INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, *CAMPUS* PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

FERNANDA CAMPEOL

MOVE:
APLICATIVO MÓVEL DE APOIO ÀS VENDAS

Orientador: Prof. Me. Fernando Abrahão Afonso
Coorientador: Prof. Me. Carlos Alberto Petry

PASSO FUNDO, 2014

FERNANDA CAMPEOL

MOVE:
APLICATIVO MÓVEL DE APOIO ÀS VENDAS

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Fernando Abrahão Afonso
Coorientador: Carlos Alberto Petry

PASSO FUNDO, 2014

FERNANDA CAMPEOL

MOVE:
APLICATIVO MÓVEL DE APOIO ÀS VENDAS

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Orientador Me. Fernando Abrahão Afonso

_____

Coorientador Me. Carlos Alberto Petry

_____

Prof. Wilian Bouviér

_____

Prof. Me. Rafael Marisco Bertei

_____

Prof. Dr. Alexandre Tagliari Lazzaretti

Coordenador do Curso

PASSO FUNDO, 2014

Aos meus pais,
por tudo.
A Fernando Rambo, que me
apresentou a este desafiante
mundo da programação.

AGRADECIMENTOS

Agradeço a Deus.

Agradeço a todos que de alguma forma contribuíram para a realização deste trabalho. Em especial ao professor Fernando Abrahão Afonso que acolheu minha pesquisa como orientador e à professora Anúbis Graciela de Moraes Rossetto que me orientou nos primeiros passos do projeto do trabalho de conclusão de curso.

Agradeço aos meus pais, Fernando e Cleonice, pela dedicação, pelo apoio e pelos ensinamentos – a quem devo muito do que sou.

Agradeço ao Fernando Rambo pelas sugestões, troca de ideias e estímulo quando titubeava diante das dificuldades.

Agradeço a todos os professores do curso por seus ensinamentos e ao IFSUL pela qualidade de ensino oferecida para minha formação e de cada colega.

“Seus clientes menos satisfeitos são
sua maior fonte de aprendizado.”
Bill Gates

“As pessoas não sabem o que querem
até mostrarmos a elas.”
Steve Jobs

RESUMO

Este trabalho apresenta o estudo sobre a plataforma Android, as tecnologias XML e Web Service e o resultado da utilização das mesmas no desenvolvimento de um aplicativo para dispositivos móveis. O aplicativo desenvolvido com base nas tecnologias estudadas consiste em uma ferramenta que auxilia os vendedores de lojas de móveis, eletroeletrônicos e materiais de construção no atendimento aos clientes, uma vez que facilita a consulta ao preço dos produtos expostos no mostruário, permitindo ainda conferir o estoque sem que para isso o vendedor necessite deslocar-se até um terminal dentro da loja para fazer a verificação. Com o uso das tecnologias estudadas, desenvolveu-se um aplicativo que visa facilitar a tarefa dos vendedores, reduzindo os deslocamentos dentro da loja, ampliando a sua agilidade e eficiência no atendimento, bem como a satisfação do cliente e o lucro da empresa.

Palavras-chave: *Android*; consulta aos preços; agilidade; eficiência; *Web Service*.

ABSTRACT

This paper presents a study on the Android platform, XML and Web Service technologies and the result of the use of the same in developing an application for mobile devices. The application developed based on the studied technologies is a tool that helps sellers of furniture, electronics and construction materials in customer service shops, as it facilitates the query to the price of products displayed in showcase, allowing even check the stock without this the seller needs to move a terminal inside the store to do the checking. With the use of the studied technologies, has developed an application that aims to facilitate the task of the sellers, reducing the displacements inside the store, increasing its agility and efficient service as well as customer satisfaction and company profit.

Keywords: Android; Consultation prices; agility; efficiency; Web Service.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADT – *Android Development Tools*, p. 23
API – *Application Programming Interface*, p. 19
CPF – Cadastro de Pessoa Física, p. 28
GPS – *Global Positioning System*, p. 15
ID – Identificador, p. 16
IDE – *Integrated Development Environment*, p. 13
MER – Modelo Entidade-Relacionamento, p. 26
SDK – *Software Development Kit*, p. 13
SOA - *Service Oriented Architecture*, p. 21
SOAP – *Simple Object Access Protocol*, p. 21
TI – Tecnologia da Informação, p. 11
URI – *Uniform Resource Identifier*, p. 18
XML – *Extensible Markup Language*, p. 17

SUMÁRIO

1 INTRODUÇÃO

A Tecnologia da Informação (TI) nunca esteve tão presente no cotidiano da sociedade pós-Moderna. Hoje, tarefas executadas com o auxílio de softwares parecem simples, porém demandavam muito tempo e árduo trabalho em um período não tão remoto. As empresas dos diversos setores da economia, por sua vez, têm se dado conta de que a TI pode lhes proporcionar maior eficiência e lucratividade, apresentando-se como um diferencial aos seus colaboradores e clientes. Analisando-se essa situação, fica claro que todo programa desenvolvido para auxiliar ou satisfazer alguma necessidade da empresa e que possa trazer alguma forma de ganho, quer seja de tempo, quer seja financeiro, será bem aceita. Portanto, é papel do profissional de TI identificar essas necessidades e tentar solucioná-las.

Neste contexto, detectou-se uma necessidade do comércio ao se observar *in loco* que a grande maioria das lojas de móveis, eletroeletrônicos e materiais de construção atende aos clientes executando o mesmo *modus operandi,* ou seja, os vendedores trabalham em um grande espaço físico, percorrendo os diferentes setores/departamentos, e, quando solicitados, procuram um ponto – que aqui se definiu como "ponto de acesso fixo" - onde haja um computador em que possam consultar o preço de determinado produto através de seu código. Muitas vezes o cliente necessita de produtos de setores diferentes dentro da loja, o que demanda grande perda de tempo do vendedor para se deslocar até algum ponto de acesso fixo toda vez que o cliente deseja informação sobre determinado produto. Consequentemente, ocorre redução de eficiência, uma vez que aumenta o tempo de atendimento por cliente; logo, diminui a quantidade de clientes atendidos, uma possível comissão do vendedor e o lucro do estabelecimento.

Nesse sentido, conclui-se que o tempo gasto com deslocamentos dentro da loja implica negativamente na satisfação do cliente e na eficiência do vendedor. Logo se propôs uma solução: um aplicativo móvel de consultas aos preços das mercadorias expostas na loja, cuja razão de ser está em instrumentalizar a tarefa do vendedor tornando-o mais ágil e efetivo em seu trabalho.

Sendo assim, neste primeiro capítulo é apresentada na introdução, uma contextualização da pesquisa, a motivação para a mesma e os objetivos geral e específicos. No capítulo seguinte, contextualiza-se o uso da tecnologia da informação pelo comércio e são descritas as tecnologias que serviram de base para o desenvolvimento da aplicação, a saber:

Plataforma Android e Web Service. No capítulo três, é descrito o projeto e o desenvolvimento do aplicativo Move: Aplicativo Móvel de Apoio das Vendas, cuja descrição envolve a apresentação dos requisitos, a modelagem e o protocolo de integração. Ainda nesse capítulo, as telas construídas para o aplicativo podem ser visualizadas e compreendidas a partir de textos que descrevem suas funcionalidades. Já nas Considerações Finais, é feita uma análise geral do processo que deu origem ao aplicativo, quais foram as dificuldades encontradas e possíveis aperfeiçoamentos futuros.

1.1 MOTIVAÇÃO

As lojas de móveis, eletroeletrônicos e materiais de construção geralmente se situam em prédios com ampla área comercial, na qual as mercadorias encontram-se distribuídas. Aos vendedores, cabe apresentar o produto e/ou similares solicitado pelo cliente, informando-lhe o preço e as condições de pagamento e talvez, posteriormente, efetivar a venda. No entanto, a tarefa de informar o preço ao interessado é dificultada, pois as mercadorias, por sofrerem frequentes mudanças de valor, são identificadas na loja por um código. É através dele que o vendedor pode utilizar um dos computadores de consulta da loja para verificar informações sobre os produtos. Assim, a simples tarefa de informar o preço exige que o vendedor confira o código de identificação que está afixado na mercadoria e desloque-se até um ponto de acesso fixo dentro da loja a fim de fazer a consulta ao preço.

Motivado por este problema e pela possibilidade de resolvê-lo com o auxílio de tecnologias voltadas para a mobilidade, apresenta-se aqui uma solução: desenvolver um aplicativo para dispositivos móveis baseados na plataforma Android que substitua os pontos fixos para consultas de preço, diminuindo assim os constantes deslocamentos dos vendedores em lojas de móveis, eletroeletrônicos e materiais de construção.

Objetiva-se construir um aplicativo móvel que facilite a tarefa dos vendedores, reduzindo a necessidade de deslocamentos para consulta dos preços dentro da loja, ampliando a sua agilidade e eficiência, a satisfação do cliente e o lucro da empresa.

1.2 OBJETIVOS

Aplicar o estudo de diferentes tecnologias para o desenvolvimento de um aplicativo móvel em Android que agilize e torne mais eficiente a tarefa de consulta ao preço dos produtos expostos nos mostruários das lojas de móveis, eletroeletrônicos e materiais de construção.

1.2.1 Objetivo Geral

Desenvolver um aplicativo móvel a fim de facilitar a tarefa de consulta aos preços dos produtos expostos em lojas de móveis, eletroeletrônicos e materiais de construção, reduzindo a necessidade de deslocamentos dentro da loja e assim ampliando a agilidade e eficiência dos vendedores.

1.2.2 Objetivos Específicos

Para que se chegue ao objetivo proposto com êxito, se fez necessário:

Estudar as tecnologias Android SDK e *Web Service*;

Conhecer e utilizar a IDE Eclipse para desenvolvimento do aplicativo e do *Web Service*;

Projetar o *web service* e o aplicativo de consulta ao preço dos produtos para loja de móveis, eletroeletrônicos e/ou materiais de construção, que será instalado em um dispositivo tipo Smartphone com Android e que consultará o *web service* desenvolvido;

Implementar o *web service* e o aplicativo de consulta ao preço dos produtos para loja de móveis, eletroeletrônicos e/ou materiais de construção;

Desenvolver um protocolo de comunicação entre o aplicativo proposto e o *Web Service*.

