UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

PROTÓTIPO DE APLICATIVO DE FORÇA DE VENDAS PARA DISPOSITIVOS MÓVEIS BASEADOS NA PLATAFORMA ANDROID

FÁBIO DE OLIVEIRA

BLUMENAU
2014

2014/1-04

FÁBIO DE OLIVEIRA

PROTÓTIPO DE APLICATIVO DE FORÇA DE VENDAS PARA DISPOSITIVOS MÓVEIS BASEADOS NA PLATAFORMA ANDROID

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Francisco Adell Péricas, Mestre - Orientador

BLUMENAU
2014

2014/1-04

PROTÓTIPO DE FORÇA DE VENDAS PARA DISPOSITIVOS MÓVEIS BASEADOS NA PLATAFORMA ANDROID

Por

FÁBIO DE OLIVEIRA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas, Mestre – Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB

Blumenau, 07 de julho de 2014.

Dedico este trabalho aos familiares, amigos, professores e a todos que torceram, acreditaram e contribuíram para a realização deste.

AGRADECIMENTOS

Aos meus pais, pelo incentivo, compreensão e apoio mesmo depois de tantos anos.

Ao Leonardo Ribeiro, Felipe Oliveira e a Danieli de Paula pelo incentivo, sugestões e dedicação.

Ao meu orientador, professor Francisco Adell Péricas pela motivação, ensinamentos, conselhos, tranquilidade e por ter acreditado na conclusão deste trabalho.

Ao professor Wilson Pedro Carli, Coordenador do curso de Sistemas de Informação pela confiança, ensinamentos e por acreditar na minha capacidade mesmo depois de tantas tentativas.

Impossível é uma palavra que só os fracos conhecem.

Autor Desconhecido

RESUMO

No mundo dos negócios é fundamental que as organizações criem diferenciais para se destacarem no mercado. Uma das maneiras de se destacar é utilizar o conceito de força de vendas. A empresa Sold Out tem a necessidade de criar um diferencial e mudar o seu processo atual de vendas. Hoje a empresa utiliza vendas por consignação, um processo considerado vulnerável e sem muito controle sobre os produtos vendidos. Para melhorar o dia-a-dia da empresa, pensou-se no desenvolvimento de um aplicativo para força de vendas. Esse aplicativo foi desenvolvido para o sistema operacional Android. Foi realizada integração via *web services* entre aplicativo e servidor, centralizando assim o banco de dados em apenas um local. Os resultados se mostraram satisfatórios, pois os representantes terão facilidades em seus dias no que se refere ao processo de vendas e a empresa terá mais controle sobre o que é vendido e sobre seu estoque.

Palavras-chave: Força de Vendas. Android. *Web Services*.

ABSTRACT

In the business world it is critical that organizations create differences to detach in the market. One of the ways to stand out is by using the concept of the sales force. The company Sold Out has need to create a differential and change your current sales process. Today the company uses sales per consignment, a process considered to be vulnerable and without much control over the goods sold. To improve the day-to-day business, it was thought in the development of an application for sales force. This application was developed for the Android operating system. It was carried out via web services between application and server, thus centralizing the database in one place. The results were satisfactory, because representatives have facilities in their days with regard to the sales process and the company will have more control over what is sold on its stock.

Key-words: Sales Force. Android. Web Services.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ADT - Android *Development Tools*

APK - Android *Package File*

EA - *Enterprise Architect*

FURB - Universidade Regional de Blumenau

HTTP - *Hyper Text Transfer Protocol*

JPA - *Java Persistence API*

JPQL - *Java Persistence Query Language*

MMA - *Mixed Martial Arts*

OP - Ordem de Produção

POJO - *Plain Old Java Object*

REST - *Representational State Transfer*

SDK - Android *Software Development Kit*

SOAP - *Simple Object Access Protocol*

TCC - Trabalho de Conclusão de Curso

XML - *Extensible Markup Language*

W3C - *World Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

Em meio às diversas transformações que impactam as empresas, a tecnologia está criando alguns novos desafios. O livre fluxo de informações na internet bem como o estabelecimento de novas formas de se fazer negócio faz com que uma das mais tradicionais atividades de mercado como a administração de vendas assuma importância cada vez maior (GOLDBERG, 2005). A informação é fundamental para o sucesso na área de vendas, pois se não está por dentro de tudo que acontece no mercado, corre-se o risco de perder vendas para a concorrência. A utilização de celulares para se manter informado é fundamental no mercado de vendas.

O mercado de celulares está crescendo cada vez mais. Estudos mostram que hoje em dia mais de 3 bilhões de pessoas possuem um aparelho celular, e isso corresponde a mais ou menos metade da população mundial. O mercado corporativo também está crescendo muito e diversas empresas estão buscando incorporar aplicações móveis no seu dia a dia para agilizar seus negócios e integrar as aplicações móveis com seus sistemas de *back-end*. O Android é a resposta da empresa Google para ocupar este espaço. Consiste em uma nova plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux, com diversas aplicações já instaladas e, ainda, com um ambiente próprio de desenvolvimento (LECHETA, 2009).

Hoje é fundamental que os profissionais de várias áreas acompanhem a evolução da tecnologia, como também as transformações do mercado. Para os representantes de vendas isto não é diferente, a utilização de soluções de força de vendas contemplando dispositivos móveis já não é mais um diferencial, é praticamente fundamental para manter o mesmo nível de competitividade entre vendedores.

Para um representante de venda da marca Sold Out, que fabrica e vende roupas voltadas para praticantes de artes marciais, podem-se perceber algumas dificuldades que o vendedor enfrenta no controle e venda dos produtos. Para auxiliar no controle das vendas e estoque, como também na demonstração dos produtos aos clientes, uma alternativa viável é o desenvolvimento de um aplicativo móvel. Com isso o vendedor teria facilidade na demonstração dos produtos através de um *showroom* digital, e um controle fácil e rápido das vendas e dos produtos em estoque.

1.1 OBJETIVOS DO TRABALHO

O objetivo geral do trabalho é facilitar as operações de vendas, controle de estoque e demonstração de produtos da marca Sold Out.

Os objetivos específicos do trabalho proposto são:

a) desenvolver um aplicativo na plataforma Android que possibilite a realização de cadastros, vendas e controle de estoque de produtos;
b) disponibilizar um *web service* para integração de informações entre o dispositivo móvel e o servidor.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo contém a introdução ao tema principal deste trabalho com a apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre Força de Vendas, Dispositivos Móveis, Android, Eclipse e ADT, *Web Services*, Sistema atual e trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do sistema, iniciando com o levantamento de informações, tendo na sequência a especificação, implementação e por fim resultados e discussões.

