UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
III ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

JEANFRANCISCO POLLO STRAPASSON

APLICATIVO PARA COMPRAS EM SUPERMERCADOS

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2015

JENFRANCISCO POLLO STRAPASSON

APLICATIVO PARA COMPRAS EM SUPERMERCADOS

Trabalho de Conclusão de Curso, apresentado ao III Curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, campus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Robison Cris Brito.

PATO BRANCO
2015

Por

Jeanfrancisco Pollo Strapasson

Esta monografia foi apresentada às 16h00 do dia 04 de novembro de 2015 como requisito parcial para a obtenção do título de ESPECIALISTA, no III Curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Msc. Robison Cris Brito
Orientador
UTFPR – Campus Pato Branco

Profa. Dra. Beatriz Borsoi
Banca
UTFPR – Campus Pato Branco

Profa. Msc. Luciia Yoshie Araki
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Coordenador do curso de Especialização
UTFPR – Campus Pato Branco

DEDICATÓRIA

Dedico este trabalho aos meus familiares e amigos que me acompanharam nesta jornada que me privei várias vezes para a minha dedicação a este conhecimento.

AGRADECIMENTOS

Primeiramente agradecer a Deus por me proporcionar esta oportunidade, saúde e conhecimento para chegar até este fase da minha vida.

Aos meus familiares e amigos que me incentivaram a iniciar e a continuar esta jornada.

RESUMO

STRAPASSON, Jeanfrancisco Pollo. Aplicativo para compras em supermercados. 2015. XX f. Monografia de especialização III Curso de Especialização em Java. Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

Aplicativos para compras em supermercados permitem a utilização dos recursos mobile em uma aplicação que disponibiliza serviços relacionados à compra de produtos. Visando utilizar recursos da tecnologia Java e a plataforma Android foi desenvolvido um aplicativo de compras para supermercados. Este aplicativo, possibilitará ao usuário realizar solicitações de compras, informando quais produtos são desejados, as suas devidas quantidades e a forma de pagamento escolhida. Com estas informações o funcionário do mercado será responsável por separar o pedido e realizar a entrega da compra.

Palavras-chave: Anroid. Linguagem Java.

LISTA DE FIGURAS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

API	*Application Programming Interface*
CRUD	*Criate, Read, Update, Delete*
DAO	*Data Access Objetct*
EJB	*Enterprise Java Beans*
HTML	*Hipertext Markup Language*
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
J2EE	*Java Enterprise Edition*
JDBC	*Java Database Connectivity*
JEE	*Java Enterprise Edition*
JMS	*Java Message Service*
JMX	*Java Management Extensions*
JSF	*JavaServer Faces*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*
GPS	*Global Positioning System*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações inicias, objetivos e a justificativa, onde nas considerações iniciais é abordado o início do comércio, dos mercados e supermercados, tecnologias computadores e celulares. É finalizado com a organização do texto pela apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

Desde o início dos tempos o homem procurou suprir suas necessidades básicas por meio da pesca, caça, agricultura produção de tecidos e entre outras atividades para obtenção e produção de bens de consumo próprio. Com a melhora na forma de produzir esses itens, surgiram então as sobras que serviam com um tipo de moeda de troca por produtos diferentes, mas que também serviam para suprir necessidades essenciais. Surge então uma forma de comércio na qual os próprios itens eram as moedas de troca.

Não demorou para que surgissem as moedas (dinheiro) e bancos, centros de venda de alimentos, tecidos e utensílios que eram necessários à sobrevivência e bem-estar do homem. Era uma forma de mercado na qual vários tipos de produtos eram comercializados em um só estabelecimento, que conforme cresciam especializavam mais em um segmento, como alimentos ou tecidos e roupas.

Com o tempo surgiram as indústrias de roupas e alimentos, fruto da revolução industrial que fortaleceu esses mercados e seus segmentos, aumentando a procura pelo consumidor surgindo então os grandes supermercados. Surge assim, uma forma de o próprio consumidor escolher os itens que iria adquirir. Os itens então eram organizados no mesmo local e categorizados, facilitando a busca pelos produtos dentro de grandes estabelecimentos.

A tecnologia com a sua evolução também foi um forte aliado ao crescimento dos supermercados. Com o surgimento do computador telefone e Internet foi possível controlar de forma mais ágil o estoque desses grandes supermercados. Com o telefone, a comunicação se tornou mais fácil entre consumidor e vendedor, sendo possível realizar compras sem a presença no ambiente físico e a entrega pelo próprio supermercado.

O crescimento da Internet possibilitou aumentar ainda mais a comunicação do comprador com o vendedor, a busca pela informação começa a se tornar cada vez mais fácil o acesso. Com isso surge então um tipo de comércio eletrônico o *e-commerce*, uma forma de

disponibilizar os itens de grandes centros em um site, que com a facilidade de acesso poderia atingir muito mais clientes de uma forma mais rápida. O surgimento de dispositivos móveis, torna o acesso ao *e-commerce* mais fácil e rápido e com maior quantidade devido ao número de pessoas que utilizam um dispositivo portátil e pela forma como é organizado a busca dos produtos pela internet.

Uma das formas de auxiliar no comércio voltado para supermercado é por meio de aplicativos mobile, que sejam desenvolvidos especificamente para realizar o processo de compra ao consumidor. Este trabalho apresenta o desenvolvimento de um aplicativo mobile que utiliza tecnologias como Java para a construção de um software de compra em supermercados.

1.2 OBJETIVOS

O objetivo geral deste trabalho refere-se ao resultado gerado ao longo do desenvolvimento, software. Os objetivos específicos são as complementações do aplicativo.

1.2.1 Objetivo Geral

Implementar um aplicativo mobile para realizar compras em supermercados.

1.2.2 Objetivos Específicos

- Implementar um servidor para disponibilizar informações como produtos, preços e acompanhamento da compra;
- Fornecer uma maneira de realizar compras e acompanhamento do processo de compra e entrega;
- Implementar dois aplicativos um para o cliente e outro para o funcionário do supermercado, identificando e modelando os requisitos de cada aplicativo.

1.3 JUSTIFICATIVA

A escolha de desenvolver um aplicativo que possa realizar compras em supermercados, possibilita ao seu usuário realizar compras de produtos que periodicamente

são de sua necessidade sem deslocar-se até o supermercado. Utilizando métodos que antes eram apenas encontrados em produtos que podem demorar semanas para chegar. Assim este trabalho, exemplifica uma das formas de utilizar a tecnologia Java no desenvolvimento de software e aplicativos mobile.

A utilização de smartphones teve um aumento significativo nos últimos anos, sendo utilizado como forma de comunicação, internet Bank, compras, entre outros afins, através de aplicativos.

Um aplicativo de *e-commerce* que com o uso de tecnologias que permitem o acesso de clientes a produtos disponibilizados por grandes e pequenos supermercados tende a facilitar o processo de compra. Os clientes podem, por exemplo, evitar o deslocamento até o supermercado, evitar filas e ter os produtos que adquiriu entregues em casa.

A utilização de celulares em 2012 chega a 1,1 bilhão de usuários no mundo todo, 66% do mercado são Android. (GAZETADOPOVO, 2013).

No Brasil a soma do número de lojas das 50 maiores redes de supermercados chega a 4761 sendo possível um alto número de usuários que podem ter a possibilidade de utilizar o software para realizar compras. (ABRIL, 2013)

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos, sendo este o primeiro que apresenta a ideia e contexto do sistema, incluindo os objetivos e justificativa.

O Capítulo 2 é composto pelo referencial teórico que fundamenta a proposta conceitual do software construído. Sendo centrado no funcionamento de um supermercado e como funciona, basicamente um software de gerenciamento de estoque e emissão de notas e histórico de *smartphones* sua utilização com a plataforma Android.

No Capítulo 3 estão os materiais e método utilizado para o desenvolvimento deste trabalho. Os materiais se referem às ferramentas e tecnologias utilizadas e o método contém as atividades realizadas.

O Capítulo 4 contém o sistema desenvolvido com exemplos de códigos, telas, modelagem utilizada para a implementação. Os códigos e telas são a exemplificação pela apresentação do sistema descrevendo suas funcionalidades. A modelagem é apresentada em forma de documentos de análise e projeto.

No capítulo 5 está a conclusão com as considerações finais.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico deste trabalho, que é centrado em aplicações em Android, Java e o funcionamento de um supermercado. O resultado deste trabalho é um aplicativo implementado com essas duas tecnologias, que poderá servir como um auxílio no funcionamento do supermercado.

2.1 FUNDAMENTOS DE MERCADO/SUPERMERCADO

Hipermercados, supermercados e mercearias são empresas criadas para unir o útil ao conforto. São nessas empresas que você encontra vários tipos de produtos que, em sua maioria, fazem parte do dia a dia de uma casa. Alimentos, bebidas, produtos de limpeza, produtos de beleza, conveniência, produtos para animais. Todos esses itens você pode adquirir em um supermercado. Tudo isso foi reunido para facilitar as compras do cliente. (SEBRAE MINAS, 2013).