2 MOVE: APLICATIVO MÓVEL DE APOIO À VENDA

Segundo Prates e Ospina *apud* Oliveira: “O propósito básico da informação, dentro do contexto organizacional (...) é o de habilitar a empresa a alcançar seus objetivos por meio do uso eficiente dos recursos disponíveis (pessoas, materiais, equipamentos, tecnologia, dinheiro, além da própria informação)” (2004). Desta forma, as empresas sentem a necessidade de automatizar seus processos desde a produção até o atendimento aos seus clientes. Esta busca crescente por ferramentas que auxiliem seus processos evidencia um novo olhar do gestor para diferenciar-se num mercado cada vez mais competitivo.

Assim, as empresas de softwares produziram diversas ferramentas para gestão dos processos empresariais. Neste contexto, se tem diferentes softwares, escritos em distintas linguagens de programação, rodando em plataformas diferentes servindo as empresas especialmente no processo de venda finalizado no balcão da loja, ou seja, quando o cliente efetiva sua compra.

A proposta aqui, a partir deste cenário, é oferecer uma ferramenta que automatize o processo anterior à efetivação da venda no balcão, isto é, que torne mais efetiva e ágil a função do vendedor de apresentar o produto e seu respectivo preço ao cliente, auxiliando-o na compra.

2.1 INSTRUMENTALIZAÇÃO DO COMÉRCIO PELA TECNOLOGIA DA INFORMAÇÃO

A Tecnologia da Informação tem sido adotada pelos diferentes ramos de atividades dos vários setores da economia. Logo, a TI figura hoje como uma necessidade tanto para as atividades primárias e secundárias, como para o terceiro setor, oferecendo principalmente recursos que auxiliam a gestão dos negócios e a obtenção da excelência na produção e nos serviços.

Em se tratando do setor terciário, especificamente das atividades comerciais, a manutenção da empresa no mercado está ligada basicamente a dois fatores principais: um interno, que diz respeito à gestão, tarefa que hoje, em muitas empresas, é otimizada pela Tecnologia da Informação e um externo, que diz respeito à satisfação dos clientes, algo que está ligado diretamente à qualidade do atendimento.

Aqui se apresenta um aplicativo móvel para auxiliar os vendedores de lojas de móveis, eletroeletrônicos e materiais de construção a desempenhar melhor a sua função de atendimento aos clientes. Nesse caso, o sucesso da empresa depende do sucesso nas vendas, porém o sucesso de uma venda não é medido exclusivamente pela sua efetuação ou não, mas principalmente pelo grau de satisfação do cliente. Esse grau de satisfação depende da cordialidade, agilidade e eficiência do vendedor, além do resultado provocado pelo bom atendimento, geralmente, a fidelização do cliente.

2.2 PLATAFORMA ANDROID

O Android é um sistema operacional desenvolvido pela Open Handset Alliance, uma aliança com mais de 33 empresas e liderada pela Google. É um projeto de código aberto baseado no kernel do Linux destinado a dispositivos móveis.

Existem importantes características do Android que tem contribuído para o seu sucesso enquanto plataforma *mobile*, tais como:

Suporte Java: no Android as aplicações são desenvolvidas em linguagem Java.

Suporte de Hardware: oferece suporte a múltiplas ferramentas de hardware, como: câmeras, ecrãs sensíveis ao toque, GPS[1], acelerômetros, giroscópios, sensores de pressão, termômetros, multhi-touch.

O Android oferece também aos seus usuários e desenvolvedores uma loja de aplicações, sendo que estas podem ser baixadas e instaladas diretamente no dispositivo. Portanto, os desenvolvedores possuem uma vitrine que apresenta nacional ou globalmente seu produto e os consumidores de apps têm acesso a inúmeras aplicações gratuitas e/ou pagas.

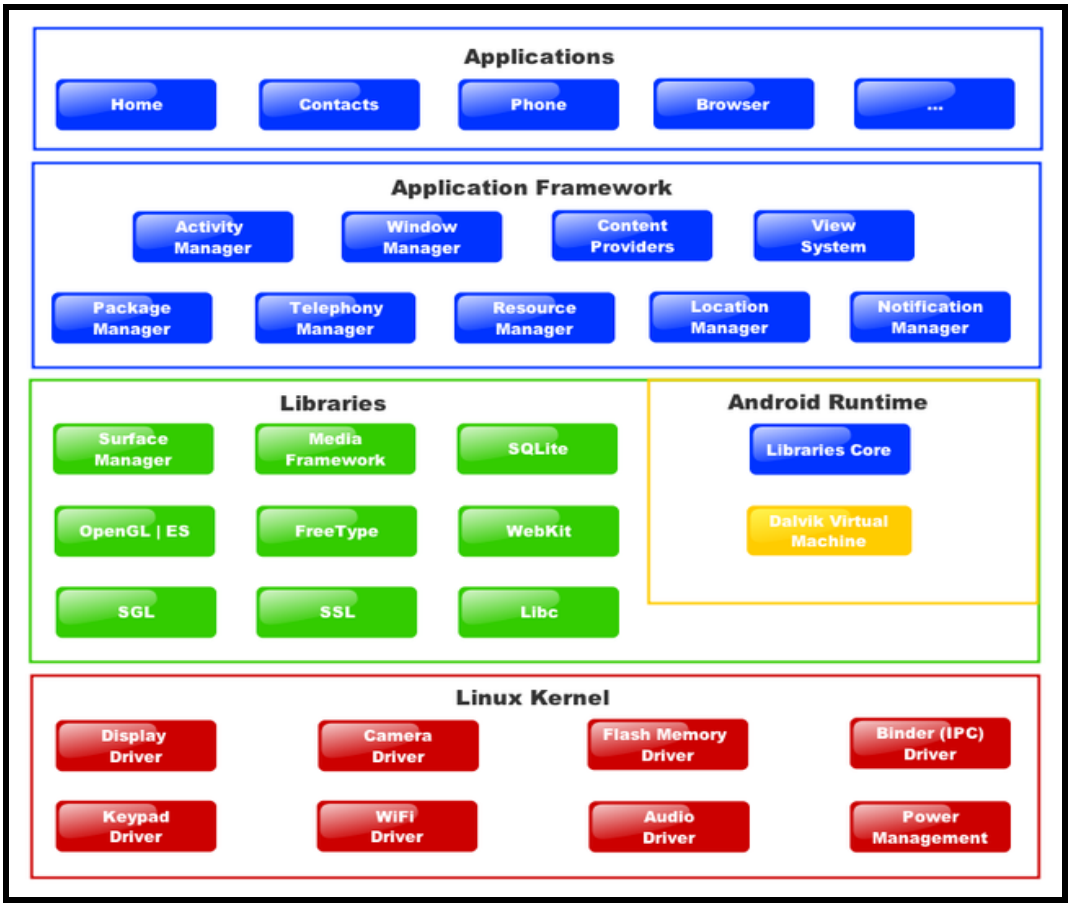
Além disso, ressalta-se que as vendas de dispositivos Android e os downloads de aplicativos estão crescendo exponencialmente. Desde a primeira geração de telefones Android em 2008, bilhões de aplicativos foram baixados do Android Market e, hoje, mais de 500 mil dispositivos são ativados diariamente (DEITEL *et al.*, 2013, P. 9). Assim, fica claro que há grandes oportunidades no mercado para os desenvolvedores de aplicativos Android.

[1] *Global Positioning System:* sistema de navegação por satélite que fornece a um aparelho receptor móvel a sua posição

2.2.1 A Arquitetura Android

A plataforma Android, segundo Amaral *et all* (2012, p.3) baseia-se em uma estrutura composta por um núcleo baseado em Linux, bibliotecas, o Android Runtime, framework de aplicações e as aplicações em geral. Tal estrutura pode ser visualizada na figura 1.

Figura 1 – Arquitetura Android



Fonte: blogteih.com.br/blog

O Android oferece algumas aplicações básicas desenvolvidas na linguagem Java: navegador, mapa, agenda, cliente de e-mail, contatos, entre outros. As bibliotecas são um ponto essencial da plataforma, pois permitem que as aplicações explorem os múltiplos recursos oferecidos por ela.

As aplicações desenvolvidas para a plataforma Android são escritas na linguagem Java. O arquivo gerado pós-compilação é empacotado com os demais recursos consumidos pela aplicação em um arquivo com o sufixo *.apk*. É através deste arquivo que o aplicativo pode ser distribuído e instalado nos dispositivos móveis pelos usuários.

As aplicações são executadas em processos próprios e cada um deles possui sua máquina virtual Java. Conforme descreve Martins (2009, p.6): “Para cada aplicação é

designado um ID de usuário Linux único e as permissões são dadas de forma que os arquivos fiquem visíveis apenas para a aplicação dona."

Os quatro tipos de componentes básicos de uma aplicação Android são instanciados em tempo de execução, no momento em que se torna necessária a sua utilização. Eles são declarados em um arquivo de manifesto estruturado em linguagem XML[2] presente no pacote *.apk*. No arquivo de manifesto também dão declaradas as bibliotecas utilizadas pela aplicação, as permissões de acesso, os requisitos e a versão. Estes componentes são definidos como:

1. Atividade (Activity): geralmente é uma tela para o usuário, ou seja, a interface visual;

2. Serviços: com eles são executados processamentos em segundo plano, logo não apresentam interface visual;

3. Provedores de Conteúdo: disponibilizam para outras aplicações dados específicos da aplicação em execução;

4. Receptores de Broadcast: possuem a função de responder a eventos, permanecendo inativos quando não solicitados.

Os componentes anteriores, com exceção dos provedores de conteúdos, são ativados por mensagens assíncronas, as quais contém a ação a ser executada. Martins (2009, p.7) descreve que tais componentes são ativados através de um objeto denominado *Intent* que carrega a mensagem; dessa forma, para iniciar uma atividade é necessário enviar um *Intent* cujo conteúdo especifique esta intenção. Em suma, "(...) as *Intents* podem ser definidas como mensagens enviadas por um componente da sua aplicação (uma *activity*, por exemplo) para o Android, informando a intenção de inicializar outro componente, da mesma aplicação ou de outra." (MONTEIRO, 2012, p.36)

Ainda segundo Martins:

> Existem dois tipos de *Intents*: explícitos e implícitos. No primeiro, o componente que deve ser executado já é definido explicitamente. No segundo, a escolha do componente é feita pelo sistema operacional que, baseado em alguns critérios, determina qual componente responde melhor àquela intenção naquele momento. (2009, p.7)

Conforme Monteiro (2012, p.36), as Intents são um recurso indispensável do Android, uma vez que permitem que as aplicações colaborem entre si ao disponibilizarem funcionalidades que podem ser reutilizadas, não havendo a necessidade de importar códigos ou dependências para dentro da aplicação. Através dos *Intents*, é possível iniciar novas

[2] *Extensible Markup Language:* formato para a criação de documentos com dados organizados de forma hierárquica.

activities, como fazer uma busca e selecionar um contato do telefone, abrir uma página da web, isso apenas aproveitando funcionalidades já existentes, disponibilizadas pelos aplicativos instalados no aparelho.

