M. V.; **Introdução a Arquitetura Android.** Sociedade Educacional Três de Maio, 2012.

CINTRA, André. **4 motivos do porque ter um aplicativo para sua empresa** Disponível em: <http://www.postdigital.cc/blog/artigo/4-motivos-do-porque-ter-um-aplicativo-para-a-sua-empresa>. Acesso em 24 mar 2017.

DIAS, Raphael. **Desenvolvimento Android: Tudo o Que Você Precisa Saber Para Começar.** Disponível em: < http://producaodejogos.com/desenvolvimento-android/> Acesso em 01 mai. 2017.

FELIPE, Alex. **Consumindo API REST no Android com Retrofit em Kotlin – Parte 1**. Disponível em <https://medium.com/collabcode/consumindo-api-rest-no-android-com-retrofit-em-kotlin-parte-1-5e752ab8a877>. Acesso em 04 out 2018.

FERREIRA, V. H. A; JENSEN. V. S.; **Relações virtuais de consumo: Perspectivas de Direitos no e-Commerce.** Universidade Federal de Santa Maria, 2002.

FINKELSTEIN. Maria Eugenia Reis. **Direito do comércio eletrônico**. 2. ed. Rio de Janeiro: Elsevier. 2011.

OGLIARI BRITO, Ricardo da Silva, Robison Cris. **Android - Do Básico ao Avançado**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2014.

MEYER, Maximiliano. **A história do Android.** Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android> Acesso em 07 mai 2017.

KLIER, Katrina**. Conheça a Geração do Celular: A Geração Y e o Marketing Móvel**. Disponível em <https://www.benchmarkemail.com/br/blogs/detail/conheca-a-geracao-do-celular-a-geracao-y-e-marketing-movel> **Acesso em 07 mai 2017**.

PRESSMAN, Roger. **Engenharia de software**. Rio de Janeiro: McGraw-Hill, 2008.

SCHOLZ, Simon; VOGEL, Lars; WEISER, David. **Using Retrofit 2.x as REST client – Tutorial.** Disponível em: <http://www.vogella.com/tutorials/Retrofit/article.html>. Acesso em 02 out. 2018.

THIENGO, Vinícius. **Library Retrofit 2 no Android.** Disponível em: <https://www.thiengo.com.br/library-retrofit-2-no-android>. Acesso em 12 out. 2018.

VIEIRA, Bruno. **Retrofit 2: o que devemos saber.** Disponível em: <https://imasters.com.br/android/retrofit-2-o-que-devemos-saber>. Acesso em 11 out. 2018.