No quarto capítulo tem-se as conclusões deste trabalho bem como apresentam-se sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda assuntos para compreensão do trabalho, tais como conceito de Força de Vendas, Dispositivos Móveis, Android, Eclipse e Android *Development Tools* (ADT), *Web Services*, Sistema Atual além de Trabalhos Correlatos.

2.1 FORÇA DE VENDAS

No mundo dos negócios, é fundamental que as organizações criem diferenciais para se destacarem no mercado. Algumas empresas adotam o conceito de força de vendas para criar este diferencial. Segundo Paulilo (2014), força de vendas é um time de pessoas capacitadas e apaixonadas por fazer o que precisa ser feito para que a venda aconteça. Os integrantes da força de vendas realizam venda pessoal, que se refere à comunicação pessoal de informações para persuadir um *prospect* (cliente em potencial) a comprar algo, que pode ser um bem, um serviço ou uma ideia que atenda suas necessidades individuais (FUTRELL, 2003).

Para criar um diferencial ainda maior, as empresas estão investindo em tecnologia, criando assim uma dupla de sucesso, força de vendas aliada a tecnologia. Segundo Peppers e Rogers (2004, p. 77), a utilização da tecnologia de sistemas de informação para automatização de algumas etapas no processo de vendas pode forçar a disciplina e a adesão aos processos definidos pela empresa.

Embora a automação da força de vendas não seja um conceito novo, o que tem despertado interesse neste assunto são as poderosas ferramentas de informação e comunicação cada vez mais acessíveis, como notebooks mais leves e com maior capacidade de armazenamento, modems de alta velocidade, redes de dados por tecnologia celular com conexão sem fio e softwares que permitem o fácil compartilhamento de informações (FUTRELL, 2003).

A tecnologia permite que os profissionais tenham acesso a quase todo tipo de informações ou dado concebível sobre o produto que estão vendendo, possibilitando a melhoria do desempenho individual de vendas e de serviço. Um dos objetivos da utilização de tecnologia em vendas é ajudar o profissional de vendas a aumentar a velocidade com que consegue encontrar e qualificar seus prováveis clientes potenciais (FUTRELL, 2003). Com

uma equipe de vendas utilizando um sistema moderno e eficiente, a imagem da empresa tende a melhorar perante os clientes. Segundo Goldberg (2005) a imagem da empresa é imediatamente associada ao desempenho e ao comportamento das equipes de vendas.

Um sistema de automação de força de vendas deve contemplar algumas funcionalidades básicas, como o controle de pedidos, a configuração de produtos, a gerência de contatos, o planejamento de visitas e a produtividade pessoal (PEPPERS; ROGERS, 2004).

Com a utilização de um sistema de automação de vendas é possível ter alguns ganhos no processo da empresa, como a facilidade no acesso de informações, melhoria do desempenho individual de vendas e de serviço, maior controle das informações de vendas, expansão do alcance de venda, entre outras.

2.2 DISPOSITIVOS MÓVEIS

Com o passar dos anos, a evolução da computação fez com que os computadores diminuíssem de tamanho e se tornassem fáceis de utilizar em qualquer lugar. Hoje, é possível acessar qualquer dado a partir de dispositivos que vivem conosco por todos os lugares, os dispositivos móveis. A computação móvel permite que usuários tenham acesso a serviços independentemente de sua localização. Isso requer suporte à mobilidade e existência de infraestrutura de comunicações sem fio (JOHNSON, 2007).

Um dispositivo móvel deve ter algumas características básicas como possuir um grau de mobilidade, um tamanho reduzido, realizar processamento, trocar informações com alguma rede, independência de rede fixa (POSSER, 2006).

As características de um dispositivo móvel são basicamente:

a) tamanho reduzido;
b) baixo consumo de energia;
c) memória e processamento limitados;
d) conectividade limitada;
e) monitoramento do nível de energia para prevenção de perda de dados.

Com o avanço da tecnologia o conceito e principalmente as características dos dispositivos móveis sofreram algumas mudanças. Os primeiros dispositivos móveis foram os *palmtops*, muito utilizados no mundo corporativo, abriu as portas para o desenvolvimento de

sistemas em dispositivos de tamanho reduzido. Foram amplamente utilizados por vendedores, representantes, técnicos externos entre outros profissionais que atuavam fora da empresa.

O primeiro dispositivo móvel de grande sucesso foi o Newton, lançado pela Apple em 1992. Na Figura 1, pode-se visualizar o aparelho.

Figura 1 - Newton lançado em 1992

Fonte: França (2011).

O celular é outro tipo de dispositivo móvel muito utilizado. O primeiro celular foi criado em 1973 pela Motorola. Uma década depois, foi criado também pela Motorola o primeiro celular a ser comercializado (RENATO, 2014). Desde então, o crescimento acelerado do mercado de celulares, a concorrência se tornou cada vez maior entre as fabricantes e o investimento em tecnologia aumentou exponencialmente na ultima década. O conceito de celular que inicialmente era um dispositivo de telefonia móvel foi sendo alterado com o passar dos anos, hoje em dia os usuários estão procurando *smartphones*, celulares com diversos recursos como câmeras, músicas, *bluetooth*, boa interface visual, jogos, GPS, acesso a internet e TV digital (LECHETA, 2009).

2.3 ANDROID

A partir do crescimento da fabricação dos dispositivos móveis, a tendência é de que surjam sistemas operacionais potentes para acompanhar o crescimento do mercado. A partir desta evolução foi desenvolvido pela *Open Handset Aliance*, a plataforma Android.

Considerada uma plataforma móvel completa, aberta e livre para desenvolvimento (ANDROID DEVELOPERS, 2004).

Segundo Android Developers (2004), Android é um pacote de softwares para dispositivos móveis que inclui um sistema operacional, um *middleware* e aplicativos de características essenciais, como um emulador e um depurador, e toda a biblioteca de desenvolvimento. Através do Android *Software Development Kit* (SDK) é possível criar novas aplicações para a plataforma. A linguagem utilizada para programação das aplicações é o Java (ANDROID DEVELOPERS, 2004).

O Android é a aposta da Google para atender a grande necessidade dos usuários de dispositivos móveis. Consiste em uma plataforma de desenvolvimento para aplicativos móveis, bastante poderosa, ousada e flexível que foi baseada em um sistema operacional Linux (LECHETA, 2009).