"As informações contidas nas Intents são utilizadas pelo Android para localizar o componente adequado, geralmente uma activity, para executar a ação desejada. Quando o nome de componente é informado, o Android inicializa exatamente aquele componente, sem necessidade de avaliar a ação ou categoria." (MONTEIRO, 2012, p.39)

Conforme Monteiro (2012, p. 39), caso o nome do componente não seja informado, o Android precisa consultar quais são os componentes existentes com habilidade de executar a ação desejada e que pertence às categorias existentes na *Intent*. O Android pode ainda procurar por componentes capazes de resolver a *Uri* repassada e também de lidar com o formato dos dados, *o MIME type*, informado.

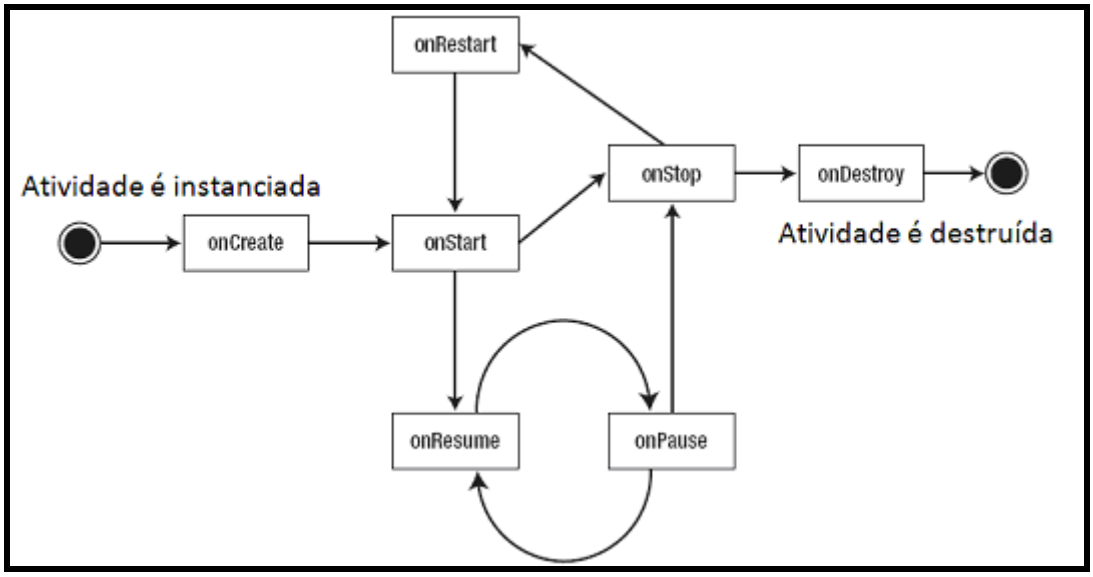
A categoria a que um componente pertence, os dados que ele consegue tratar, bem como a definição das ações que ele está apto a resolver é feito através de *intent* do tipo *filters*, cuja configuração encontra-se no arquivo AndroidManifest.xml.

Quando o usuário inicia uma aplicação e o primeiro componente precisa ser executado, inicia-se um processo em uma única *thread.* Isso é possível, porque o Android possui um mecanismo que faz chamada de procedimentos remotos, que são executados em outros processos.

O ciclo de vida de cada componente de uma aplicação se inicia no momento que ele é instanciado e termina quando ele é destruído. Durante a execução, o estado do componente pode sofrer várias alterações. As alterações entre um estado e outro são notificadas através de chamadas aos métodos especiais*: onStart(), onStop(), onResume(), onPause().*

Na figura 2, observam-se as mudanças de estado de uma atividade durante o seu ciclo de vida, desde o momento que é criada (*onCreate*) até o momento em que é destruída (*onDestroy*).

Figura 2 - Ciclo de Vida de uma Atividade



Fonte: Dall Agnese, 2013

O esquema exposto na figura 2 é explicado por Martins (2009, p.6):

> Durante o tempo entre as chamadas dos métodos *onStart()* e *onStop()*, a atividade pode ser vista pelo usuário. Entre os métodos *onResume()* e *onPause()* a atividade está na frente de todas as outras atividades e o usuário está interagindo com ela. E, devido ao fato de o *onPause()* ser o único método cuja chamada é garantida antes do processo ser eventualmente destruído, ele deve ser utilizado para armazenar, de forma persistente, dados para posterior restauração.

Outros componentes que também sofrem mudanças de estado ao longo de seu ciclo de vida são os serviços e, portanto, possuem métodos que ao serem implementados respondem a essas mudanças.

Os receptores de broadcast, cuja função é responder a eventos, possuem apenas um método de ciclo de vida, o qual é chamado no momento que chega uma mensagem, isto é, quando ele responde à mensagem. Assim, o método é posto em execução e o componente é considerado ativo.

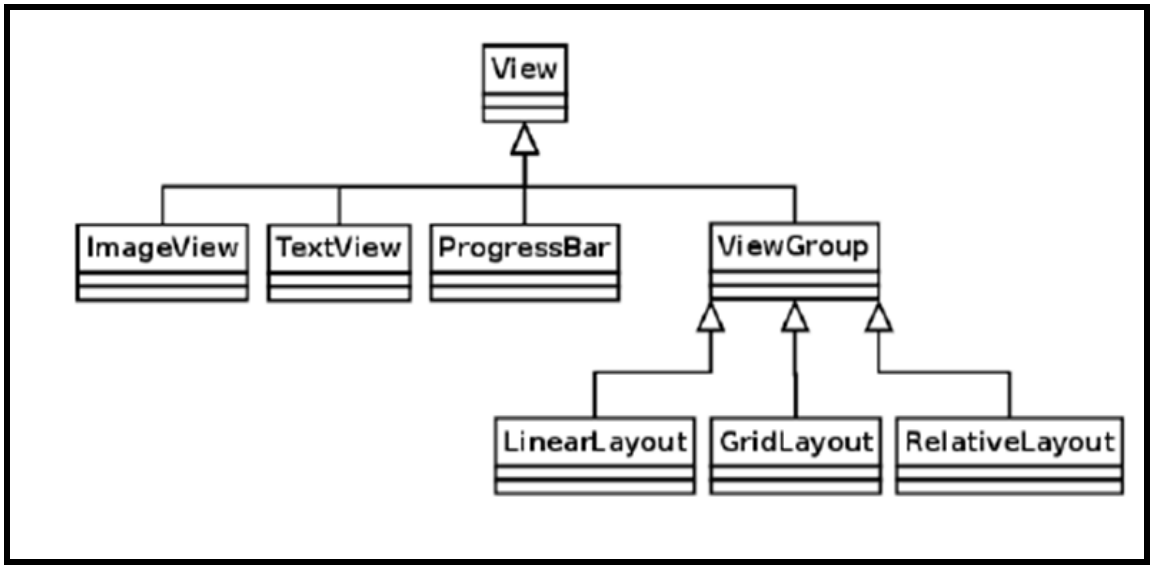
2.2.2 Interface com o usuário

A criação de interfaces gráficas em aplicações Android pode ser obtida de duas maneiras, com a API Java ou através de arquivos XML. A maneira mais eficiente para obter organização e facilidade na manutenção do código fonte, é criar documentos XML para as telas, mantendo o código-fonte separado da parte visual. (LECHETA, 2010, p.34).

Na construção da interface feita através de arquivos de layout baseados em XML, a estrutura básica é uma árvore de elementos XML. Cada folha desta árvore é o nome de uma classe *View* e seus ramos são da classe *ViewGroup*.

Cada *View* é responsável por controlar uma área retangular da tela, na qual podem ser implementados objetos, como, por exemplo, campo de texto e botão – que são chamados *widgets.* Já a classe *ViewGroup* é a base dos layouts com a qual é possível organizar a maneira como os objetos serão apresentados na tela – ela é capaz de conter outras *Views*.

Figura 3 – Hierarquia de Views



Fonte: Monteiro, 2012, p. 51

Na figura 3, observa-se a organização hierárquica das *views*. É possível organizar os objetos de várias formas na tela, entre elas citam-se as formas linear, relativa ou absoluta, no entanto, cada uma delas necessita de um layout específico. Embora, segundo Martins (2009, p.9), seja possível especificar exatamente as coordenadas e o tamanho do componente a ser desenhado na tela com a abordagem *AbsoluteLayout*, é preferível evitá-la. A organização em layouts não absolutos é mais vantajosa, pois possibilita uma melhor adaptação da interface da aplicação às diferentes telas (resolução e tamanho distintos) dos dispositivos nos quais pode ser executada.

Ainda de acordo com Martins (2009, p.9):

> A plataforma disponibiliza diversos *widgets* e layouts prontos que, em conjunto, permitem criar uma interface com o usuário bastante rica. Porém, se necessário, também é possível construir e adicionar componentes customizados. E, quando o objetivo for personalizar objetos de forma uniforme, podem ser aplicados estilos e temas.

Também é possível criar menus padronizados a partir de três tipos: opção, submenu e contexto. A opção é exibida ao se pressionar a tecla menu do dispositivo e, quando um item de menu é selecionado, algumas vezes, revela-se um submenu. O contexto caracteriza-se como um tipo de menu executado ao se pressionar de forma longa um objeto *View* (MARTINS, 2009, p.9).

A aplicação pode ainda se comunicar com o usuário, notificando-o sobre um evento que tenha ocorrido. O Android oferece três formas de notificação do usuário:

1. *Toast*: apropriado para mensagens breves que não exigem interação com o usuário;

2. Diálogo: utilizado para chamar atenção do usuário e solicitar-lhe uma resposta;

3. Barra de Estado. Utilizada para notificar o usuário quando a aplicação está rodando em segundo plano. Neste caso, com a barra de estado, se a aplicação possuir permissão para utilizar certos recursos do dispositivo, poderá emitir um sinal sonoro ou acionar o vibrador do dispositivo.

Apesar de todos os recursos oferecidos pelos dispositivos móveis Android, cabe ao desenvolvedor estar atento às diferentes características dos mesmos, tais como: o tamanho da tela e sua resolução e a presença apenas do teclado virtual, para que ele possa construir uma solução que seja intuitiva e fácil de usar.

Vale lembrar que a plataforma Android pode estar presente em dispositivos com características bem variadas. Além disso, como escreve Martins (2009, p.14), "é necessário projetar a aplicação de forma a possibilitar uma execução adequada aos diversos dispositivos nos quais ela pode estar inserida."