Desta forma podemos entender que hipermercados, supermercados e mercearias basicamente tende a funcionar da mesma forma, diferenciando o tamanho e local de atuação dentro das cidades.

Podemos entender que um Software de gerenciamento destes estabelecimentos podem se classificar como Enterprise Resource Planning (ERP’s) como coloca o autor (DAVENPORT, 2002) "conhecidos igualmente como sistemas integrados de gestão (ou ERP), são, na verdade, pacotes de aplicativos de computador que dão suporte à maioria das necessidade de informação de uma empresa".

Podemos definir algumas das mais usuais e comuns destas funções dos pacotes que contempla um ERP, em Haberkorn (1999) podemos encontrar os principais módulos:

- Compras;
- Faturamentos;
- Estoque;
- Custos;
- Planejamento e controle da produção;
- Financeiro;
- Contabilidade;
- Livros Fiscais;

- Ativo Fixo;
- Folha de Pagamento

Endentemos então que um ERP pode realizar dês de o controle de estoque, até o faturamento do produto no caixa para o cliente. Sendo possível realizar a emissão de Cupom/Nota do Consumidor, controle de entrada/saída dos itens em estoque, entrada de notas.

2.2 DISPOSITIVOS MÓVEIS ANDROID

O Android é uma plataforma para desenvolvimento e execução de programas para dispositivos móveis, robusta e de fácil utilização/aprendizagem. Foi construída inicialmente por uma pequena empresa de Palo Alto (Califórnia nos EUA), chamada Android Inc. A mesma foi adquirida pela Google em 2005 (OGLIARI; BRITO, 2014).

A grande aposta do Android são os novos aparelhos celulares, mais conhecidos como *smartphones*, que são celulares com grande capacidade de processamento e que integram vários recursos como alta conectividade com a internet como *Global Positioning System* (GPS), sensores e telas sensíveis ao toque (OGLIARI; BRITO, 2014).

Desta forma, os *smartphones* podem se tornar uma grande ferramenta para acesso aos mais variados aplicativos como mapas, que podem ser visualizados de vários locais por serem terem acesso à internet e GPS.

O Android é um projeto de código aberto e por este motivo, surgem novas versões em ritmo acelerado. Embora seja comum atribuir que Android foi desenvolvido exclusivamente pela Google, na verdade ele foi desenvolvido pela Android Inc., adquiria em 2005 pela Google (OGLIARI; BRITO, 2014).

Os aplicativos que são desenvolvidos para a plataforma Android, podem ser publicados na loja virtual Play Store, sendo possível a compra e *download* desses aplicativos.

3 MATERIAIS E MÉTODOS

Neste capítulo serão apresentadas as ferramentas e métodos utilizados para o desenvolvimento deste trabalho. Os matérias se referem as ferramentas para modelagem e implementação do sistema. O método contém os principais procedimentos utilizados para o desenvolvimento do software.

3.1 MATERIAIS

As ferramentas e as tecnologias utilizadas para as atividades de modelagem, implementação e execução são:

a) ClickCharts Diagram Flowchart Software para modelagem do sistema;

b) Eclipse Luna 4.4.0 para a *Integrated Development Environment* (IDE) de desenvolvimento do servidor;

c) Java como linguagem de programação;

d) Gson como biblioteca para comunicação de dados entre aplicativo e servidor.

e) Android Studio 1.1.0 para a IDE de desenvolvimento do aplicativo mobile;

f) MySQL Server 5.6 para banco de dados;

g) Glassfish 4 como servidor da aplicação.

3.1.1 ClickCharts Diagram

ClickCharts Diagram é uma ferramenta que tem como funções a criação de modelo, processos em representação visual, organizações complexas, criações de diagramas de fluxos de dados. A Figura 1 apresenta a interface dessa ferramenta.

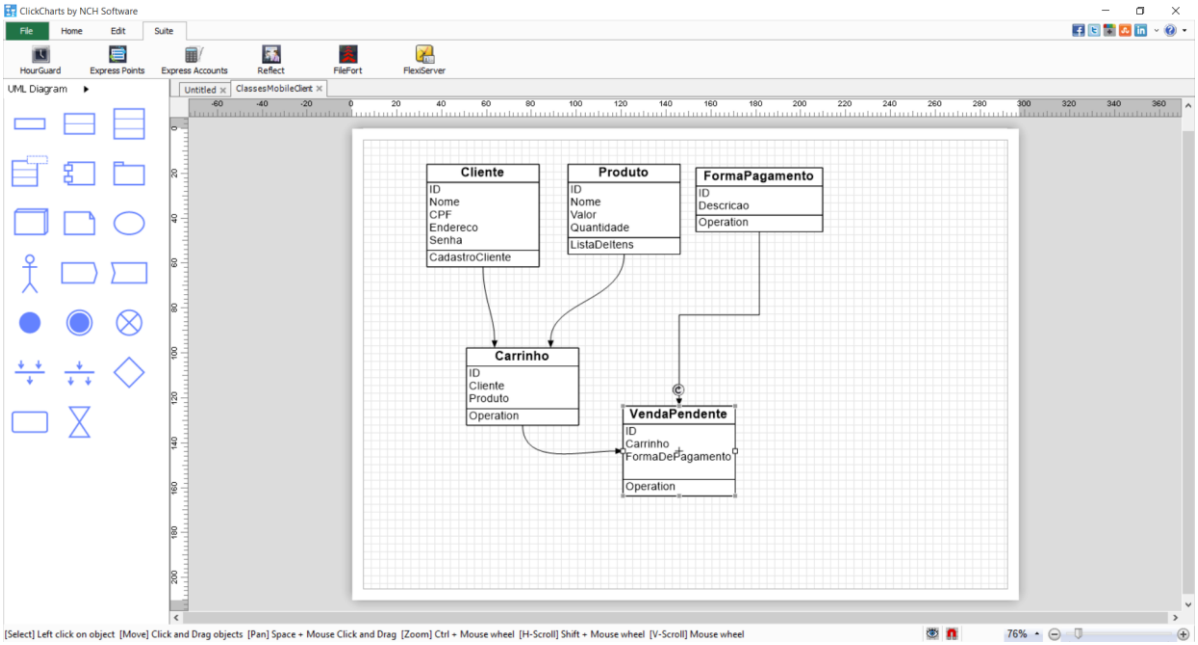

Figura 1 - ClickCharts Diagram

3.1.2 Eclipse Luna 4.4.0

Eclipse (ECLIPSE, 2015) é uma IDE de desenvolvimento de software multi-linguagens. Com o uso de *plugins* é possível desenvolver, além da linguagem Java, C C++, PHP e Ruby. A Figura 2 apresenta a dela principal da IDE Eclipse.

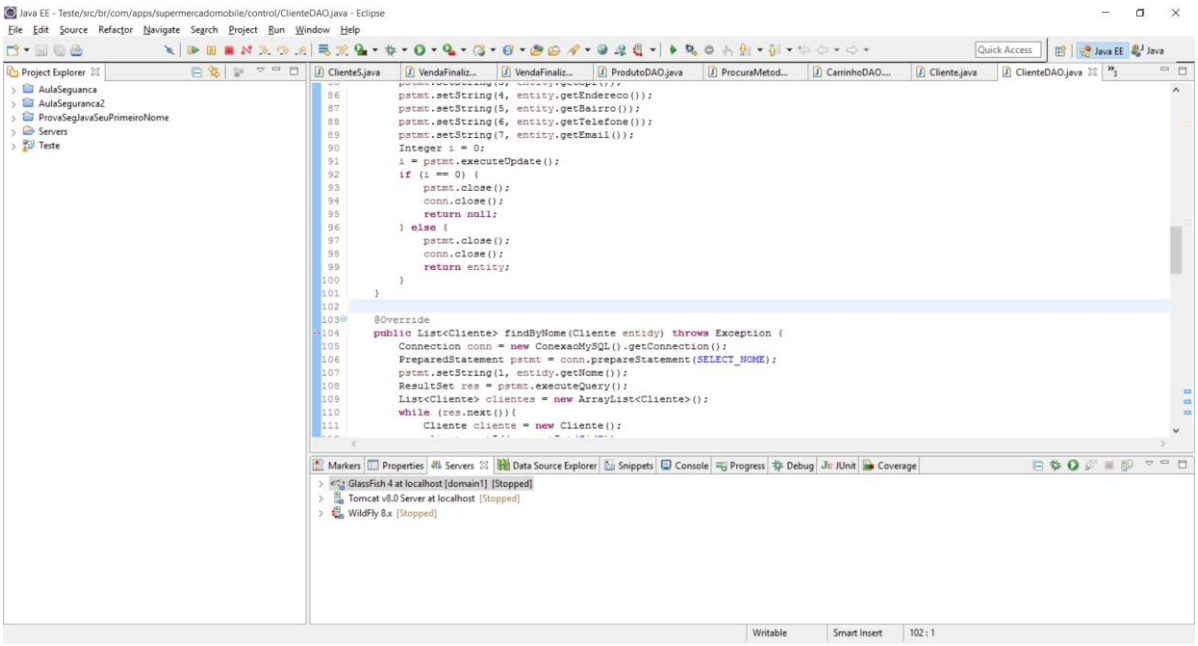

Figura 2 - Tela inicial da IDE Eclipse

3.1.3 Java como linguagem de programação

A tecnologia Java foi lançada tendo como principal foco aplicativos para dispositivos como televisores e equipamentos domésticos. Com o passar dos anos e evolução da tecnologia, deixou de ser aplicada apenas para dispositivos de pequeno porte e foi iniciada a utilização em aplicações como servidores robustos e de alta segurança.