A plataforma Android trouxe uma nova experiência ao usuário de celulares *smartphones*, com uma interface rica em detalhes e uma flexibilidade jamais vista até então. Sua arquitetura é muito flexível, você pode integrar aplicações nativas com aplicações desenvolvidas por você mesmo, e até mesmo substituir uma aplicação nativa por outra de desenvolvimento próprio. Não existe diferença na aplicação nativa e a aplicação de um desenvolvedor independente (LECHETA, 2009).

A flexibilidade na integração ou substituição de aplicativos se deve ao fato da plataforma Android ser completamente livre e de código aberto (*open-source*), fazendo com que cada vez mais desenvolvedores criem conteúdo para o desenvolvimento e melhoria da plataforma (LECHETA, 2009). O Android fornece ao usuário uma plataforma de classe mundial para a criação de aplicativos, bem como um mercado aberto para distribuir instantaneamente (ANDROID DEVELOPERS, 2004).

Todos os benefícios que o Android possui, fez com que se tornasse popular rapidamente. Hoje, existem centenas de milhões de dispositivos móveis em mais de 190 países ao redor do mundo utilizando a plataforma.

Segundo Android Developers (2004), mais de um milhão de novos dispositivos com Android são ativados por dia em todo o mundo. São realizados por mês mais de 1,5 bilhões de *downloads* através do Google Play, loja de virtual da Google, onde você pode fazer baixar aplicativos, filmes, jogos e até livros digitalizados.

2.4 ECLIPSE E ADT

O Eclipse é o ambiente de desenvolvimento preferido pela Google. Existe um *plugin* chamado Android Development Tools (ADT) que visa facilitar o desenvolvimento, os testes e a compilação de projetos de software para dispositivos móveis (LECHETA, 2009).

A linguagem utilizada é o Java. Usada para criar desde programas simples a páginas *web*, Java pode ser encontrada hoje em dia nos seguintes lugares: a) servidores *Web*; b) banco de dados relacionais; c) computadores *mainframe*; d) telefones; e) telescópios orbitais; f) assistentes digitais pessoais; entre outros (CADENHEAD; LEMAY, 2005).

Utilizando o *plugin* ADT é possível executar o emulador do Android diretamente do Eclipse. Para distribuição do aplicativo Android, é necessário compilar o projeto Java em um arquivo de extensão *Android Package File* (APK), que é um arquivo binário compactado com as classes compiladas e os recursos do aplicativo (LECHETA, 2009).

O ADT *plugin* amplia os recursos do Eclipse facilitando a criação de novos projetos para Android. Com ele é possível criar interface de usuário de aplicativos, adicionar pacotes com base no quadro API Android, depurar aplicativos usando as ferramentas do SDK do Android, como também distribuir a aplicação em formato .apk. O ADT também fornece editores XML personalizados que permitem editar arquivos XML específicos do Android em uma interface de usuário baseada em formulário (ANDROID DEVELOPERS, 2004).

2.5 WEB SERVICES

Webs services foram criados para construir aplicações que são serviços na internet. Porém não faz parte do conceito de *web service* a criação de interfaces gráficas para os usuários. É comum encontrar textos afirmando que *web services* disponibilizam serviços somente para desenvolvedores, ou que *web services* nada mais são do que chamada de métodos usando XML (PAMPLONA, 2010).

A plataforma básica de *web service* é *Extensible Markup Language* (XML) mais *Hyper Text Transfer Protocol* (HTTP). O XML fornece uma linguagem que pode ser usada entre diferentes plataformas e linguagens de programação e ainda interpretar mensagens e funções complexas. O protocolo HTTP é o protocolo de internet mais utilizado (W3SCHOOLS, 2013).

Com a utilização do *web service* é possível integrar duas ou mais aplicações entre si, mesmo que não sejam desenvolvidas na mesma linguagem. Os dados são convertidos para uma linguagem universal que ambas as aplicações entendem, possibilitando a troca de informações. Na Figura 2 tem-se um esquema do funcionamento do *web service*.

Figura 2 - Funcionamento de um *Web Service*

Fonte: Toolsweb (2011).

Para realizar a comunicação com o *web service* é necessário uma implementação do protocolo *Simple Object Access Protocol* (SOAP) definido no *World Wide Web Consortium* (W3C). Este protocolo é o responsável pela independência que o *web service* precisa. Atualmente encontram-se várias implementações disponíveis em várias linguagens (PAMPLONA, 2010).

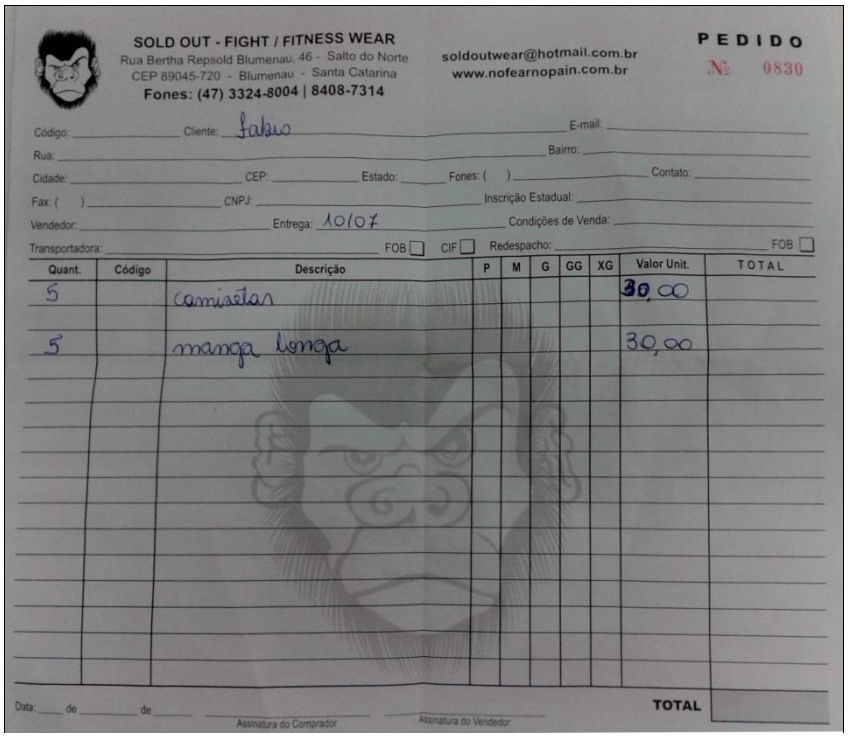
2.6 SISTEMA ATUAL

A empresa Sold Out, localizada em Blumenau estado de Santa Catarina. Foi fundada há cerca de 10 anos pelo empresário Atirson de Oliveira que atua no segmento de *fight wear* (roupas de lutas e artes marciais). A empresa oferece produtos de vestuários, tais como regatas, bermudas, casacos de moletom, jaquetas e mochilas, cujo diferencial são as estampas voltadas para diversos tipos de lutas como, por exemplo, *Muay Thai*, Judô, *Jiu Jitsu* e até o esporte do momento *Mixed Martial Arts* (MMA).