2.3 WEB SERVICE

Cada vez mais, os Web Services são utilizados em aplicações comerciais por permitirem que sistemas escritos em linguagens distintas se comuniquem por meio de serviços (métodos) que são expostos para que outros módulos ou sistemas possam acessá-los. Para isso, se faz uso da tecnologia XML associada ao protocolo SOAP. Esta tecnologia funciona como um canal comum de comunicação e juntos constituem a arquitetura SOA, Service Oriented Architecture ou Arquitetura Orientada à Serviço.

A implementação do protocolo SOAP é possível para as várias linguagens de programação móvel, dentre elas, a plataforma Android. No entanto, no cenário móvel, “a tecnologia teve que ser “enxugada” devido a características intrínsecas aos Web Services.” (Devmedia, 2013) Dessa forma, pode-se trabalhar com estruturas de dados mais complexas, manipular várias linhas retornadas no XML de resposta a um serviço, carregar uma hierarquia de árvore, mesmo com as capacidades restritas dos dispositivos móveis referentes ao processamento e ao espaço de memória reduzido.

Uma importante ação para que se possa consumir um Web Service em Android, é necessário importar a biblioteca KSoap2 para dentro do projeto.

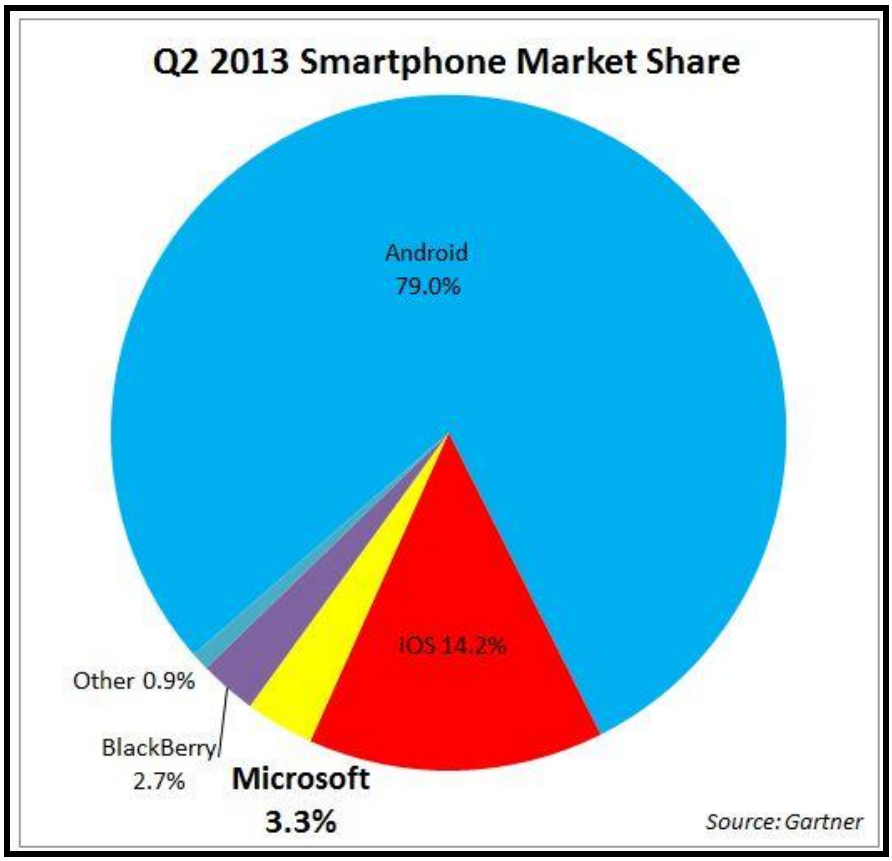
3 PROJETO DO SISTEMA

Neste capítulo são apresentadas as etapas de desenvolvimento do aplicativo móvel, bem como do *web service* propostos. Três requisitos-chave foram observados durante a fase de projeto e desenvolvimento do aplicativo. Esses requisitos, além de outros, são apresentados por Darwin (2012, p.84) como forma de se construir um aplicativo bem-sucedido. No primeiro, ele cita que “o aplicativo deve atender às necessidades do usuário de maneira atraente, única e elegante.” No segundo e no terceiro requisito observado, Darwin diz que “um aplicativo deve ser estável, escalonável, utilizável e responsivo” e que “áreas-chave de funcionalidade devem estar prontamente acessíveis”. Portanto, estes três requisitos serviram de base para o desenvolvimento desta proposta.

3.1 CARACTERÍSTICAS GERAIS DA SOLUÇÃO

O aplicativo móvel Move foi desenvolvido para rodar em aparelhos Android. Optou-se em desenvolver para a plataforma Android, uma vez que a mesma encontra-se em ascensão no Market Share de dispositivos móveis e na época da elaboração do projeto do aplicativo, no segundo trimestre de 2013, conforme dados da Gartner, era esta a plataforma que dominava o mercado mundial com uma fatia de quase 80%, como fica claro na figura 4.

Figura 4 - Market Share Android



Fonte: Gartner

Seu objetivo maior é fornecer uma ferramenta de apoio à venda. Ele substitui os constantes deslocamentos dos vendedores para realizar consultas aos pontos de acesso fixos, em lojas de móveis, eletroeletrônicos e materiais de construção por consultas em um aplicativo instalado em aparelhos móveis de posse dos vendedores. Com ele os vendedores reduzem a peregrinação dentro da loja, ampliam a sua agilidade e eficiência, a satisfação do cliente e o lucro da empresa. Seu desenvolvimento foi focado na simplicidade e praticidade, com telas de fácil manuseio.

O Move foi desenvolvido no IDE Eclipse 3.4, denominada Juno. O Eclipse foi escolhido por ser um ambiente de desenvolvimento gratuito e oficial que oferece um *plugin* com as ferramentas de desenvolvimento Android - ADT (*Android Development Tools*) e além do SDK (*Software Development Kit*).

O *Android Development Tools* possibilita a edição dos *layouts*, o gerenciamento de recursos e conta com uma boa integração com o emulador, bem como a criação de arquivos XML.

O SDK oficial do Android oferece além das bibliotecas para o desenvolvimento, outras importantes ferramentas de apoio. O emulador é um exemplo importante de ferramenta oferecido pelo SDK, ele permite simular um dispositivo smartphone com Android, o que possibilita testes mais próximos do real, além de simular a aparência do aplicativo em diferentes tamanhos de tela.

A versão do Android escolhida para o desenvolvimento do aplicativo foi a 2.3.3 – *Gingerbread* (Android 2.3 até 2.3.7), uma vez que no período em que estava sendo projetado o aplicativo, esta era a versão mais utilizada, como se verifica em reportagem da revista Exame de 10 de agosto de 2012: "O Google informou que a versão mais utilizada do seu sistema operacional móvel é a *Gingerbread* (versões 2.3-2.3.7), presente em 57,5% dos dispositivos Android em serviço no mundo". Atualmente, após a criação da versão *Jelly Bean* (4.1), a versão *Gingerbread* passou a ser a segunda mais utilizada e suportada pela maioria dos equipamentos Android.

O *Web Service* ApoioWS que alimenta e permite o uso das diferentes funcionalidades do aplicativo faz uso da tecnologia XML associada ao protocolo SOAP - Protocolo Simples de Acesso a Objetos. Como servidor de aplicação foi utilizado Glassfish na versão 3.1.2.

3.1.1 Definição de Requisitos

O aplicativo instrumentalizará a tarefa de consultar o preço dos produtos expostos nas lojas e que geralmente são identificados apenas com um código. Essa tarefa é cumprida mediante consulta a algum ponto de acesso fixo dentro da loja, isto é, um computador que se encontra instalado em algum ponto do espaço da loja, o que ocasiona constantes deslocamentos dos vendedores. Neste contexto, o uso do aplicativo, instalado nos dispositivos móveis que serão usados pelos vendedores, tornará mais ágil o atendimento.

A utilização do aplicativo dar-se-á da seguinte forma:

Primeiramente os vendedores efetuarão o *login* informando um nome de usuário e senha e, assim, terão disponíveis todas as funcionalidades do aplicativo.

Na tela inicial, há botões para acesso à tela de consulta aos dados dos produtos, à tela do carrinho de compras, cuja função será exclusivamente a de armazenar os produtos que o cliente consultou o preço e deseja de fato comprar, à tela Sobre e à tela Sair.

Para que a consulta aos dados sobre os produtos expostos na loja – entre eles o preço – possa acontecer, o aplicativo faz uma requisição ao *web service* que busca as informações

sobre o produto na base de dados através do seu código numérico ou de parte de sua descrição. Feito isso, se tem disponível na tela as principais informações do produto, tais como: descrição, preço, estoque (disponibilidade) na loja, categoria e unidade de medida. Além disso, há três botões: o primeiro, para enviar o produto para o carrinho de compras; o segundo, para consultar novo produto; o terceiro, para voltar à tela inicial.

A tela Carrinho de Compras apresenta informações gerais sobre o carrinho montado a partir dos produtos que o cliente deseja comprar, ou seja, o total de itens do carrinho e o valor total do carrinho, além do CPF do respectivo cliente. Nela encontram-se cinco botões: Produtos deste Carrinho, através do qual se tem acesso à lista dos produtos do carrinho; Consultar Novo Produto, que permite adicionar novo produto a um carrinho já existente; Armazenar Carrinho, permite atualizar o carrinho no banco de dados caso seja feita a inserção de novo produto ou a exclusão; Cancelar, usado para retornar à tela com a lista dos carrinhos do CPF pesquisado e Excluir, que possibilita deletar um carrinho do banco de dados.

Retornando ao menu inicial, o botão Consultar Carrinho de Compras permite abrir uma tela aonde, ao se informar o CPF de determinado cliente com um carrinho já cadastrado, é possível visualizar sua lista de carrinho(s) e a partir daí ter acesso às informações armazenadas para cada carrinho cadastrado daquele cliente. É possível, assim, alterar e ou excluir o carrinho selecionado deste cliente.