A linguagem Java é considerada simples porque permite o desenvolvimento de aplicativos para diferentes sistemas operacionais e arquiteturas de hardware, sem que o programador tenha que se preocupar com detalhes de infra-estrutura. Dessa forma, o programador consegue desempenhar seu trabalho de uma forma mais produtiva e eficiente. (MENDES, 2009).

Entre suas vantagens, a tecnologia java permite um portabilidade dos programas desenvolvidos em grande escala, isso devido a *Java Virtual Machine* (JVM), permitindo que o mesmo aplicativo seja executado em plataformas distintas, como Windows, Linux e Mac OS.

Java permite a criação de aplicativos que implementam o conceito *multithread*, incluindo sofisticados mecanismos de sincronização entre processos. O multithreading é uma técnica de programação concorrente, que permite projetar e implementar aplicações paralelas de forma eficiente (MENDES, 2009).

3.1.4 GSON

Gson é uma biblioteca Java que pode ser usado para converter objetos Java em sua representação *JavaScript Object Notation* (JSON). Ele também pode ser usado para converter uma cadeia JSON para um objeto Java equivalente. Gson pode trabalhar com objetos Java arbitrários incluindo objetos pré-existentes que você não tem o código-fonte (GITHUB, 2015)

3.1.5 Android Studio

O Android Studio é uma IDE de desenvolvimento para a plataforma Android baseado no InteliJ Community, sendo possível realizar o desenvolvimento em Java *HyperText Markup Language* (HTML), realizar debugs, testes e empacotar os aplicativos (DEVELOPERS, 2015). A Figura 3 apresenta a tela da IDE de desenvolvimento da IDE Android Studio.

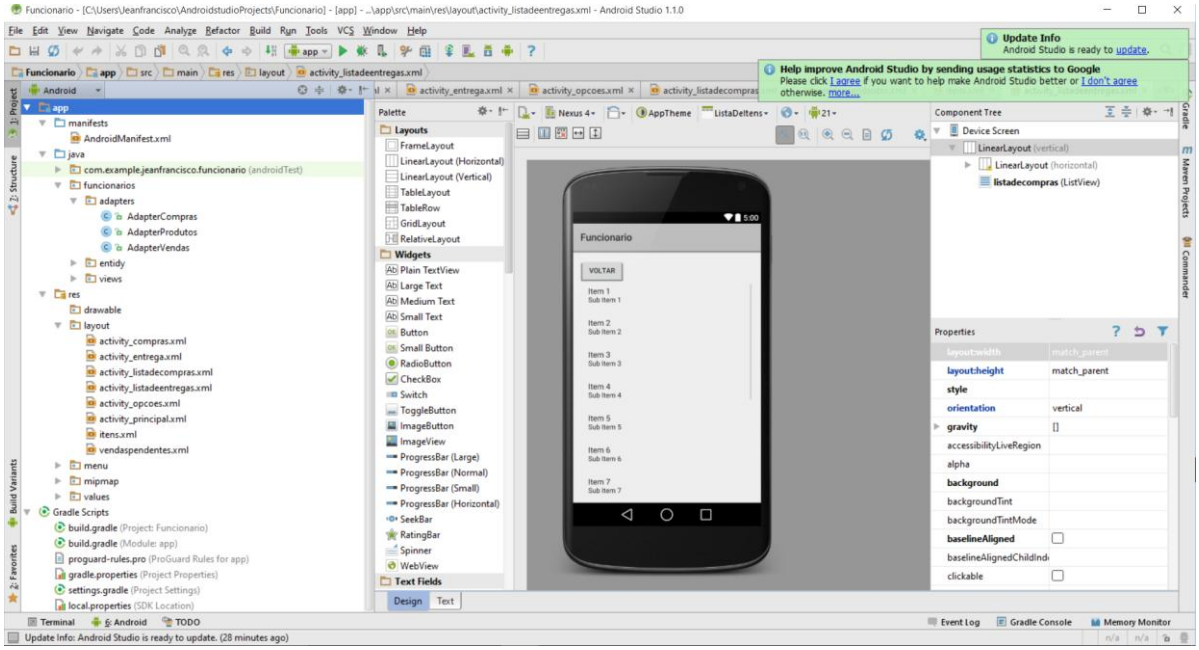

Figura 3 - Tela inicial da IDE Android Estudio

3.1.5 MySQL

O MySQL é um sistema de gerenciamento de banco de dados relacional, muito utilizado em servidores Web, por conter como principais características velocidade, flexibilidade e grande facilidade a integração a programação de sites dinâmicos.

É um software Open Source e sendo assim é possível usar e modificar o programa. Teve como um dos objetivos iniciais o desenvolvimento para gerenciamento de bancos grandes, de uma maneira muito mais rápida que as soluções existentes.

3.1.6 Glassfish 4

O GlassFish é um servidor de aplicação desenvolvido pela Sun Microsystems para a plataforma *Java Enterprise Edition* (J2EE), ele é um servidor gratuito e *open source*, porém também existe uma versão proprietária chamada GlassFish Enterprise Server. O J2EE disponibiliza padrões para contêineres Web e EJB (*Enterprise Java Beans)*. O GlassFish suporta todas as especificações da API (*Application Programming Interface)* Java, JDBC (*Java Database Connectivity*), RMI (*Remote Method Invocation*), JavaMail, JMS (*Java Message Service*), JMX (*Java Management Extensions*), bem como algumas especificações

específicas para componentes Java EE, como *servlets*, *portlets*, JSF (*JavaServer Faces*) e diversas tecnologias de *web services* (WIKI, 2015).

3.2 MÉTODO

O Processo Unificado surgiu como um processo popular iterativo para o desenvolvimento de software visando à construção de sistemas orientados a objetos (LARMAN, 2005).

Como explica Scott (2002), esse processo pode ser projetado para aprendizado e treinamentos contínuos não sendo necessário realizar treinamentos extensivos para se tornar produtivo.

Uma das características do processo unificado é organizar o trabalho dividindo quatro fases principais para o desenvolvimento, sendo elas: concepção, elaboração, construção e transição. Nos subseções a seguir, as quatro fases desse processo são explanadas.

3.2.1 Fase de Concepção

Na fase de concepção é definidas uma visão aproximada do software, pode se dizer que não são especificados os detalhes de uma aplicação, são realizados os casos de negócios do software e gerado um escopo com algumas estimativas vagas.

Segundo Scott (2002), o objetivo da fase de concepção é estabelecer a viabilidade do sistema proposto executando algumas tarefa. A primeira é definir o escopo do sistema. A segunda esboçar uma arquitetura candidata, formada pelas versões iniciais dos seis diferentes modelos. A terceira é identificar os riscos críticos e determinar quando e como o projeto os abordará. E a quarta, e última, iniciar a análise econômica do projeto, mostrando se o projeto vale a penas, com base em estimativas iniciais de custo, esforço, cronograma e qualidade do produto.

3.2.2 Fase de Elaboração

A fase de elaboração trata de definir uma visão refinada, comparada a visão da concepção, trata de uma resolução de altos riscos, e identificação da maioria dos requisitos, com um escopo mais realista.

Seguindo com o pensamento de Kendall Scott (2002), nesta fase são tratadas as

seguintes tarefas: a primeira seria capturar a maioria dos requisitos funcionais válidos. A segunda, seria de expandir a arquitetura candidata em uma base arquitetônica, que seria uma versão interna do sistema. A terceira, de abordar riscos significativos, de uma forma contínua. E a quarta, de finalizar a análise econômica do projeto e preparar um plano que contenha detalhes suficientes para orientar a construção.

3.2.3 Fase de Construção

Na fase de construção é definida a implementação de elementos restantes, que tenham um menor risco e que sejam mais fáceis e realizada a preparação para a implementação.

Scott (2002), explica que no desenvolvimento de um projeto, as tarefas executadas nesta etapa envolvem a construção do sistema de um modo iterativo e incremental, de uma forma que tenha certeza da viabilidade do sistema na forma executável.