A distribuição dos produtos é realizada em academias, loja de suplementos, em outros estabelecimentos do ramo e ainda existem representantes que vendem os produtos de porta em porta.

Os representantes recebem uma determinada quantidade de produtos para vendas em consignação. No ato da retirada dos produtos é acordada uma data de devolução e os valores de cada linha de produto. Neste momento é gerado um documento, chamado de pedido, com a informação da quantidade o do preço de cada linha produto. Este pedido possui duas vias, uma via é de responsabilidade da empresa e a outra do representante. Na Figura 3, pode ser visto um exemplo de pedido entregue para os representantes.

Figura 3 - Documento entregue ao representante



SOLD OUT - FIGHT / FITNESS WEAR
Rua Bertha Repsold Blumenau, 46 - Salto do Norte
CEP 89045-720 - Blumenau - Santa Catarina
Fones: (47) 3324-8004 | 8408-7314
soldoutwear@hotmail.com.br
www.nofearnopain.com.br

PEDIDO
Nº 0830

Código: _____ Cliente: fabio _____ E-mail: _____
Rua: _____ Bairro: _____
Cidade: _____ CEP: _____ Estado: _____ Fones: ( ) _____ Contato: _____
Fax: ( ) _____ CNPJ: _____ Inscrição Estadual: _____
Vendedor: _____ Entrega: 10/07 Condições de Venda: _____
Transportadora: _____ FOB ☐ CIF ☐ Redespacho: _____ FOB ☐

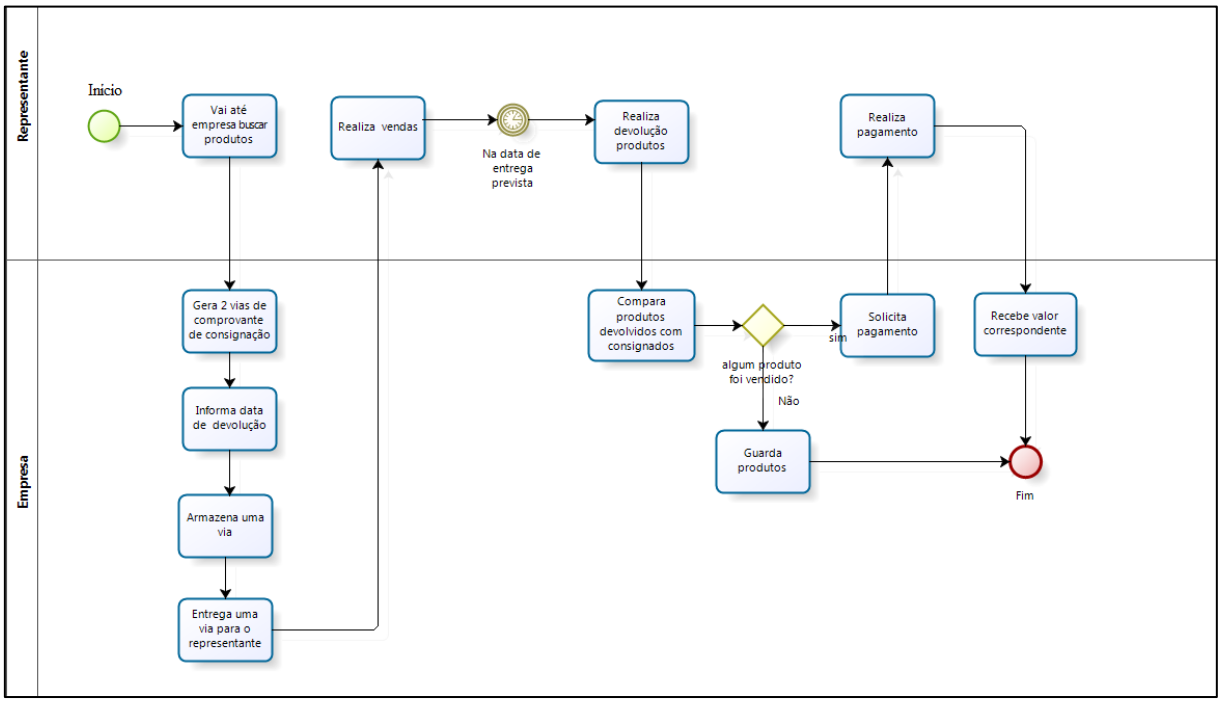
Quant.	Código	Descrição	P	M	G	GG	XG	Valor Unit.	TOTAL
5		camisetas						30,00	
5		manga longa						30,00	

Data: ___ de _____ de _____ Assinatura do Comprador _____ Assinatura do Vendedor TOTAL

Fonte: Sol Out (2014).

Após a retirada dos produtos podem ocorrer duas situações distintas. O representante vende os produtos, ficando com sua parte do valor da venda e paga para a empresa o valor previamente acordado. Ou caso o produto não seja vendido, o representante devolve a mercadoria para a empresa na data de entrega estipulada. Na Figura 4, pode ser visto o funcionamento do processo de consignação realizado pela empresa.

Figura 4 - Processo de consignação



A empresa não interfere no valor final da venda, apenas no valor que o representante deve retornar para a empresa. O percentual de lucro que o representante tem com a venda dos produtos é definido por ele mesmo, sem nenhum critério definido pela empresa.

2.7 TRABALHOS CORRELATOS

Pode-se citar como trabalhos correlatos o Trabalho de Conclusão de Curso (TCC) do aluno Tiago Machado na Universidade Regional de Blumenau (FURB) e o aplicativo para Android chamado AppVenda$.

O trabalho de Tiago Machado tem como objetivo o desenvolvimento de uma aplicação para controle de força de vendas para dispositivos móveis, baseado em PALM OS e um gerador (parte da comunicação entre o Ponto de Venda e o sistema corporativo da empresa) que irá rodar no *desktop*. O intuito é que a aplicação seja genérica, possibilitando a utilização para uma vasta gama de empresas. A aplicação foi desenvolvida em *Visual Basic* no ambiente de desenvolvimento Ns Basic (MACHADO, 2007).