Para o desenvolvimento do aplicativo foram considerados os requisitos funcionais enumerados a seguir:

1 – Realizar login e logout de usuário (vendedor);

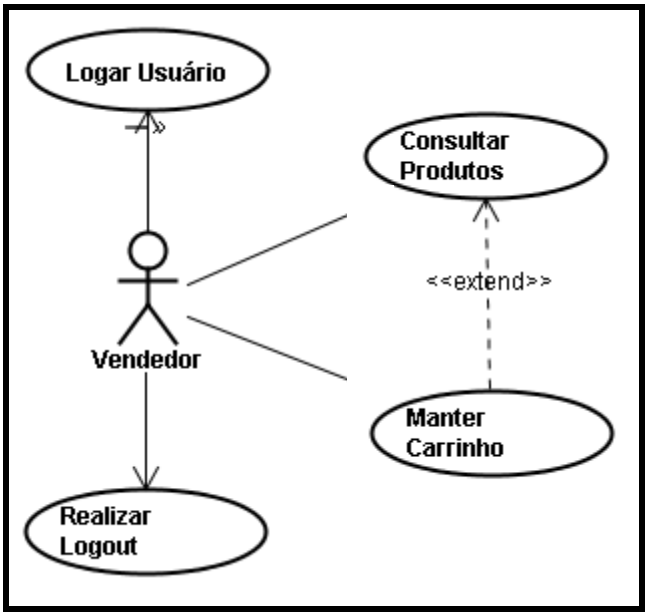
2 – Consultar aos produtos através do código numérico ou parte do nome/descrição;

3 – Manter Carrinho;

A manutenção do carrinho diz respeito às operações de inclusão, alteração, exclusão e listagem dos produtos.

A figura 5 apresenta o diagrama de casos de uso para os requisitos funcionais que foram definidos para o sistema.

Figura 5 – Diagrama de Casos de Uso



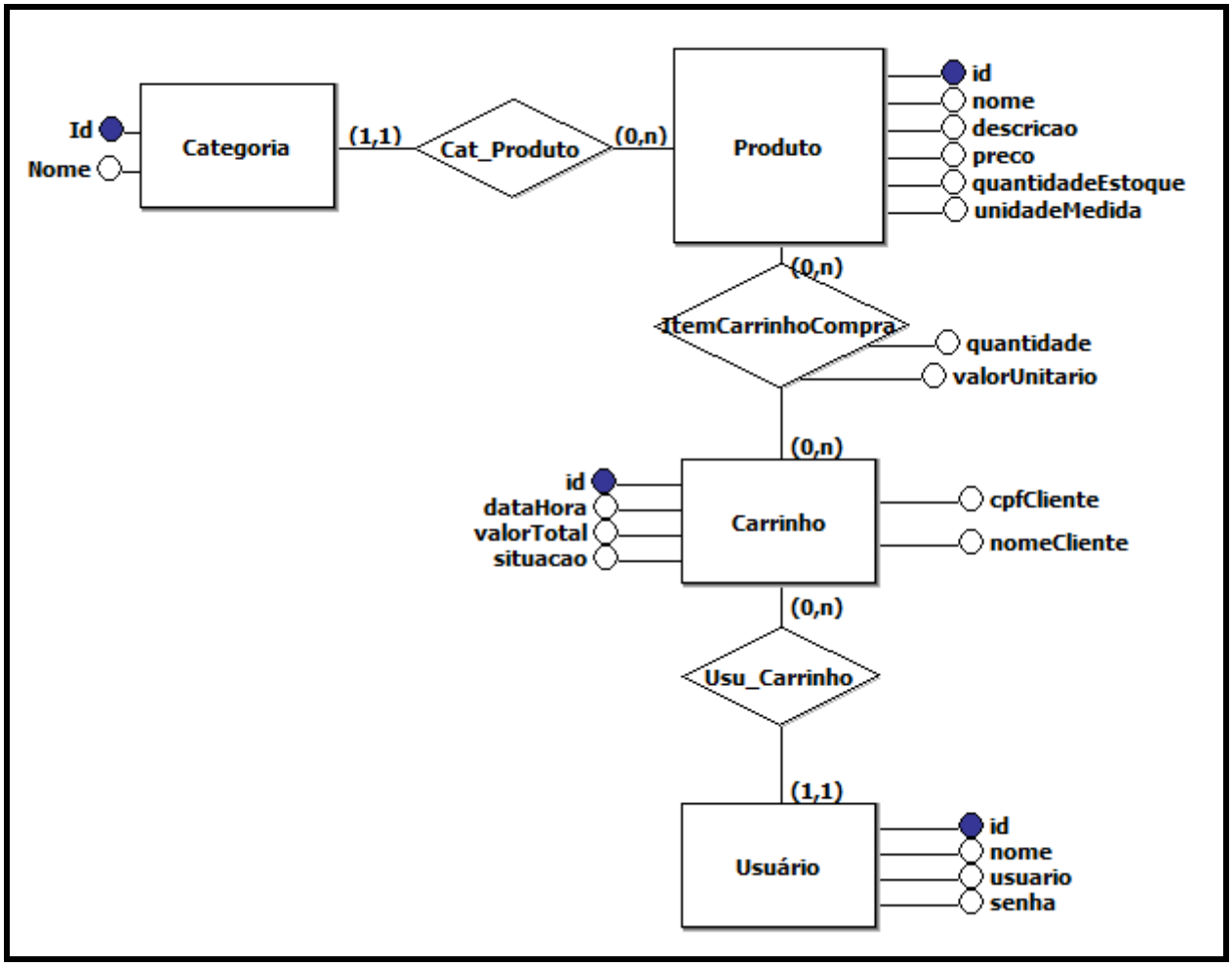
Fonte: do autor

Para melhor visualização e compreensão do comportamento do aplicativo no nível do banco de dados será apresentado o Modelo Entidade-Relacionamento.

3.1.2 Modelo Entidade-Relacionamento

O Modelo Entidade-Relacionamento (MER – Figura 6) descreve de maneira conceitual como serão organizados os dados utilizados no sistema. O projeto do banco de dados para a aplicação conta com as seguintes entidades:

Figura 6 – Modelo Entidade-Relacionamento do Aplicativo Move



Fonte: do autor

USUÁRIOS: Responsável por armazenar as informações sobre os usuários que estarão habilitados para fazer o login e acessar o sistema, no caso serão os vendedores. A tabela usuários conterá os seguintes campos, conforme tabela 1:

Tabela 1 - Usuários

Atributo	**Descrição**	**Tipo**
id #	Identificador do Usuário	Integer
Usuário	Usuário	String
nome	Nome do Usuário	String
Senha	Senha	String

Fonte: do autor

PRODUTOS: Contém os dados sobre os produtos, os quais podem ser consultados pelos usuários do aplicativo (Tabela 2).

Tabela 2 - Produto

Atributo	Descrição	Tipo
id #	Código Interno do Produto	Integer
Nome	Nome do Produto	String
descricao	Descrição do Produto	String
Preco	Preço do Produto	Double
quantidadeEstoque	Quantidade em Estoque	Double
categoria	Categoria do Produto	Categoria
unidadeMedida	Unidade de Medida	String

Fonte: do autor

CARRINHO: Armazena as informações sobre os produtos que o cliente solicitou que fossem inseridos em seu carrinho de compras durante o atendimento, os quais ele pretende adquirir. Essa tabela contém os campos listados na tabela 3:

Tabela 3 - Carrinho Compra

Atributo	Descrição	Tipo
id #	Identificador do carrinho	Integer
dataHora	Data e hora de criação do carrinho	Date
cpfCliente	CPF do Cliente	String
nomeCliente	Nome do Cliente	String
usuarioVendedor	Usuário Vendedor	Usuario
valorTotal	Valor Total	Double
Situação	Situação do Carrinho (orçamento ou compra efetivada)	String

Fonte: do autor

Dessa forma, o vendedor vai adicionando ao carrinho de compras os produtos que o cliente deseja comprar. Após realizada a negociação, o aplicativo móvel exibe a lista de

produtos escolhidos. Então, o vendedor efetiva a venda no sistema de vendas da loja baseado na lista de compras criada.

ITENS_CARRINHO: É responsável por guardar os produtos adicionados no carrinho. Ela contém os campos listados na tabela 4:

Tabela 4 - Itens do Carrinho

Atributo	**Descrição**	**Tipo**
carrinhoCompra #	Carrinho de Compra	CarrinhoCompra
produto#	Produto	Produto
quantidade	Quantidade	Double
valorUnitario	Valor Unitário	Double

Fonte: do autor

CATEGORIA: Responsável por armazenar a categoria dos produtos. Armazena as informações dispostas na tabela 5:

Tabela 5 - Categoria

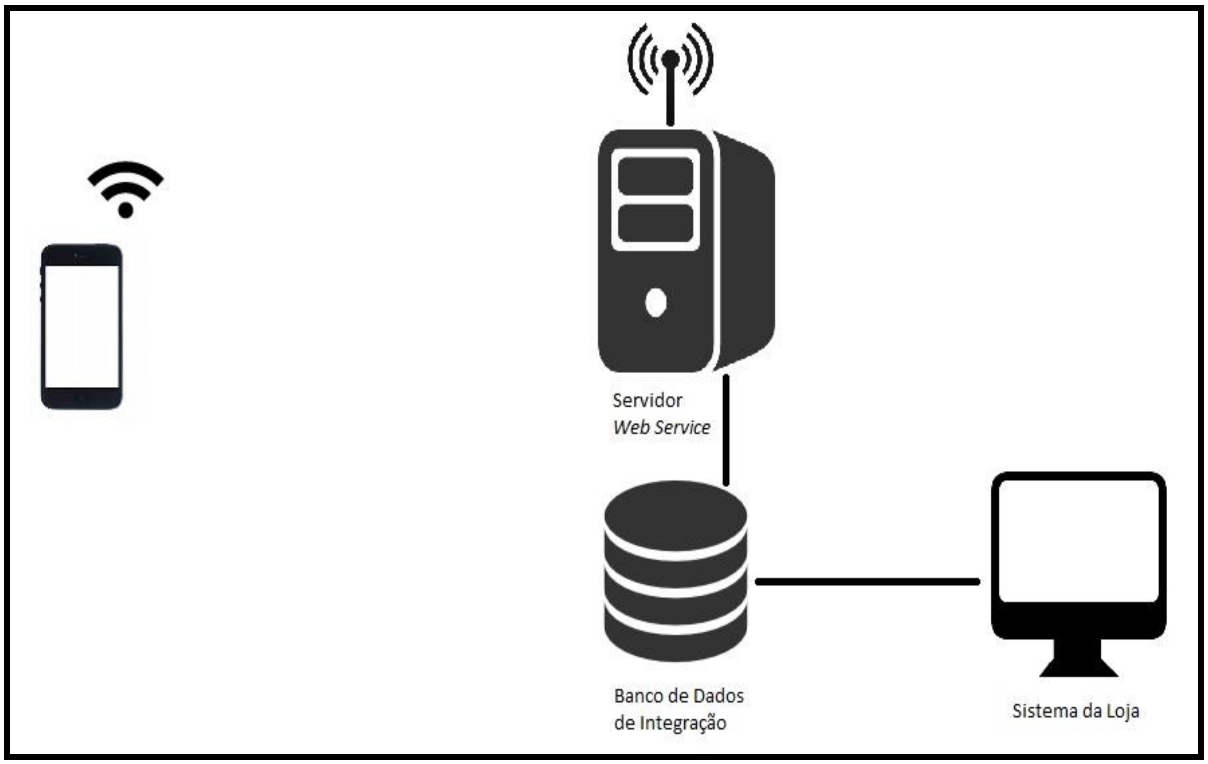
Atributo	**Descrição**	**Tipo**
id #	Identificador da Categoria	Integer
descricao	Descrição	String