3.2.4 Fase de Transição

Nesta fase acontece o encontro do software com o ambiente, para o seu uso em produção. Podendo ser considerado versões betas, nas quais podem ser destacados detalhes que faltaram no momento da implementação e de usabilidade do sistema com o usuário. Pode ser considerada uma das fases mais fáceis de serem tratadas, porém uma das mais importantes. É nessa fase que o cliente validará se o sistema atende às suas solicitações representadas por requisitos.

4 RESULTADOS E DISCUSSÃO

Neste capítulo é apresentado o resultado da realização deste trabalho. Os aplicativos foram desenvolvidos com a finalidade de realizar as compras em supermercados. Inicialmente o sistema é apresentado na Seção 4.1, em seguida, na Seção 4.2 é apresentada a sua modelagem. Na Seção 4.3 suas telas e funcionalidades. E na Seção 4.4 os exemplos de codificação gerada.

4.1 APRESENTAÇÃO DO SISTEMA

O servidor é responsável por disponibilizar os produtos para a compra ao aplicativo cliente, e responsável também por disponibilizar as compras realizadas para o aplicativo funcionário. O primeiro aplicativo é direcionado para o cliente, responsável por disponibilizar os itens e realizar as compras. O segundo, é direcionado para o funcionário, possibilitando efetivar as compras realizadas pelos clientes e efetuando a compra e as informações de forma de pagamento e entrega da mercadoria.

4.2 MODELAGEM DO SISTEMA

Os requisitos foram identificados a partir da necessidade de realizar as compras por um meio mais rápido sendo identificados para a modelagem da seguinte maneira:

a) Controlar clientes
   - Realizar cadastro de clientes;
   - Realizar *login* dos clientes;
   - Fazer pedido de compras;
   - Consultar *status* do pedido;

b) Controlar funcionários
   - Realizar *login*;
   - Validarcompra;
   - Entregar da compra;

Os requisitos são apresentados como casos de uso (Figura 4).

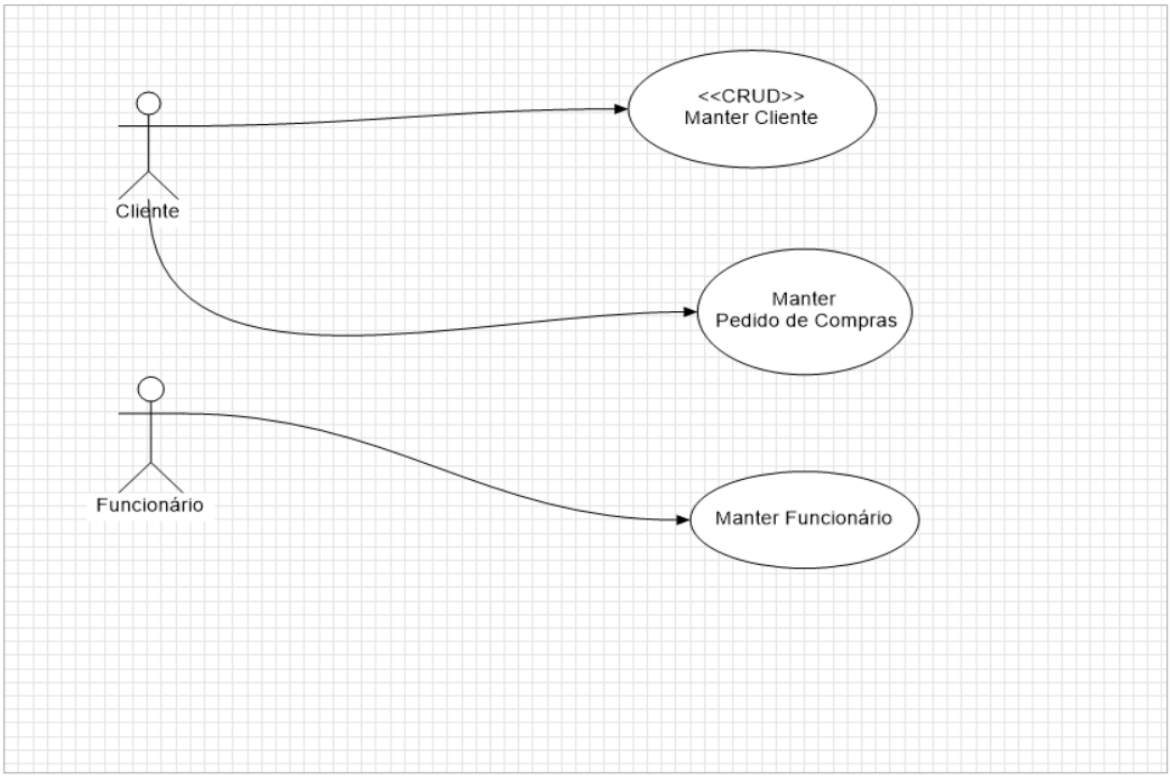

Figura 4 - Diagrama de casos de uso

A Figura 5 Apresenta uma visão geral do sistema, processo inicial da compra até a finalização com a entrega da compra.

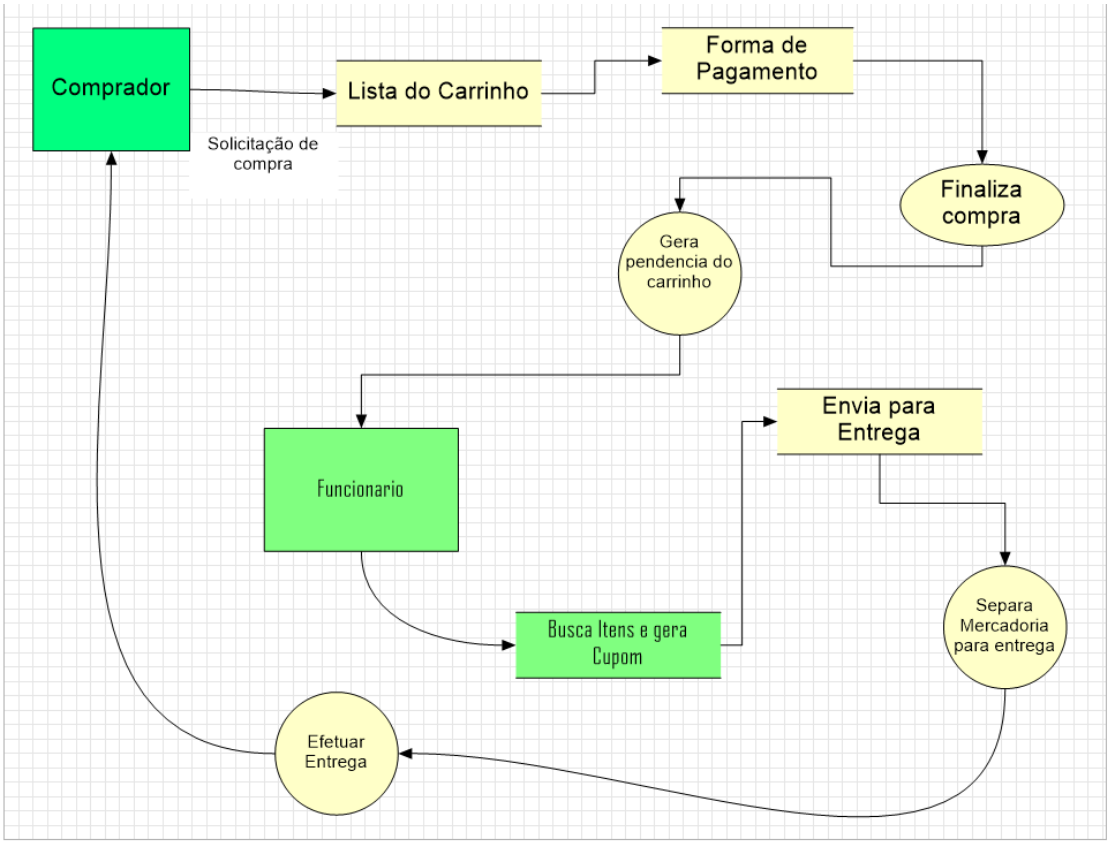

Figura 5 – Visão Geral do Software

A Figura 6 apresenta o diagrama de entidades do banco de dados.