As funcionalidades que tornam o trabalho de Machado (2007) e este trabalho correlatos se referem à automação da força de vendas. Ambos voltados para dispositivos

móveis e realizando integração com um servidor, permitindo ganho na velocidade do processo de troca de informações e maior eficiência na operação de vendas.

Existem alguns aspectos que diferenciam os trabalhos, como o fato do trabalho de Machado ser voltando para PALM OS e este trabalho ser aplicado para Android. Outra diferença é que a aplicação voltada para PALM OS tem o intuito de ser utilizada por diversas empresas, já este trabalho tem como objetivo atender especificamente a empresa Sold Out. Na Figura 5, pode ser visto uma tela de cadastro de clientes no emulador de PALM OS.

Figura 5 - Cadastro de clientes para Palm Os

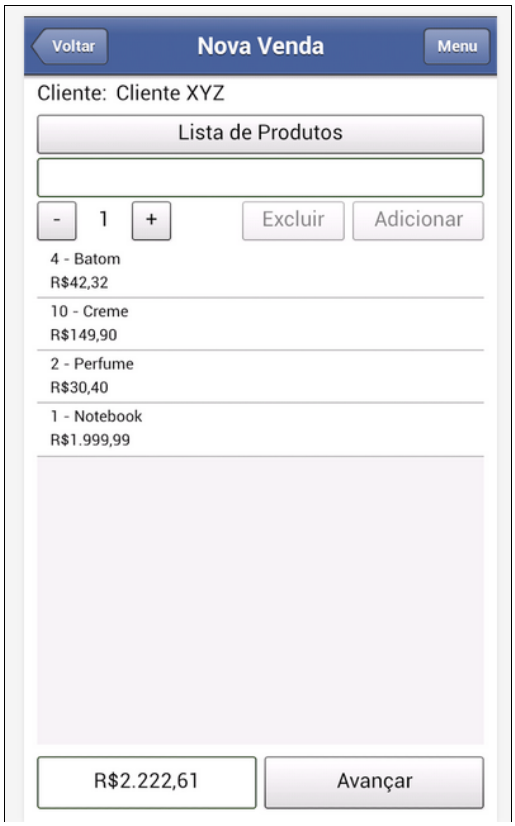
Fonte: Machado (2007).

O aplicativo AppVenda$, desenvolvido pela empresa Comunicar TI é um aplicativo simples e intuitivo com o objetivo de controlar vendas diretas. O aplicativo simplifica, organiza e controla as vendas diretas, facilitando a centralização das informações. Possui diversas funcionalidades, como: a) cadastro de clientes; b) cadastro de produtos; c) controle de vendas; d) controle de pagamentos; e) relatório de vendas; entre outros (GOOGLE PLAY, 2014).

As funcionalidades em comum entre o aplicativo e este trabalho se referem aos cadastros de clientes, produtos, controle de vendas, bem como a plataforma que ambos funcionam, o Android. O AppVenda$ possui mais funcionalidades que este trabalho, como geração de relatórios, *backup* e restauração de dados. É um aplicativo gratuito e pode ser utilizado por qualquer pessoa e/ou empresa, diferentemente deste trabalho, que possui como

foco atender a empresa Sold Out. (GOOGLE PLAY, 2014). Na Figura 6, pode ser vista a tela de realização de vendas.

Figura 6 - Tela de vendas do AppVenda$



Fonte: Google Play (2014).

3 DESENVOLVIMENTO

Neste capítulo estão descritos os aspectos técnicos utilizados no desenvolvimento do sistema, bem como o levantamento de informações: requisitos funcionais, não funcionais e regras de negócio. Além dos diagramas utilizados no desenvolvimento: diagrama de casos de uso, diagrama de atividades e modelo de entidade e relacionamento. Estão descritas também as técnicas e ferramentas utilizadas, a operacionalidade da implementação e os resultados obtidos.

3.1 LEVANTAMENTO DE INFORMAÇÕES

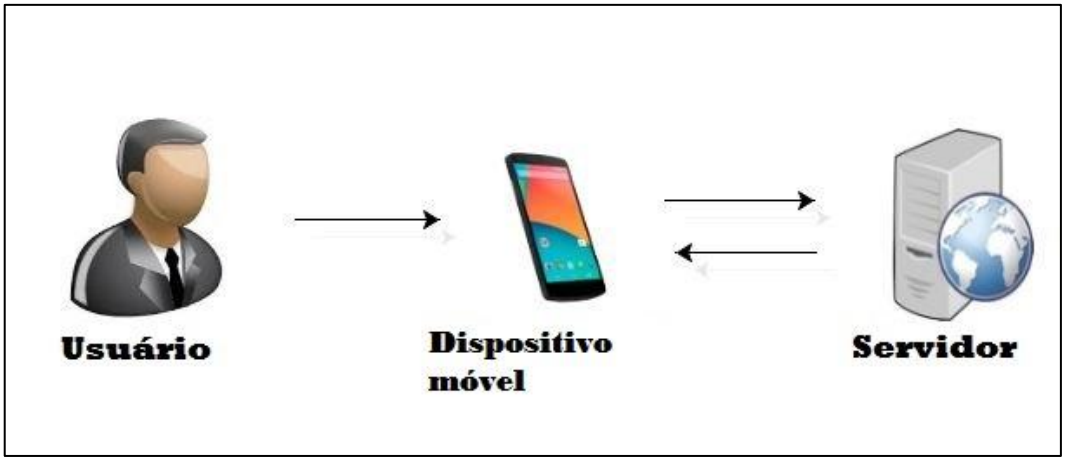
A partir de entrevistas informais com representantes e o proprietário da empresa e conforme pode ser visto na seção do sistema atual, notou-se que os maiores problemas no funcionamento da empresa são relacionados com o processo de consignação:

a) risco com representantes: ao realizar o processo de consignação a empresa acaba confiando nos representantes, correndo o risco de não receber o valor das mercadorias do representante, bem como o representante perder as mercadorias;
b) avarias nos produtos: os produtos entregues aos representantes podem sofrer avarias, já que estão sujeitos a provas pelas pessoas interessadas;
c) venda incerta: não se sabe os produtos disponibilizados aos representantes serão vendidos, e desta forma, a empresa corre o risco de perder vendas;
d) falta de controle de entrada e saída: não existe nenhum tipo de controle dos produtos disponibilizados aos representantes, além do pedido feito em papel no dia da retirada dos produtos, gerando incerteza sobre quantidade de produtos com cada representante;
e) falta de controle de estoque: devido à incerteza de quantos produtos serão vendidos, não há como controlar o estoque de forma eficaz;
f) preços indefinidos: como o preço final dos produtos é definido pelo representante, podem ocorrer preços abusivos, sujando a reputação da empresa.