Fonte: do autor

3.1.3 Protocolo de Integração

As informações sobre os produtos disponíveis na base de dados da loja são populadas em um banco de dados PostgreSQL intermediário, ou seja, uma base de integração. As informações do banco de integração são consultadas através do *web service* denominado ApoioWS conforme esquema da figura 7:

Figura 7 - Arquitetura do Funcionamento Aplicativo-Web Service-Banco de Dados



Fonte: do autor

Este web service oferta os serviços necessários ao funcionamento do aplicativo Android:

1. Serviço para autenticar os usuários, ou seja, os vendedores e demais funcionários aptos a utilizarem a aplicação. O usuário informa seu “usuario” e sua “senha”, aguardando a autenticação;

2. Serviço para obter produtos pelo seu código, que é responsável por encontrar determinado produto no banco de dados com base no seu código e devolver tais dados para a aplicação;

3. Serviço para obter produto por parte de seu nome ou de sua descrição, que é responsável por encontrar determinado produto no banco de dados com base em um trecho de seu nome/descrição e devolver uma lista de produtos que contenha produtos com aquele trecho do nome.

4. Serviço para cadastrar Carrinho de Compra, que armazena do Carrinho de Compras de determinado cliente no banco;

5. Serviço para alterar o Carrinho de Compras, que é responsável pelas atualizações feitas no carrinho, tais como adição ou deleção de produtos.

6. Serviço para excluir um Carrinho de Compras, que deleta do banco de dados um carrinho de determinado cliente;

7. Serviço para obter Carrinho de Compras a partir do CPF do cliente, encontra os carrinhos de compras de determinado cliente, armazenados no banco de dados, com base em seu CPF.

8. Serviço para obter produtos do Carrinho de Compras de determinado cliente, encontra os produtos do carrinho de compras de um cliente a partir no código identificador do carrinho.

Esses oito serviços contemplam todas as funcionalidades oferecidas pela aplicação.

3.2 FUNCIONAMENTO

O aplicativo Move foi desenvolvido com o uso de duas tecnologias atuais na área da mobilidade: *web service* e Android. Dessa forma, o usuário vendedor pode executar sua tarefa de consulta ao preço dos produtos de forma mais ágil e efetiva, tendo na palma de sua mão a ferramenta adequada.

3.2.1 Aplicativo Move

A partir dos requisitos definidos para o desenvolvimento do aplicativo Move, foram construídas as telas e implementadas as funcionalidades das mesmas. A seguir são descritas cada uma delas e apresentadas suas respectivas funcionalidades.

Como o *Web Service* realiza consultas a usuários, produtos e carrinhos, foram criadas três classes clientes no aplicativo para conectar-se ao *web service* e receber seu retorno. Assim as requisições solicitadas ao *Web Service* ocorrem através das classes WSConnectionUsuario, WSConnectionProduto e WSConnectionCarrinho.

As três classes estendem da classe DadosConexao, na qual estão definidos os dados que serão necessários para fazer a conexão com o *web service*, como as *Strings* que compõem a URL do serviço, e implementam a classe *Serializable*. A comunicação das mesmas com os serviços se deu com a inclusão da biblioteca Ksoap2 ao projeto.

A classe cliente WSConnectionUsuario possui apenas o método autenticarUsuario e é chamada logo na primeira tela para realizar o login do usuário do aplicativo. Assim é encaminhada a primeira requisição em um SoapSerializationEnvelope ao web service com o

"usuário" e a "senha". A estrutura de definição da requisição e do envelope é a mesma nos métodos das três classes clientes, como apresentado a seguir:

Figura 8 - Trecho do método autenticarUsuario

```
28          //Criação da requisição
29          SoapObject request = new SoapObject(NAMESPACE, AUTENTICAR_USUARIO);
30          //Criação do envelope
31          SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
32                  SoapEnvelope.VER11);
33          //O envelope recebe a requisição
34          envelope.setOutputSoapObject(request);
35
36          StringBuffer paramUsuario = new StringBuffer();
37          StringBuffer paramSenha = new StringBuffer();
38
39          // Definição dos parâmetros do servico
40          paramUsuario.append(usuario);
41          paramSenha.append(senha);
42
43          // Definiçao do nome dos parâmetros (chave)
44          request.addProperty("usuario", paramUsuario.toString());
45          request.addProperty("senha", paramSenha.toString());
46
47          Usuario usuarioRetorno = new Usuario();
48
49          HttpTransportSE httpTransport = new HttpTransportSE(URL, TIMEOUT_CONEXAO);
```

Fonte: do autor

Na figura anterior, que mostra um trecho do método autenticarUsuario da classe WSConnectionUsuario se tem, primeiramente, a criação da instância de um objeto do tipo SoapObject. A seguir é instanciado um objeto do tipo SoapSerializationEnvelope e adicionado o SoapObject denominado "request" no mesmo. Então são definidos os parâmetros que representam as propriedades do objeto SoapObject criado e estes são adicionados ao "request". A chamada ao servidor de aplicações é feita usando o HttpTransportSE sendo passado o SoapSerializationEnvelope com a requisição gerada.

Após a execução do serviço, o envelope irá conter seu retorno. Neste caso, como se pode ver na figura abaixo, o retorno é um objeto "usuario" do qual são extraídas duas propriedades através do método *getProperty* e que serão necessárias durante a execução da aplicação: o "id" e o "nome" do usuário.

Figura 9 - Sequência do método autenticarUsuario

```
48
49          HttpTransportSE httpTransport = new HttpTransportSE(URL, TIMEOUT_CONEXAO);
50          try {
51              //É realizada a chamada do servico
52              httpTransport.call("", envelope);
53              //Se foi encontrado o usuário no BD...
54              if (envelope.getResponse() != null) {
55                  SoapObject ob = (SoapObject) envelope.getResponse();
56
57                  usuarioRetorno.id = Integer.parseInt(ob.getProperty("id")
58                          .toString());
59                  usuarioRetorno.nome = ob.getProperty("nome").toString();
60
61                  return usuarioRetorno;
62
63              } else { // Se não encontrou o usuario - acesso negado
64
65                  usuarioRetorno = null;
66              }
```

Fonte: do autor

A classe cliente WSConnectionProduto possui dois métodos, cujo objetivo é requisitar dados referentes ao(s) produto(s) armazenados na base de dados de integração. O método obterProdutoByCodigo recebe, como parâmetro, um código que irá compor a requisição feita ao *web service*. Este método poderá receber como retorno do *web service* um objeto do tipo Produto, que estará contido no SoapSerializationEnvelope, ou uma resposta do tipo *null* e disponibilizará este resultado para o aplicativo.

Já o método obterProdutoByDescricao recebe, como parâmetro, uma String denominada “descricao” que contém pelo menos três caracteres. A “descricao” compõe a requisição feita ao *web service*. Como retorno do web service, este método receberá um *array* contendo a lista dos produtos com a descrição enviada por parâmetro.

A classe cliente WSConnectionCarrinho possui cinco métodos que fazem requisições referentes ao(s) carrinho(s) de compras para o *web service*.

O método cadastrarCarrinhoCompra recebe, como parâmetro, um objeto do tipo CarrinhoCompra que contém as propriedades do carrinho de compras, inclusive os itens daquele carrinho. Como retorno deste método se tem um dado do tipo booleano que determina o sucesso ou o fracasso ao se cadastrar o carrinho de compras.

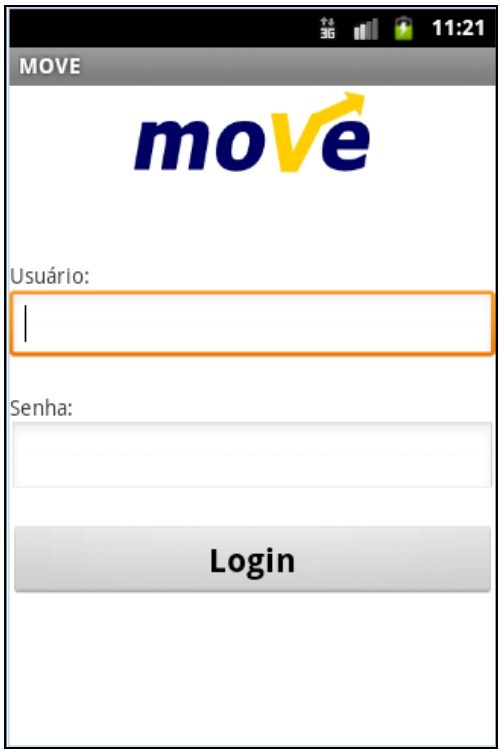
O método obterCarrinhoCompraByCpfCliente recebe, como parâmetro, uma String que contém o CPF do cliente e tem, como retorno, a lista de carrinhos de compras daquele CPF. O método obterProdutosCarrinhoCompraCliente complementa o método anterior ao receber o código de um dos carrinhos de compras e retornar uma lista com seus itens.

Por fim, por meio da classe cliente WSConnectionCarrinho se pode excluir um carrinho com o método excluirCarrinhoCompra ou alterá-lo pelo método alterarCarrinhoCompra. O método de exclusão recebe um objeto do tipo carrinhoCompra e retorna um dado do tipo booleano como resultado da requisição feita ao *web service*. O método de alteração também recebe um objeto do tipo carrinhoCompra e, como retorno, fornece um dado do tipo booleano que determina o sucesso ou fracasso da requisição feita ao *web service*.

Assim, as três classes clientes apresentadas, dão suporte a todo o funcionamento da aplicação que será descrito na sequência.

Na tela inicial do aplicativo, é feito o *login* dos usuários autorizados a consultar as informações armazenadas no banco de dados através do *Web Service*. Através desta tela é estabelecida a primeira conexão com o *Web Service* para autenticar o usuário.