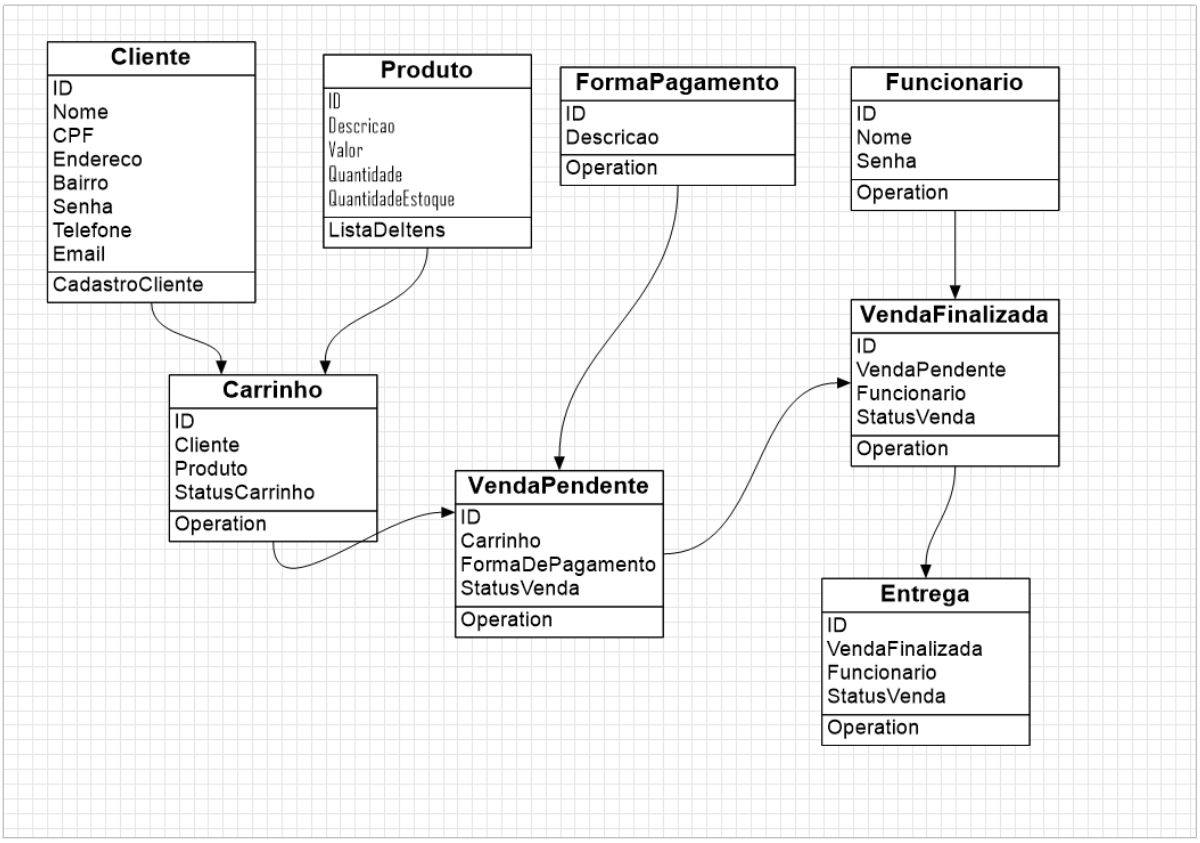

Figura 6 - Diagrama de entidades e relacionamentos do banco de dados

4.3 APRESENTAÇÃO DO SISTEMA

A seguir é apresentado na seção 4.3.1 o sistema de aplicativos do Cliente, onde é realizado o processo de solicitação de compras, e na seção 4.3.2 o aplicativo do funcionário, responsável por realizar a efetivação da compra e entrega da mercadoria.

4.3.1 Aplicativo Cliente

Na Figura 7 é apresentada a tela de *login* do cliente. Nessa tela é validado o CPF e senha do cliente, sendo possível também o acesso para o cadastro do uso do aplicativo.

Figura 7 - Tela de login

Na Figura 8 é possível visualizar a tela de cadastro do cliente contendo as informações pertinentes ao cliente e os dados de entrega das mercadorias adquiridas. Onde os campos de endereço, telefone e e-mail são obrigatórios.

Figura 8 - Tela de Cadastro de Cliente

A Figura 9 apresenta as opções do cliente que são Efetuar pedido de Compra e Status de compra. Na opção Efetuar Compra é possível visualizar a lista de itens disponíveis para a compra e as quantidades desejadas.

Figura 9 - Tela de Opções do Cliente

A Figura 10 apresenta os itens a serem escolhidos, sendo possível visualizar o nome dos itens e o valor unitário, e salvar o pedido de compra.

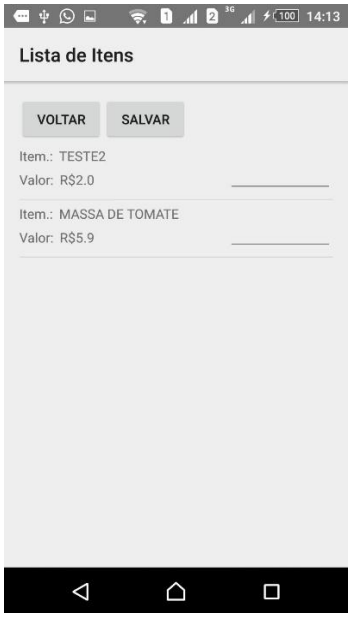

Figura 10 - Tela de Itens

A Figura 11 apresenta a tela de opções de forma de pagamento disponível para o pedido de compras.

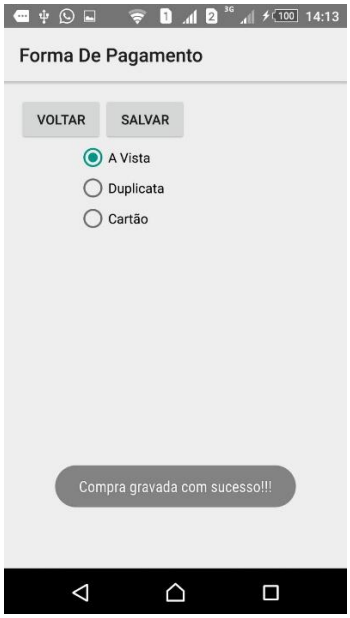

Figura 11 - Tela de Opções de Pagamento

Após essas telas o processo de pedido de compra é finalizado, disponibilizando para os funcionários uma pendência, para a efetivação da compra. A Figura 12 apresenta a lista de pedidos de compra efetuada pelo cliente.

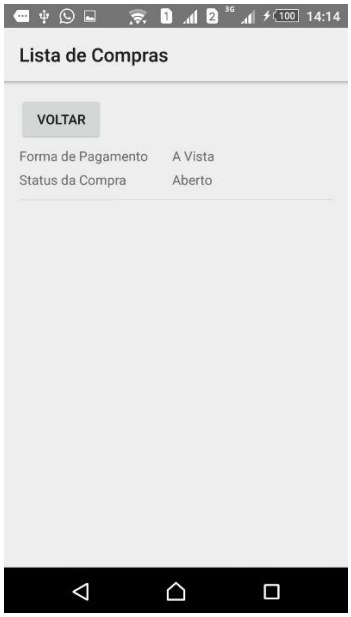

Figura 12 - Tela de Status do Pedido

4.3.2 Aplicativo Funcionário

Na Figura 13 é apresentada a tela de *login* do aplicativo do funcionário, usando a mesma lógica do Cliente CPF com a senha.

Figura 13 - Tela de Login do Funcionário

A Figura 14 apresenta as opções disponibilizadas para o funcionário, efetivar o pedido da compra e realizar a entrega da compra.

Figura 14 - Tela de Opções do Funcionário

Ao acessar a opção Efetivar Compras é apresentada uma lista com os pedidos pendentes, como apresenta a Figura 15.

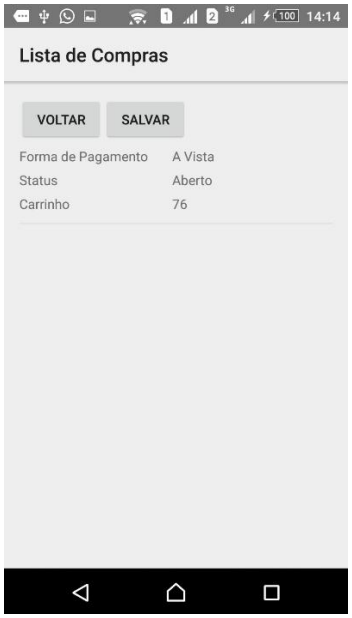

Figura 15 - Tela de Pedidos Pendentes

Ao clicar em um dos itens disponíveis na lista são apresentados os itens da compra efetuada pelo cliente, como é apresentado na Figura 16.

Figura 16 - Tela de Lista de Compras

Ao salvar, o sistema entende que a compra foi efetuada e, assim, será realizada uma pendência para a entrega das compras. A Figura 17 apresenta a tela que lista todas as pendências de entrega que estão em estado de abertas.

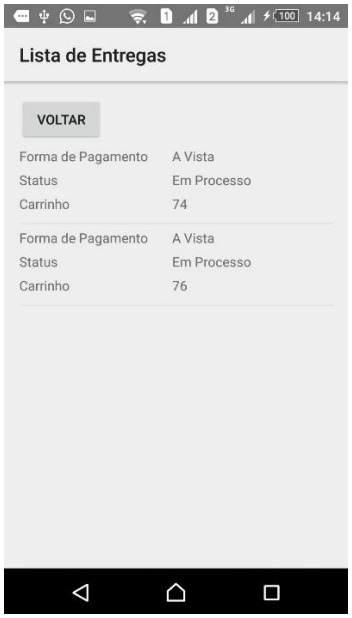

Figura 17 - Tela de Lista de Entregas

Ao clicar em cima do item da lista disponível é apresentado para o funcionário os dados de endereço e contato do cliente, para que seja possível realizar a entrega, como apresenta a Figura 18.