Para atender as necessidades descritas, pensou-se em uma solução que modifica diversos processos da empresa. Uma das alterações é para que não seja mais necessária a

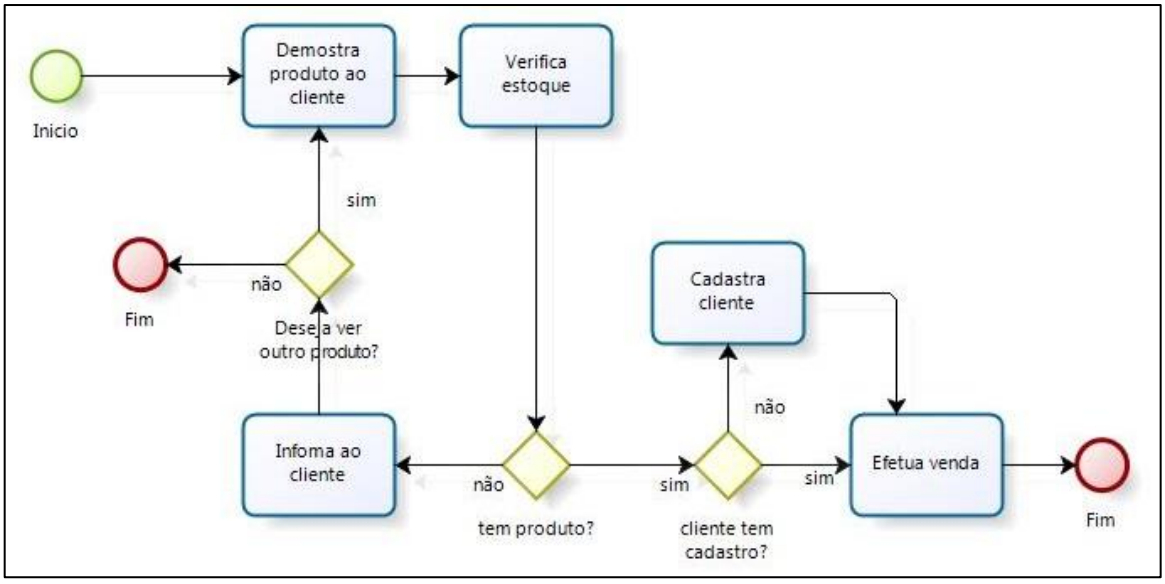
realização de consignação de produtos. A solução desenvolvida para a empresa contempla um aplicativo para dispositivo móvel, onde o representante e o proprietário poderão interagir para realizar cadastros, demonstração de produtos bem como realizar vendas. Essa aplicação terá integração com o servidor onde os dados serão armazenados. Na Figura 7, pode ser visto o funcionamento da solução.

Figura 7 – Funcionamento da solução



Para não haver mais consignação de produtos, os representantes utilizarão o aplicativo para realizar a demonstração dos produtos e em seguida, caso o cliente possua interesse em comprar o produto, a venda é realizada no mesmo aplicativo. O processo é simples e prático, conforme pode ser visto no fluxo de procedimento de vendas na Figura 8.

Figura 8 - Venda de produtos

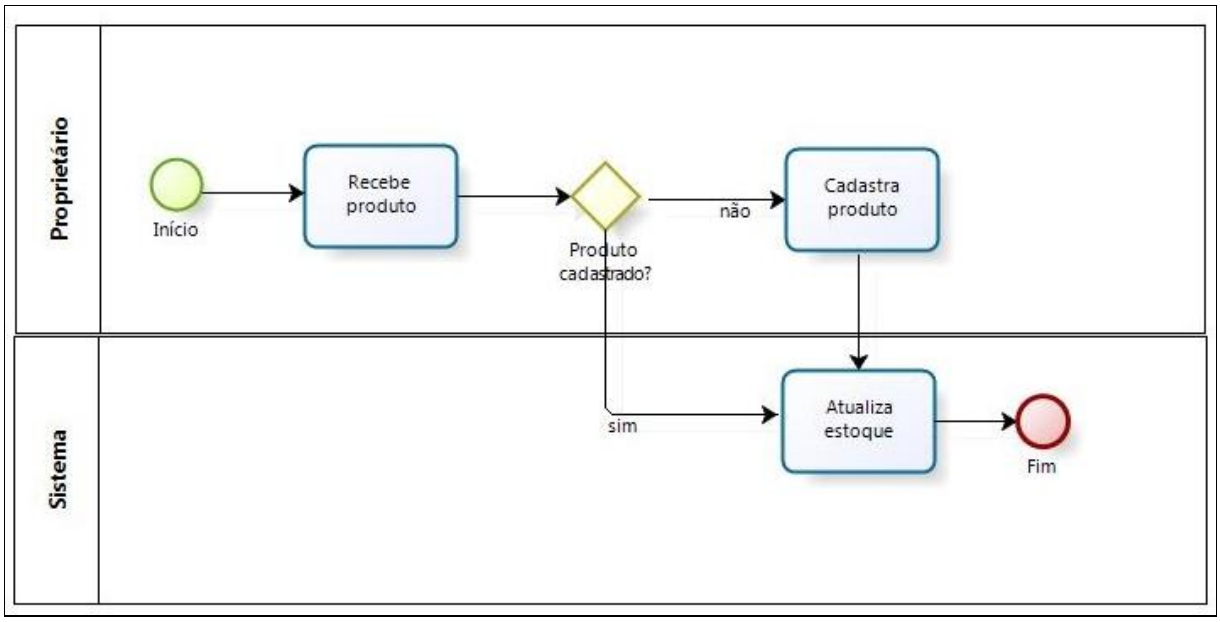


Sem o processo de consignação, a entrega do produto ao cliente não será no momento da venda. Com o novo processo a entrega passa a ser realizada via correio, já que a empresa tem no cadastro do cliente o seu endereço. Como as vendas são atualizadas diretamente com o servidor, o funcionário responsável pela separação e postagem dos produtos tem a informação bem atualizada, podendo agilizar a entrega, garantindo a satisfação do cliente.

Por não haver mais consignação de produtos, a empresa poderá realizar também o controle de seu estoque, fazendo com que haja mais controle sobre a produção. Isto porque a partir da realização do controle de estoques, será possível saber se há demanda para determinado produto e se deve ser confeccionado ou não. O controle de estoques foi desenvolvido neste trabalho para facilitar ainda mais o dia-a-dia.