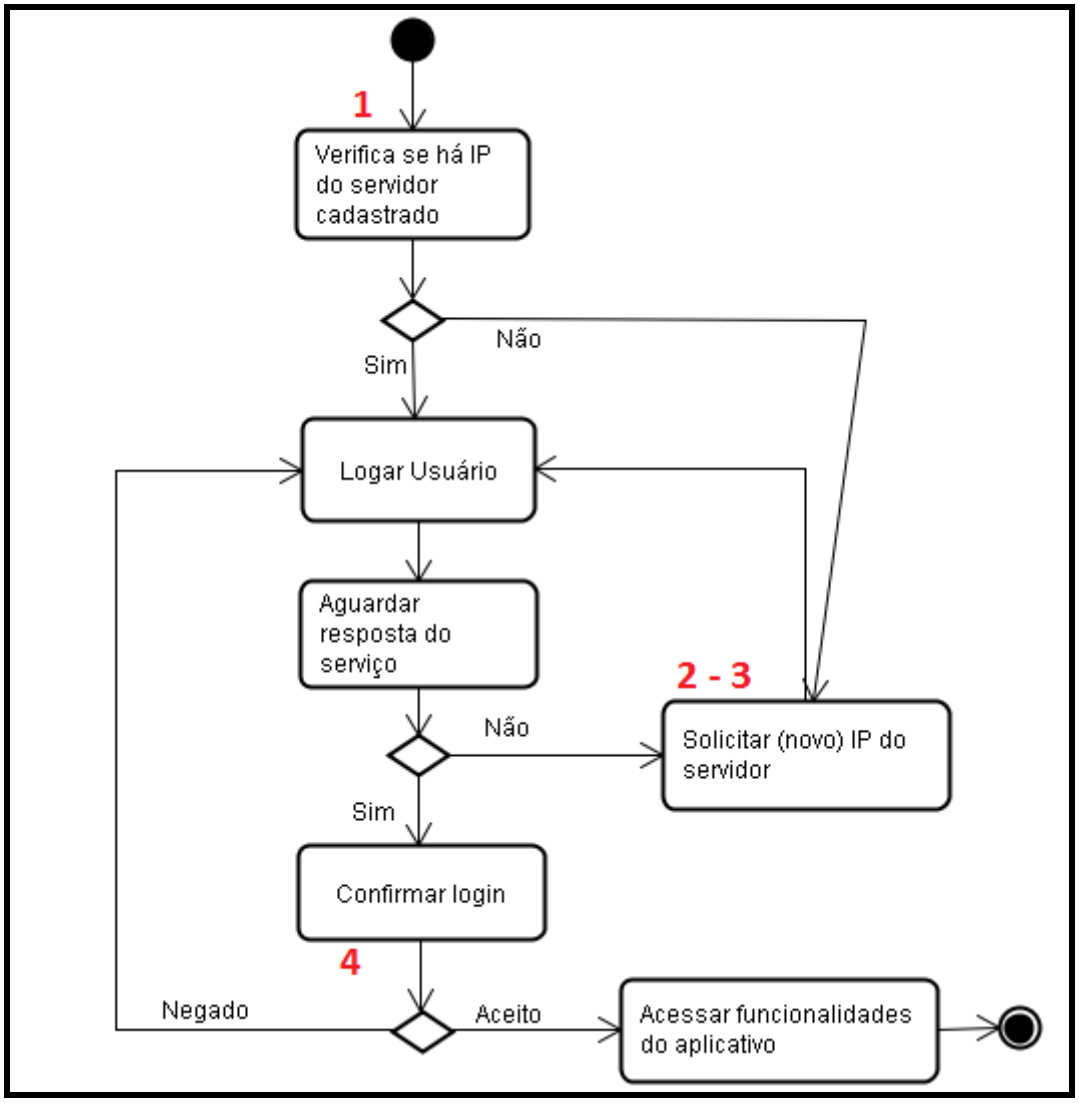
Figura 10 – Tela Inicial do Aplicativo - *login*



Fonte: do autor

Ao clicar no único botão desta tela é esperado um dos quatro resultados que podem ser visualizados na figura 11 e que são descritos na sequência:

Figura 11 – Diagrama de Atividades - *login*



Fonte: do autor

1 - Caso seja a primeira utilização do aplicativo, será aberta uma tela que solicita o endereço IP do servidor no qual está o Web Service. O endereço IP é armazenado localmente no banco SQLite e novamente é solicitado o login. Se o endereço IP for o correto e o serviço estiver rodando normalmente, será feita a verificação da autenticação do usuário.

Figura 12 – Tela de Identificação do Servidor

Fonte: do autor

2 - Se o endereço IP do servidor tiver sido alterado, o usuário será informado sobre uma possível falha na conexão e também será aberta a tela na qual será possível informar um novo endereço IP. Com esta ação, será realizado um UPDATE na tabela que armazena o IP no banco SQLite e uma nova tentativa de login solicitada. Desta vez, se o endereço IP for o correto e o serviço estiver rodando normalmente, será feita a verificação da autenticação do usuário.

3 – Entretanto, se o serviço não responder à requisição do aplicativo dentro do *timeout* – tempo limite - estabelecido de 5 segundos, o usuário será informado sobre uma possível falha de conexão e deverá fazer nova requisição.

4 – No entanto, se o serviço estiver rodando normalmente, o mesmo verificará se aquele usuário se encontra cadastrado no banco de dados PostgreSQL com a senha informada e retornará uma confirmação de login aceito ou uma negação.

Se o usuário que fez a solicitação de login estiver cadastrado, terá seu acesso às funcionalidades do aplicativo liberado e visualizará a tela com o menu inicial. No menu inicial encontram-se quatro botões: Consultar Produto, Consultar Carrinho, Sobre e Sair.

Figura 13 – Tela do Menu Inicial

Fonte: do autor

O botão Sair faz o *logout* do usuário e o botão Sobre abre uma tela com dados gerais do aplicativo.

Figura 14 – Tela Sobre

Fonte: do autor

O botão Consultar Produto redireciona o usuário à tela de consulta aos produtos disponíveis na base de dados intermediária. Esta consulta pode ser realizada de duas maneiras, pelo código de registro do produto ou por parte de seu nome ou descrição, sendo necessário informar no mínimo três caracteres.

Figura 15 – Tela para Consultar Produto

Fonte: do autor

Ao tocar o botão Consultar, a classe ProdutoWSConnection é instanciada e dependendo do parâmetro informado a consulta pode ser feita através do código do produto ou através de uma *String* com parte da descrição do produto.

O retorno da consulta de um produto pelo código identificador é um SoapObject do tipo produto. Este retorno é apresentado ao usuário na tela Dados Produto. Nela é possível visualizar o código do produto, seu nome e descrição, preço, quantidade em estoque e unidade de medida.

Na tela Dados Produto são permitidas algumas ações ao usuário como adicionar este produto a um carrinho de compras, retornar a tela de consulta a um novo produto ou retornar ao Menu Inicial.

Figura 16 – Tela de Dados Produto

Fonte: do autor

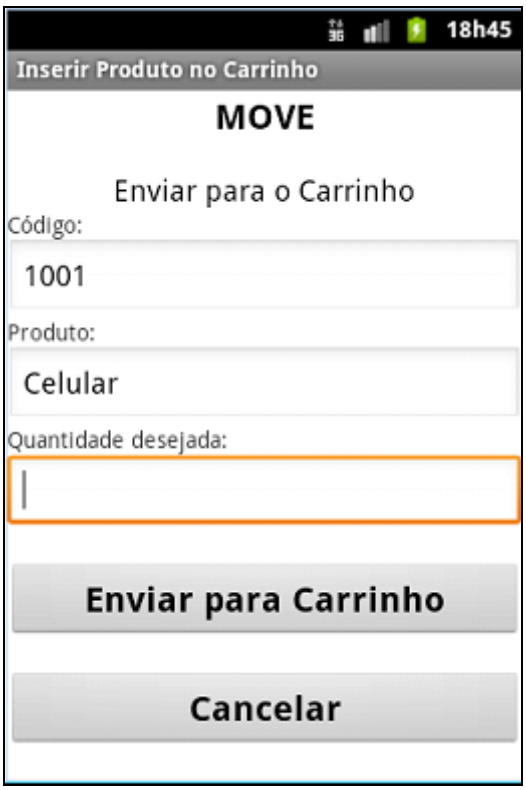
O retorno da consulta a um produto através de parte de seu nome ou descrição é uma lista de produtos. Esta lista é apresentada ao usuário em uma nova tela e nela é possível selecionar o produto desejado com um clique através do método onClickItem. Com isso, o objeto selecionado passa a ser apresentado na tela Dados Produto. Neste caso, o usuário também pode adicionar este produto a um carrinho de compras, retornar a tela de consulta a um novo produto ou retornar ao Menu Inicial.

Figura 17: Tela Lista de Produtos – Consulta pela Descrição

Fonte: do autor

Quando o usuário clica no botão Enviar para Carrinho, uma tela é aberta solicitando a quantidade do produto desejado. Então, ao informar a quantidade e confirmar no botão Enviar para Carrinho é aberta a tela Carrinho de Compras. Também é possível cancelar a ação. Neste caso o usuário é redirecionado ao Menu Inicial.

Figura 17 – Tela para Enviar Produto para Carrinho Novo



Fonte: do autor

Retornando ao Menu Inicial, através do botão Consultar Carrinho, o usuário visualiza a tela Consultar Carrinho na qual deve informar o CPF do cliente para realizar a consulta.

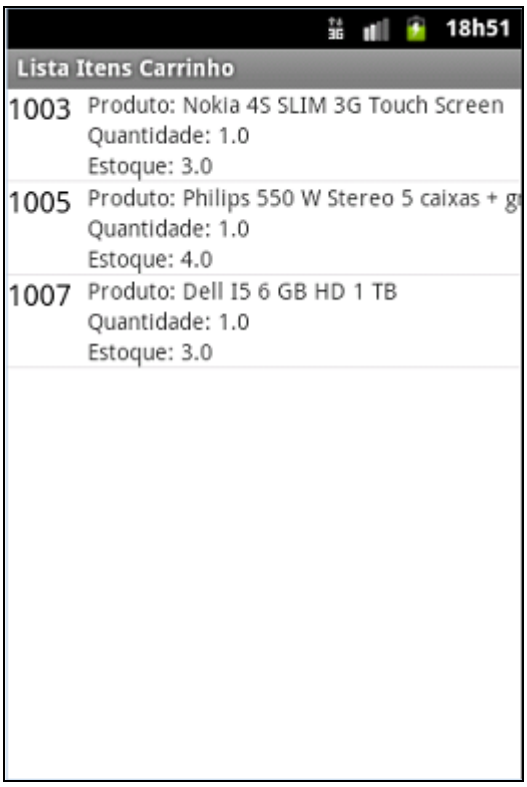
Figura 18 – Tela para Consultar Carrinho

Fonte: do autor

Ao clicar no botão Consultar, o usuário tem acesso a uma lista em uma nova tela, na qual pode visualizar os carrinhos de compras do cliente armazenados com base em seu CPF. Esta lista permite a seleção de um dos carrinhos através de um toque.