Figura 18 - Tela de Dados para a Entrega

Com esta tela, é encerrado o processo de compra iniciado pelo aplicativo Cliente.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção são apresentados os principais códigos utilizados para realizar a comunicação entre o servidor e os aplicativos Cliente e Funcionário.

Para exemplificar será apresentado o processo de *Login* do cliente. Esse processo inicia na tela do aplicativo do Cliente, enviando as informações para o Sevidor, que realiza a consulta no banco de dados verificando CPF e Senha. A Listagem 1 apresenta os códigos gerados para a tela em linguagem XML.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".PrincipalActivity">
    <EditText
        android:id="@+id/etLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/textViewCpf"
        android:layout_marginTop="14dp"
        android:ems="10"
        android:inputType="number"
        android:focusable="true" />

    <EditText
        android:id="@+id/etSenhaLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/etLogin"
        android:layout_alignRight="@+id/etLogin"
        android:layout_below="@+id/textViewSenha"
        android:layout_marginTop="16dp"
        android:ems="9"
        android:inputType="textPassword"
        android:focusable="true">

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textViewSenha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="Senha"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_below="@+id/etLogin"
        android:layout_alignParentStart="true" />

    <TextView
        android:id="@+id/textViewCpf"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CPF Cliente"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />

    <Button
        android:id="@+id/btLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        android:onClick="btLogin"
        android:layout_centerVertical="true"
        android:layout_toEndOf="@+id/textViewSenha" />

    <Button
        android:id="@+id/btCriarConta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Criar Conta"
        android:onClick="btCriarConta"
        android:layout_centerVertical="true"
        android:layout_alignEnd="@+id/etSenhaLogin"
        android:layout_marginEnd="60dp" />


</RelativeLayout>
```

Listagem 1 - Tela de Login

Ao clicar no botão de Login é disparado o evento OnClick, que está codificado na classe PrincipalActivityView. Esse evento busca as informações de CPF e senha informados e envia para o Servidor, como apresenta a Listagem 2.

```
package com.example.jeanfrancisco.teste.view;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.example.jeanfrancisco.teste.R;
import com.example.jeanfrancisco.teste.entidy.Cliente;
import com.google.gson.Gson;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class PrincipalActivityView extends Activity {

    public EditText etLogin;
    public EditText etSenhaLogin;
    public Button btLogin;
    public String leitura;
    public Cliente clienteretorno;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if
it is present.
        getMenuInflater().inflate(R.menu.menu_principal, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();

        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void btLogin(View v) throws InterruptedException {

        etLogin = (EditText) findViewById(R.id.etLogin);
        etSenhaLogin = (EditText) findViewById(R.id.etSenhaLogin);
```

```
        Cliente cliente = new Cliente();
        cliente.setCpf(etLogin.getText().toString());
        cliente.setSenha(etSenhaLogin.getText().toString());

        etLogin = (EditText) findViewById(R.id.etLogin);
        etSenhaLogin = (EditText) findViewById(R.id.etSenhaLogin);
        networkthread ob = new networkthread(cliente);
   }

    public void iniciar(Cliente login) {
        Toast.makeText(getApplicationContext(), leitura,
Toast.LENGTH_SHORT).show();

        if (login == null){

        } else {
            Intent intent = new Intent();

            intent.setClass(this, OpcoesView.class);

            intent.putExtra("cliente", clienteretorno);

            startActivity(intent);

            finish();
         }
    }

    public void btCriarConta(View v) {

        Intent intent = new Intent();

        intent.setClass(this, ClienteInserirView.class);

        // Inicia a nova tela.
        startActivity(intent);

        finish();
    }

    class networkthread implements Runnable {
        private static final String TAG = "Test";
        Cliente cliente = new Cliente();

        public networkthread(Cliente cliente) {
            this.cliente = cliente;
            Thread t = new Thread(this);
            t.start();

            long delayMillis = 5000;
            // 5 segundos
            try { t.join(delayMillis);
                if (t.isAlive()) {
```

```
                    // Tempo excedido; Thread ainda não terminou
seu processamento
                } else {
                   iniciar(clienteretorno);
                }
            } catch (InterruptedException e) {
                // Thread foi interrompida por alguma excessão
lançada

            }
        }

        @Override
        public void run() {
            Log.v(TAG, "---------> INICIO");
            try {
                URL urlObj = new
URL("http://192.168.14.99:8080/Teste/ClienteS");
                HttpURLConnection httpConn = (HttpURLConnection)
urlObj.openConnection();
                httpConn.setDoInput(true);
                httpConn.setDoOutput(true);
                httpConn.setUseCaches(false);
                httpConn.setRequestMethod("POST");
                httpConn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
                httpConn.setRequestProperty("Content-Language",
"pt-BR");
                httpConn.setRequestProperty("Accept",
"application/octet-stream");
                httpConn.setRequestProperty("Connection",
"close");
                DataOutputStream os = new
DataOutputStream(httpConn.getOutputStream());

                cliente.setMetodo("FAZERLOGIN");
                Gson gson = new Gson();
                String json = gson.toJson(cliente);

                os.writeUTF(json);
                os.close();
                String msg = httpConn.getResponseMessage();
                int code = httpConn.getResponseCode();
                if (code == 200) {
                    DataInputStream dis = new
DataInputStream(httpConn.getInputStream());

                    String retorno = dis.readUTF();
                    Gson returngson = new Gson();
                   clienteretorno = returngson.fromJson(retorno,
Cliente.class);

                    if (clienteretorno == null) {
                        leitura = "Erro no login";
                    } else{
```

```
                        leitura = "Usuário Logado";
                    }
                    dis.close();
                } else {
                    Log.i(PrincipalActivityView.class.getName(),
String.format("Código invalido:: %s", msg));
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }


    public void mostrarErro(String erro) {
            AlertDialog.Builder builder = new
AlertDialog.Builder(PrincipalActivityView.this);
            builder.setMessage("Erro")
                    .setPositiveButton("Ok", new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface
dialog, int which) {
                            etLogin.requestFocus();

                        }
                    })
                    .setNegativeButton(null, null);

            builder.create();
            builder.show();

        }

        public void abrirTela() {

            Intent home = new Intent(PrincipalActivityView.this,
FormaDePagamentoView.class);
            startActivity(home);

        }

}
```

Listagem 2 – Classe PrincipalActivityView

Após o envio para o Servidor, a classe ClienteS recebe as informações e envia para o banco de dados, validando as informações e retornando o objeto, conforme apresenta a Listagem 3.

```
package br.com.apps.supermercadomobile.servlets;
```

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import br.com.apps.supermercadomobile.entity.Carrinho;
import br.com.apps.supermercadomobile.utils.ProcuraMetodos;

import com.google.gson.Gson;

@WebServlet(name = "CarrinhoS", urlPatterns = {"/CarrinhoS"})
public class CarrinhoS extends HttpServlet{

     private static final long serialVersionUID = -
8995586145103054733L;

         @Override
         protected void service(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {


           System.out.println("INICIOU CONEXÃO");
              resp.setContentType("text/plain;charset=UTF-8");
              DataInputStream din = new
DataInputStream(req.getInputStream());
              //Vazio não passa
              String json =   din.readUTF();

              //Procura o método vindo do mobile
              Carrinho carrinho = new Carrinho();
                 try {
                       carrinho = procuraMetodo(json);
                 } catch (Exception e) {
                       // TODO Auto-generated catch block
                       e.printStackTrace();
                 }
                 //Retorna
              Gson gson = new Gson();
              String returnjson = gson.toJson(carrinho);
              OutputStream out = resp.getOutputStream();
              DataOutputStream dos = new DataOutputStream(out);
              dos.writeUTF(returnjson);
              dos.writeUTF("\ntexto");
              dos.close();
         }

           public Carrinho procuraMetodo(String json) throws
Exception{
```

```
            Gson gson = new Gson();
            Carrinho carrinho = gson.fromJson(json,
Carrinho.class);

               ProcuraMetodos pm = new ProcuraMetodos();
               return pm.procurarMetodoCarrinho(carrinho);
               }

        @Override
        protected void doGet(HttpServletRequest request,
HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }


        @Override
        protected void doPost(HttpServletRequest request,
HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }

        @Override
        public String getServletInfo() {
            return "Short description";
        }

        private void processRequest(HttpServletRequest request,
HttpServletResponse response) {
            throw new UnsupportedOperationException("Not yet
implemented");
        }
}
```