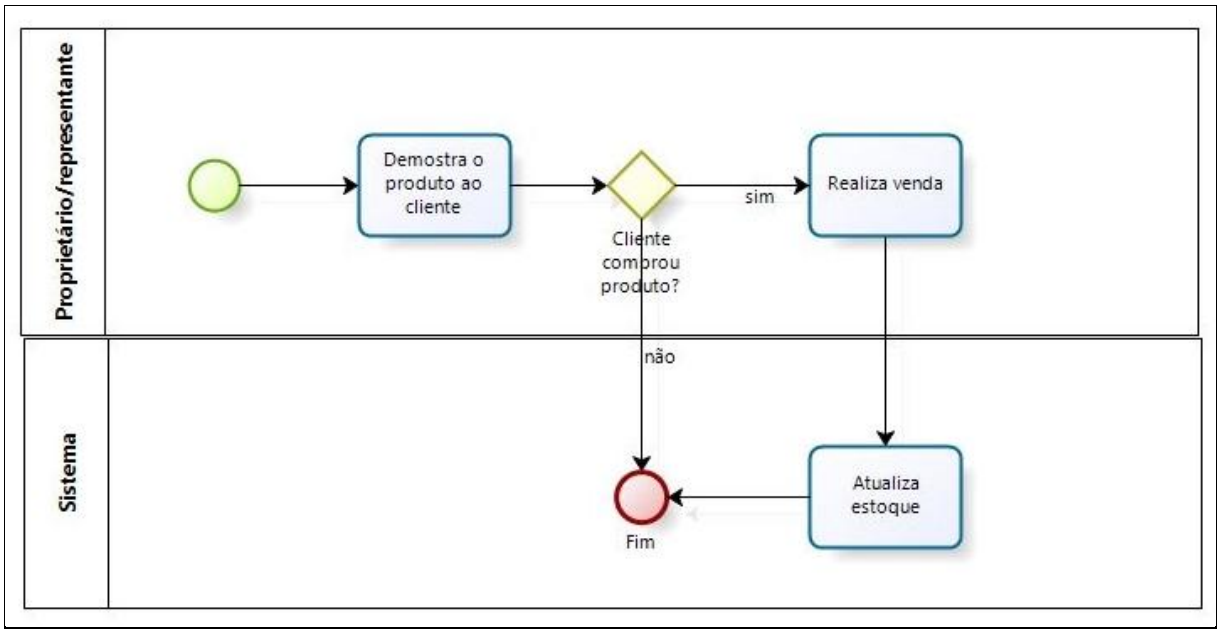
Sempre que forem recebidos produtos, o sistema irá acrescentar no estoque os produtos recebidos, fazendo com que as informações fiquem sempre atualizadas. Na Figura 9 pode ser visto o fluxo do processo de recebimento de produtos.

Figura 9 - Recebimento de produto



Ao realizar a venda de produtos também ocorrerá o mesmo processo, só que neste caso, haverá diminuição dos produtos vendidos do estoque. O fluxo do processo de atualização do estoque no processo de vendas pode ser visto na Figura 10.

Figura 10 - Atualização do estoque na venda de produtos



Optou-se por realizar o desenvolvimento de forma que o banco de dados não fique no próprio dispositivo móvel, isto porque poderá haver várias pessoas interagindo ao mesmo tempo com o sistema. Desta forma, haverá o aplicativo e o servidor. Serão utilizadas várias ferramentas para proporcionar o desenvolvimento da solução, sendo elas:

a) Eclipse para implementação do código fonte;

b) PostgreSql para armazenamento dos dados;

c) Sparx Systems Enterprise Architect para modelagem de diagramas.

3.2 ESPECIFICAÇÃO

A seguir é apresentada a especificação do sistema, contemplando os requisitos funcionais, requisitos não funcionais, além dos diagramas de casos de uso e diagrama de classes. Para criar o diagrama de caso de uso e o diagrama de classes foi utilizada a ferramenta Enterprise Architect (EA).

3.2.1 REQUISITOS DO SISTEMA

Nesta subseção são apresentadas as principais características do sistema. Com base na seção 2.4 e em entrevistas com os principais *stakeholders* do projeto, foram elaborados os requisitos para atender as principais necessidades encontradas atualmente.

No Quadro 1 são apresentados os requisitos funcionais do sistema relacionados com seus respectivos casos de uso.

Quadro 1 - Requisitos funcionais

Requisitos Funcionais	**Caso de Uso**
RF01: O sistema deverá permitir ao proprietário cadastrar produtos.	UC01
RF02: O sistema deverá permitir ao usuário manter clientes.	UC02
RF03: O sistema deverá permitir ao proprietário controlar estoque.	UC03
RF04: O sistema deverá possibilitar ao usuário efetuar a venda de produtos.	UC04
RF05: O sistema deverá permitir ao representante consultar o estoque.	UC05

No Quadro 2 são apresentados os requisitos não funcionais utilizados no desenvolvimento do sistema.