Ao selecionar um carrinho, o usuário abre a tela que apresenta os dados do carrinho, ou seja, quantidade total desejada e valor total, além da situação do carrinho. É também possível visualizar a lista com os itens do carrinho e as informações específicas de cada item.

Figura 19 – Tela de Itens do Carrinho



Fonte: do autor

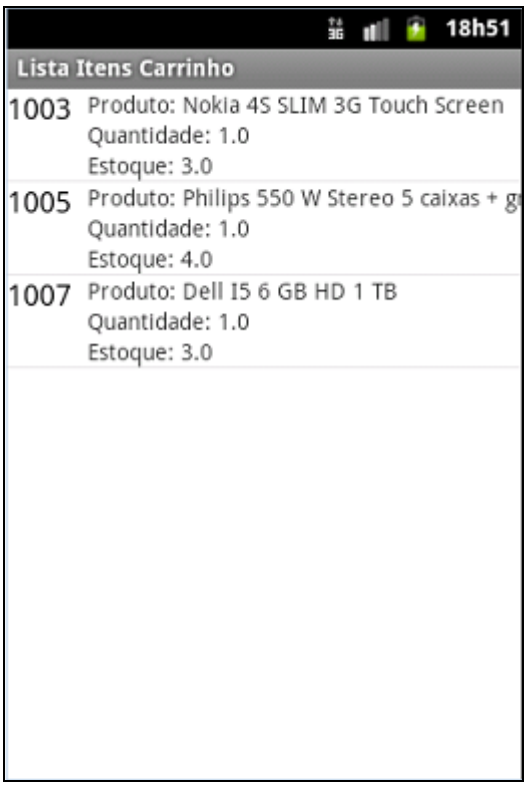
Na tela Carrinho de Compras existem botões que oferecem diferentes funcionalidades ao usuário. É possível verificar quais são os produtos do carrinho, consultar um novo produto e adicioná-lo ao carrinho já existente, armazenar novamente o carrinho no banco de dados se ele tiver sido alterado, excluir o carrinho que está sendo visualizado ou cancelar as ações sobre o carrinho.

Figura 20 – Tela do Carrinho de Compras

Fonte: do autor

Selecionando o botão Produtos deste Carrinho o usuário vê a lista dos produtos que pertencem aquele carrinho, por ora denominados de itens do carrinho.

Figura 21 – Tela com Lista de Itens do Carrinho de Compras



Fonte: do autor

Ao escolher um dos produtos através de um clique, o usuário visualiza a tela com todos os dados do produto escolhido e tem a possibilidade de executar as seguintes ações: Excluir do Carrinho, Consultar Novo Produto ou Voltar.

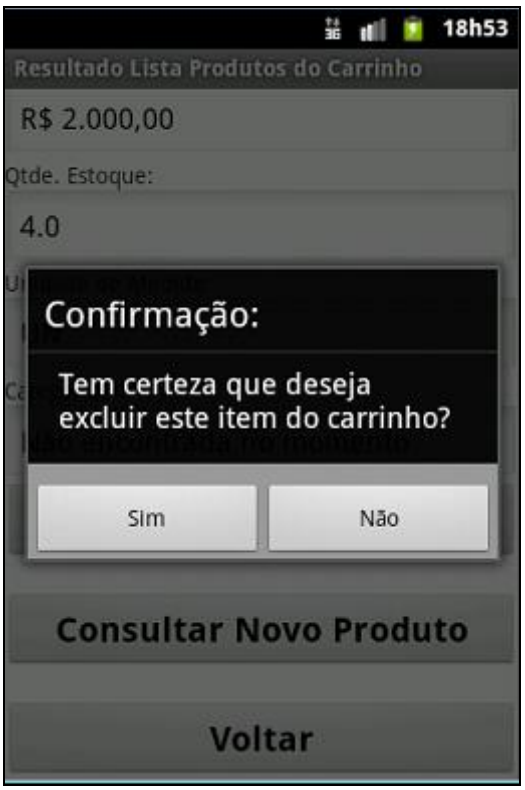
Figura 22 – Tela de Exclusão de Itens do Carrinho de Compras

Fonte: do autor

Na tela apresentada anteriormente pode-se voltar para a lista dos produtos daquele carrinho através do botão Voltar; consultar um novo produto que se deseja adicionar ao carrinho; ou excluir do carrinho o produto que está sendo apresentado na tela pelo botão Excluir do Carrinho.

Selecionando Excluir do Carrinho, é aberta uma tela de alerta que solicita a confirmação por parte do usuário sobre a exclusão solicitada.

Figura 23 – Tela de Confirmação de Exclusão de Item do Carrinho de Compras



Fonte: do autor

Se o usuário confirmar clicando no botão OK, o produto será excluído e será visualizada a tela do carrinho de Compras com o novo valor total do carrinho. Porém, se a opção de exclusão for negada, o usuário voltará para a lista com os produtos do carrinho.

Assim sendo, o aplicativo Move oferece funcionalidades as quais dão suporte a etapa de venda que ocorre entre vendedor e cliente. Com o aplicativo desenvolvido, o vendedor pode tornar-se mais ágil e efetivo, realizando seu trabalho com mais qualidade e êxito.

4 CONSIDERAÇÕES FINAIS

O desenvolvimento do aplicativo aqui apresentado passou por diferentes etapas. Todas contribuíram para o amadurecimento da ideia inicial e permitiram definir as funcionalidades indispensáveis para que a proposta fosse alcançada com êxito. Dessa forma, se estudou as tecnologias Android e *Web Service*, e a comunicação entre as duas utilizando a biblioteca Ksoap2.

Como resultado se construiu uma ferramenta de apoio à venda com uma arquitetura distribuída e que acessa uma base de dados de integração. Tais características permitem que o aplicativo Move seja utilizado pelas lojas sem prejuízos e necessidades de ajuste ao sistema de vendas pré-existente.

O aplicativo ainda não está pronto para seu uso efetivo, pois necessita que sejam feitos testes com um grande volume de informações na base de dados. É preciso verificar o comportamento da ferramenta em um cenário onde há vários aparelhos rodando o Move e fazendo requisições ao *web service*, o qual estará consultando uma base de dados de integração, com dezenas de milhares de produtos cadastrados. Somente a partir daí será possível determinar se o aplicativo já oferece reais condições de uso ou se há ajustes a serem feitos.

Assim sendo, se tem como resultado da pesquisa a ferramenta Move de apoio à venda em lojas de móveis, eletroeletrônicos e materiais de construção que, embora se encontre em uma versão de testes, aspira ser utilizada em larga escala no atendimento aos clientes, de modo que os vendedores se tornem mais ágeis e efetivos no desempenho de sua função.

4.1 TRABALHOS FUTUROS

A próxima etapa no desenvolvimento da ferramenta Move é executar testes em uma escala maior para que se possa determinar as suas fragilidades em um ambiente real de uso e, então, se fazer as correções necessárias.

Por conseguinte, pretende-se desenvolver um integrador que lerá e escreverá na base de dados de integração, permitindo que os dados sejam constantemente atualizados, além de tornar possível à loja acessar o carrinho de compras de determinado cliente e com ele fazer a efetivação da venda no balcão. Neste caso, o vendedor marcará a situação da lista de compras como ‘Efetivada’, ou, caso o cliente decida aguardar para efetivar a compra, a lista será marcada como “Orçamento”.

REFERÊNCIAS

Exame.com. Gingerbread é a verão mais usada do Android. Home/Tecnologia, 10 de Set 2012. <http://exame.abril.com.br/tecnologia/noticias/gingerbread-e-a-versao-mais-usada-do-android--2/> Acesso em: 30 Abr 2013.

<http://www.devmedia.com.br/artigo-webmobile-23-utilizando-web-services-no-google-android/12322> Acesso em: 19 Ago 2013.

AMARAL, Luis Fernando D. do et al. Utilização de Códigos QR em Dispositivos Móveis para Cadastro e Compartilhamento Automático de Informações Pessoais. **Anais do VIII Workshop de Visão Computacional.** Goiás, 2012.

BLOGTEIT. Disponível em: <http://blogteih.com.br/blog/2012/06/plataforma-android-visao-geral-parte-2/>
Acesso em: 08 mai. 2013.

COMACHIO, Vanderson. **Funcionamento de banco de dados em Android**: Um estudo experimental utilizando SQLite. 2011.Trabalho de Diplomação (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – UTFPR, Medianeira, 2011.

DALL AGNESE, Josué Fernandes. Tudo sobre o Google Android OS. Disponível em: <http://www.oficinadanet.com.br/artigo/outros_sistemas/google_android_os> Acesso em: 10 mai. 2013.

DARWIN, Ian F. Android cookbook. São Paulo: Novatec Editora, 2012.

DEITEL, Paul [et al.]. Android para Programadores: uma abordagem baseada em aplicativos. Tradução: João Eduardo Nóbrega Tortello. Porto Alegre: Bookman, 2013.

GARTNER, Q2 2013 Smartphones Market Share. Disponível em: <http://www.minyanville.com/sectors/technology/articles/Microsoft-By-the-Numbers253A-What-You/8/28/2013/id/51484> Acesso em: 29 mai. 2014.

LECHETA, Ricardo R. **Google Android**: aprenda a criar aplicações para dispositivos móveis com Android SDK / Ricardo R. Lecheta. 2. ed. São Paulo : Novatec Editora, 2010.

MARTINS, Rafael J. Werneck de A. **Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android**. 2009. Projeto Final (Graduação em Engenharia de Computação) – PUC RJ, Rio de Janeiro, 2009. Disponível em: <http://www.icad.puc-rio.br/~projetos/android/files/monografia.pdf> Acesso em: 05 abr. 2013.

MONTEIRO, João Bosco. **Google Android**: Crie aplicações para celulares e tablets. Editora Casa do Código, 2012.

PRATES, Glaúcia Aparecida; OSPINA, Marco Túlio. Tecnologia da informação em pequenas empresas: fatores de êxito, restrições e benefícios. Junho de 2004. Disponível em: <http://www.scielo.br/scielo.php?pid=S1415-65552004000200002&script=sci_arttext> Acesso em: 11 jun. 2014