Listagem 3 – Classe ClienteS

A classe ClienteS recebe o objeto e envia para a classe ProcuraMetodos, responsável por identificar qual é o método que o objeto vai executar dentro da classe DAO do objeto, que faz a comunicação com o banco de dados. A Listagem 4 apresenta a Classe ProcuraMetodos.

```
package br.com.apps.supermercadomobile.utils;


import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;


import br.com.apps.supermercadomobile.control.CarrinhoDAO;
```

```
import br.com.apps.supermercadomobile.control.ClienteDAO;
import br.com.apps.supermercadomobile.control.EntregaDAO;
import br.com.apps.supermercadomobile.control.FormaPagamentoDAO;
import br.com.apps.supermercadomobile.control.FuncionarioDAO;
import br.com.apps.supermercadomobile.control.ProdutoDAO;
import br.com.apps.supermercadomobile.control.VendaFinalizadaDAO;
import br.com.apps.supermercadomobile.control.VendaPendenteDAO;
import br.com.apps.supermercadomobile.entity.Carrinho;
import br.com.apps.supermercadomobile.entity.Cliente;
import br.com.apps.supermercadomobile.entity.Entrega;
import br.com.apps.supermercadomobile.entity.FormaPagamento;
import br.com.apps.supermercadomobile.entity.Funcionario;
import br.com.apps.supermercadomobile.entity.Produto;
import br.com.apps.supermercadomobile.entity.VendaFinalizada;
import br.com.apps.supermercadomobile.entity.VendaPendente;


public class ProcuraMetodos {

	public final String FAZERLOGIN = "FAZERLOGIN";
	public final String INSERIR = "INSERIR";
	public final String PROCURAR = "PROCURAR";
	public final String PROCURARTODOS = "PROCURARTODOS";
	public final String PROCURARTODOSA = "PROCURARTODOSA";
	public final String UPDATE = "UPDATE";
	public final String PROCURARCARRINHO = "PROCURARCARRINHO";
	public final String UPDATEF = "UPDATEF";



	public Cliente procurarMetodoCliente(Cliente objeto) throws
Exception {

		if (objeto.getMetodo().equalsIgnoreCase(FAZERLOGIN)) {
			ClienteDAO dao = new ClienteDAO();
			return dao.fazerLogin(objeto);
		} else if (objeto.getMetodo().equalsIgnoreCase(INSERIR))
```

```
{
                ClienteDAO dao = new ClienteDAO();
                return dao.insert(objeto);
          }else if (objeto.getMetodo().equalsIgnoreCase(PROCURAR))
{
                ClienteDAO dao = new ClienteDAO();
                return dao.findByIds(objeto);
          }if
(objeto.getMetodo().equalsIgnoreCase(PROCURARCARRINHO)){
                ClienteDAO dao = new ClienteDAO();
                return dao.findByCarrinho(objeto);
          }
          return null;

     }


     public   Carrinho   procurarMetodoCarrinho(Carrinho   objeto)
throws Exception {

          if (objeto.getMetodo().equalsIgnoreCase(INSERIR)) {
                CarrinhoDAO dao = new CarrinhoDAO();
                return dao.insert(objeto);
          } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
                CarrinhoDAO dao = new CarrinhoDAO();
                return dao.update(objeto);
          }if (objeto.getMetodo().equalsIgnoreCase(PROCURAR)) {
                CarrinhoDAO dao = new CarrinhoDAO();
                return dao.findByIds(objeto.getId());
          }if (objeto.getMetodo().equalsIgnoreCase(UPDATE)) {
                CarrinhoDAO dao = new CarrinhoDAO();
                return dao.update(objeto);
          }
          return null;
     }

     public  Entrega  procurarMetodoEntrega(Entrega  objeto)  throws
```

```
Exception {

            if (objeto.getMetodo().equalsIgnoreCase(INSERIR)) {
                  EntregaDAO dao = new EntregaDAO();
                  return dao.insert(objeto);
            } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
                  EntregaDAO dao = new EntregaDAO();
                  return dao.update(objeto);
            }
            return null;

      }
      public                                          FormaPagamento
procurarMetodoFormaPagamento(FormaPagamento      objeto)      throws
Exception {

            if (objeto.getMetodo().equalsIgnoreCase(INSERIR)) {
                  FormaPagamentoDAO dao = new FormaPagamentoDAO();
                  return dao.insert(objeto);
            } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
                  FormaPagamentoDAO dao = new FormaPagamentoDAO();
                  return dao.update(objeto);
            }
            return null;

      }
      public    Funcionario    procurarMetodoFuncionario(Funcionario
objeto) throws Exception {

            if (objeto.getMetodo().equalsIgnoreCase(FAZERLOGIN)) {
                  FuncionarioDAO dao = new FuncionarioDAO();
                  return dao.fazerLogin(objeto);
            } else if (objeto.getMetodo().equalsIgnoreCase(INSERIR))
{
                  FuncionarioDAO dao = new FuncionarioDAO();
                  return dao.insert(objeto);
            } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
```

```
                FuncionarioDAO dao = new FuncionarioDAO();
                return dao.update(objeto);
          }

          return null;

     }

     public Produto procurarMetodoProduto(Produto objeto) throws
Exception {

          if (objeto.getMetodo().equalsIgnoreCase(INSERIR)) {
                ProdutoDAO dao = new ProdutoDAO();
                return dao.insert(objeto);
          } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
                ProdutoDAO dao = new ProdutoDAO();
                return dao.update(objeto);
          }
          return null;

     }
     public                                        VendaFinalizada
procurarMetodoVendaFinalizada(VendaFinalizada     objeto)     throws
Exception {

          if (objeto.getMetodo().equalsIgnoreCase(INSERIR)) {
                VendaFinalizadaDAO dao = new VendaFinalizadaDAO();
                return dao.insert(objeto);
          } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
                VendaFinalizadaDAO dao = new VendaFinalizadaDAO();
                return dao.update(objeto);
          }
          return null;

     }
     public                                          VendaPendente
procurarMetodoVendaPendente(VendaPendente objeto) throws Exception
```

```
{

            if (objeto.getMetodo().equalsIgnoreCase(INSERIR)) {
                  VendaPendenteDAO dao = new VendaPendenteDAO();
                  return dao.insert(objeto);
            } else if (objeto.getMetodo().equalsIgnoreCase(UPDATE)){
                  VendaPendenteDAO dao = new VendaPendenteDAO();
                  return dao.update(objeto);
            }                           else                           if
(objeto.getMetodo().equalsIgnoreCase(UPDATEF)){
                  VendaPendenteDAO dao = new VendaPendenteDAO();
                  return dao.updatef(objeto);
            }
            return null;

      }

public   List<VendaPendente>   procurarMetodoVendaPendentes(Cliente
objeto) throws Exception {

             if (objeto.getMetodo().equalsIgnoreCase(PROCURAR)){
                  VendaPendenteDAO dao = new VendaPendenteDAO();
                  return dao.findByCliente(objeto);
            }else                                                    if
(objeto.getMetodo().equalsIgnoreCase(PROCURARTODOS)) {
                  VendaPendenteDAO dao = new VendaPendenteDAO();
                  return dao.findByAll(objeto);
            }else                                                    if
(objeto.getMetodo().equalsIgnoreCase(PROCURARTODOSA)) {
                  VendaPendenteDAO dao = new VendaPendenteDAO();
                  return dao.findByAllA(objeto);
            }
            return null;

      }
```

```
     public List<Produto>   procurarMetodoProdutos(Produto objeto)
throws SQLException {
           if (objeto.getMetodo().equalsIgnoreCase(PROCURAR)) {
                 ProdutoDAO dao = new ProdutoDAO();
                 return dao.lisdaDeProdutos();
           }else                                                  if
(objeto.getMetodo().equalsIgnoreCase(PROCURARTODOS)) {
                 ProdutoDAO dao = new ProdutoDAO();
                 return dao.lisdaDeProdutos();
           }
           return null;

     }
}
```

Listagem 4 – Classe ProcuraMetodos

Conforme apresenta a Listagem 5, a classe ClienteDao é invocada, para realizar o login do cliente, retornando o objeto Cliente caso encontrado. A listagem 6 apresenta a classe Cliente Dao.

```
package br.com.apps.supermercadomobile.control;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import conexao.ConexaoMySQL;
import br.com.apps.supermercadomobile.entity.Cliente;
import br.com.apps.supermercadomobile.interfaces.IAbstractDAO;
import br.com.apps.supermercadomobile.interfaces.IClienteDAO;

public class ClienteDAO implements IClienteDAO,
IAbstractDAO<Cliente>{
```

```
     private final String SELECT_NOME    = "SELECT id, nome, cpf,
endereco, bairro, telefone, email FROM cliente WHERE nome= ?";

     private final String SELECT_ID    = "SELECT id, nome, cpf,
endereco, bairro, telefone, email FROM cliente WHERE id= ?";

     private final String INSERT_CLIENTE = "INSERT INTO cliente
(nome, senha, cpf, endereco, bairro, telefone, email) VALUES
(?,?,?,?,?,?,?)";

     private final String SELECT_LOGIN   = "SELECT id, nome, cpf,
endereco, bairro, telefone, email FROM cliente WHERE cliente.cpf =
? and cliente.senha = ?";

     private final String UPDATE_ID      = "UPDATE cliente SET
nome=?, senha=?, cpf=?, endereco=?, bairro=?, telefone=? email=?
WHERE cliente.cpf=? ";

     private final String SELECT_CARRINHO    = "SELECT c.id,
c.nome, c.cpf, c.endereco, c.bairro, c.telefone, c.email FROM
cliente as c inner join  carrinho as car on (car.cliente = c.id)
WHERE car.id= ?";

     @Override
     public Cliente fazerLogin(Cliente entity) throws Exception {
          Connection conn = new ConexaoMySQL().getConnection();
          PreparedStatement pstmt =
conn.prepareStatement(SELECT_LOGIN);
          pstmt.setString(1, entity.getCpf());
          pstmt.setString(2, entity.getSenha());
          ResultSet res = pstmt.executeQuery();
          if (res.next()){
               entity.setId(res.getInt("id"));
               entity.setNome(res.getString("nome"));
               entity.setEndereco(res.getString("endereco"));
```

```
                entity.setBairro(res.getString("bairro"));
                entity.setTelefone(res.getString("telefone"));
                entity.setEmail(res.getString("email"));
                res.close();
                pstmt.close();
                conn.close();
                return entity;
          }
          res.close();
          pstmt.close();
          conn.close();
          return null;
     }

     @Override
     public Cliente insert(Cliente entity) throws Exception {
          Connection conn = new ConexaoMySQL().getConnection();
          PreparedStatement pstmt =
conn.prepareStatement(INSERT_CLIENTE);
          pstmt.setString(1, entity.getNome());
          pstmt.setString(2, entity.getSenha());
          pstmt.setString(3, entity.getCpf());
          pstmt.setString(4, entity.getEndereco());
          pstmt.setString(5, entity.getBairro());
          pstmt.setString(6, entity.getTelefone());
          pstmt.setString(7, entity.getEmail());
          Integer i = 0;
          i = pstmt.executeUpdate();
          if (i == 0) {
                pstmt.close();
                conn.close();
                return null;
          } else {
                pstmt.close();
                conn.close();
                return entity;
          }
```

```
	}

	@Override
	public Cliente update(Cliente entity)throws Exception {
		Connection conn = new ConexaoMySQL().getConnection();
		PreparedStatement pstmt =
conn.prepareStatement(UPDATE_ID);
		pstmt.setString(1, entity.getNome());
		pstmt.setString(2, entity.getSenha());
		pstmt.setString(3, entity.getCpf());
		pstmt.setString(4, entity.getEndereco());
		pstmt.setString(5, entity.getBairro());
		pstmt.setString(6, entity.getTelefone());
		pstmt.setString(7, entity.getEmail());
		Integer i = 0;
		i = pstmt.executeUpdate();
		if (i == 0) {
			pstmt.close();
			conn.close();
			return null;
		} else {
			pstmt.close();
			conn.close();
			return entity;
		}
	}

	@Override
	public List<Cliente> findByNome(Cliente entidy) throws
Exception {
		Connection conn = new ConexaoMySQL().getConnection();
		PreparedStatement pstmt =
conn.prepareStatement(SELECT_NOME);
		pstmt.setString(1, entidy.getNome());
		ResultSet res = pstmt.executeQuery();
		List<Cliente> clientes = new ArrayList<Cliente>();
		while (res.next()){
```

```
             Cliente cliente = new Cliente();
             cliente.setId(res.getInt("id"));
             cliente.setNome(res.getString("nome"));
             cliente.setCpf(res.getString("cpf"));
             cliente.setEndereco(res.getString("endereco"));
             cliente.setBairro(res.getString("bairro"));
             cliente.setTelefone(res.getString("telefone"));
             cliente.setEmail(res.getString("email"));
             clientes.add(cliente);
            }
            pstmt.close();
            conn.close();
            return clientes;

      }



      public Cliente findByIds(Cliente cliente) throws SQLException
{
            Connection conn = new ConexaoMySQL().getConnection();
            PreparedStatement pstmt =
conn.prepareStatement(SELECT_ID);
            pstmt.setInt(1, cliente.getId());
            ResultSet res = pstmt.executeQuery();
            Cliente entity = new Cliente();
            if (res.next()){
                  entity.setId(res.getInt("id"));
                  entity.setNome(res.getString("nome"));
                  entity.setCpf(res.getString("cpf"));
                  entity.setBairro(res.getString("bairro"));
                  entity.setEndereco(res.getString("endereco"));
                  entity.setTelefone(res.getString("telefone"));;
                  entity.setEmail(res.getString("email"));
                  pstmt.close();
                  conn.close();
                  return entity;
```

```
            }
            pstmt.close();
            conn.close();
            return null;
      }


      public Cliente findByCarrinho(Cliente cliente) throws
SQLException {
            Connection conn = new ConexaoMySQL().getConnection();
            PreparedStatement pstmt =
conn.prepareStatement(SELECT_CARRINHO);
            pstmt.setInt(1, cliente.getId());
            ResultSet res = pstmt.executeQuery();
            Cliente entity = new Cliente();
            if (res.next()){
                  entity.setId(res.getInt("id"));
                  entity.setNome(res.getString("nome"));
                  entity.setCpf(res.getString("cpf"));
                  entity.setBairro(res.getString("bairro"));
                  entity.setEndereco(res.getString("endereco"));
                  entity.setTelefone(res.getString("telefone"));;
                  entity.setEmail(res.getString("email"));
                  pstmt.close();
                  conn.close();
                  return entity;
            }
            pstmt.close();
            conn.close();
            return null;
      }


      @Override
      public List<Cliente> list() {
            // TODO Auto-generated method stub
            return null;
```

```
	}

	@Override
	public void delete(Cliente entity) {
		// TODO Auto-generated method stub

	}

	@Override
	public Cliente findById(Integer id) {
		// TODO Auto-generated method stub
		return null;
	}


}
```

Listagem 5 – Classe ClienteDao

5 CONCLUSÃO

O objetivo deste trabalho foi a implementação dos aplicativos e do servidor, a fim de auxiliar e facilitar as compras realizadas em supermercados, agilizando os pedidos em casos de pequenas compras de forma simples e rápida.

Em relação ao Servidor, a dificuldade encontrada foi a de buscar as informações de produtos e dos funcionários para que fossem disponibilizadas nos aplicativos. Visando então a implementação em qualquer supermercado, foi então criado apenas a classe produtos e funcionário com as informações básicas encontradas. Assim, para um trabalho futuro, uma implementação com outro sistema de gerenciamento de supermercado que apenas inclua essas informações no banco de dados.

Em relação aos aplicativos, uma das dificuldades foi a utilização de listas que fossem possíveis a alteração dos dados dos objetos nela contidos. Foi utilizado o Adapters, sendo possível esta implementação e disponibilização dos dados.

REFERÊNCIAS

ABRIL, **Copyright © Editora Abril S.A** em <http://exame.abril.com.br/negocios/noticias/as-50-maiores-redes-de-supermercados-do-brasil> Acesso em 26 out. 2015.

BARBOSA, Daniela. **As 50 maiores redes de supermercados do Brasil**. Disponível em: < http://exame.abril.com.br/negocios/noticias/as-50-maiores-redes-de-supermercados-do-brasil>. Acesso em: 20 out. 2015.

DAVENPORT, THOMAS H.. **Missão crítica: obtendo vantagem competitiva com os sistemas de gestão empresarial**. Porto Alegre, RS: Bookman, 2002.293

DEVELOPERS. Disponível em <http://developer.android.com/intl/pt-br/tools/studio/index.html> Acesso em: 27 out. 2015

ECLIPSE. **Eclipse Link**. Disponível em <http://www.eclipse.org/eclipselink/>. Acesso em: 11 mar. 2012.

GAZETA DO POVO. **Todos os números da internet em 2012**. Disponível em: < http://www.gazetadopovo.com.br/tecnologia/todos-os-numeros-da-internet-em-2012-b7rzqwt8es7cohwhi5pvd2eco>. Acesso em: 20 out. 2015.

GITHUB. **GitHub Inc**. Disponível em <https://github.com/google/gson/>. Acesso em 27 out. 2015.

HABERKORN, Ernesto. **Teoria do ERP - Enterprise Resource Planning**. São Paulo: Makron Books do Brasil, 1999.

LARMAN Craig, Utilizando UML e Padrões, Bookman, 2007

MENDES, Douglas Rocha. **Programação Java - com ênfase em orientação a objetos.** São Paulo Novatec Editora LTDA, 2009.

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

SCOTT, Kendall. The Unified Process Explained. Addison-Wesley, 2002

SEBRAE MINAS, Douglas Rocha. Comércio Saiba, como montar Supermercado, Mercearia e Similares. Sebrae Ponto de partida, 2013.

WIKI. Disponível em <http://wiki.cm.utfpr.edu.br/index.php/GlassFish> Acesso em 27 out. 2015