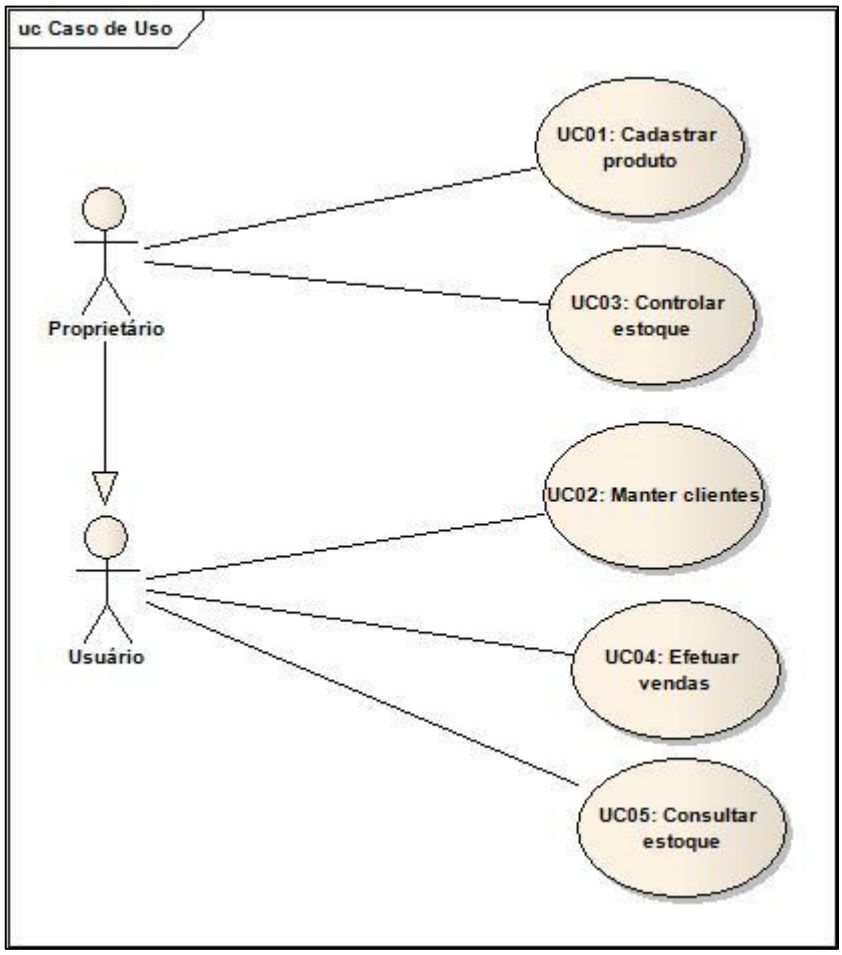
Quadro 2 – Requisitos não funcionais

Requisitos Não Funcionais
RNF01: O sistema deverá utilizar ambiente de desenvolvimento Eclipse com *plugin* ADT.
RFN02: O sistema deverá ser compatível com a plataforma Android.
RNF03: O sistema deverá utilizar banco de dados PostgreSql.
RFN04: O sistema deverá permitir a integração de dados entre aplicativo e servidor via *web service*.
RFN05: O sistema deverá utilizar como servidor de aplicação Glassfish 4.
RFN06: O sistema necessita de conexão com internet para seu funcionamento.

3.2.2 DIAGRAMA DE CASO DE USO

Na Figura 11 é apresentado o diagrama dos casos de uso protótipo de aplicativo. Os casos de usos de UC01 e UC02 são acionados somente pelo proprietário via aplicativo. Os demais casos de uso podem ser acessados por qualquer usuário, inclusive pelo proprietário. Os detalhamentos dos principais casos de uso estão disponíveis no Apêndice A.

Figura 11 - Diagrama de casos de uso



3.2.3 DIAGRAMA DE CLASSES

A seguir são demonstradas as classes utilizadas no desenvolvimento do sistema. Nas Figuras 12 e 13 são apresentados os diagramas de classes do servidor. Nas Figuras 14 e 15 são apresentados os diagramas de classes do aplicativo. O dicionário de dados está descrito no Apêndice B.

Figura 12 - Diagrama de classes

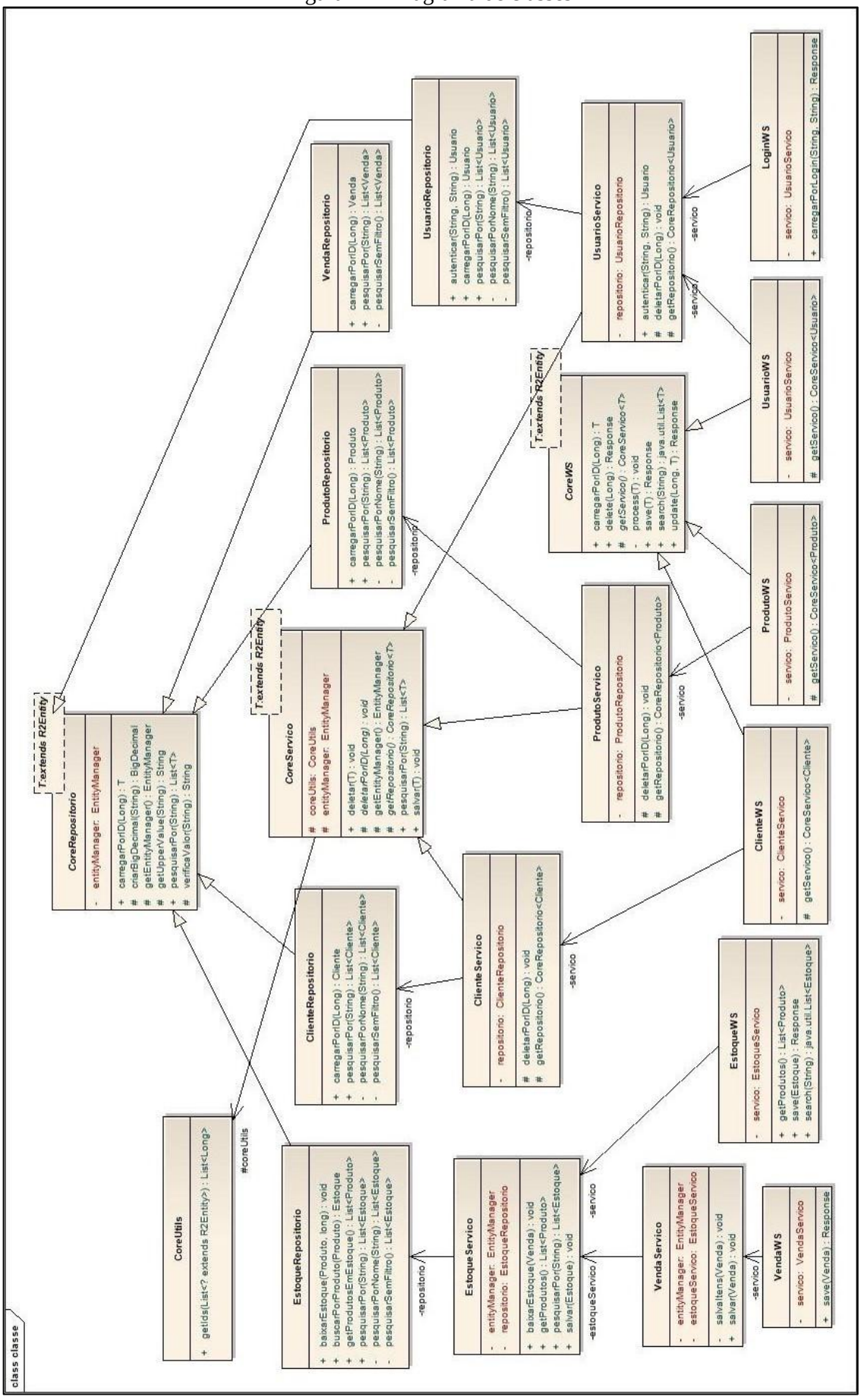
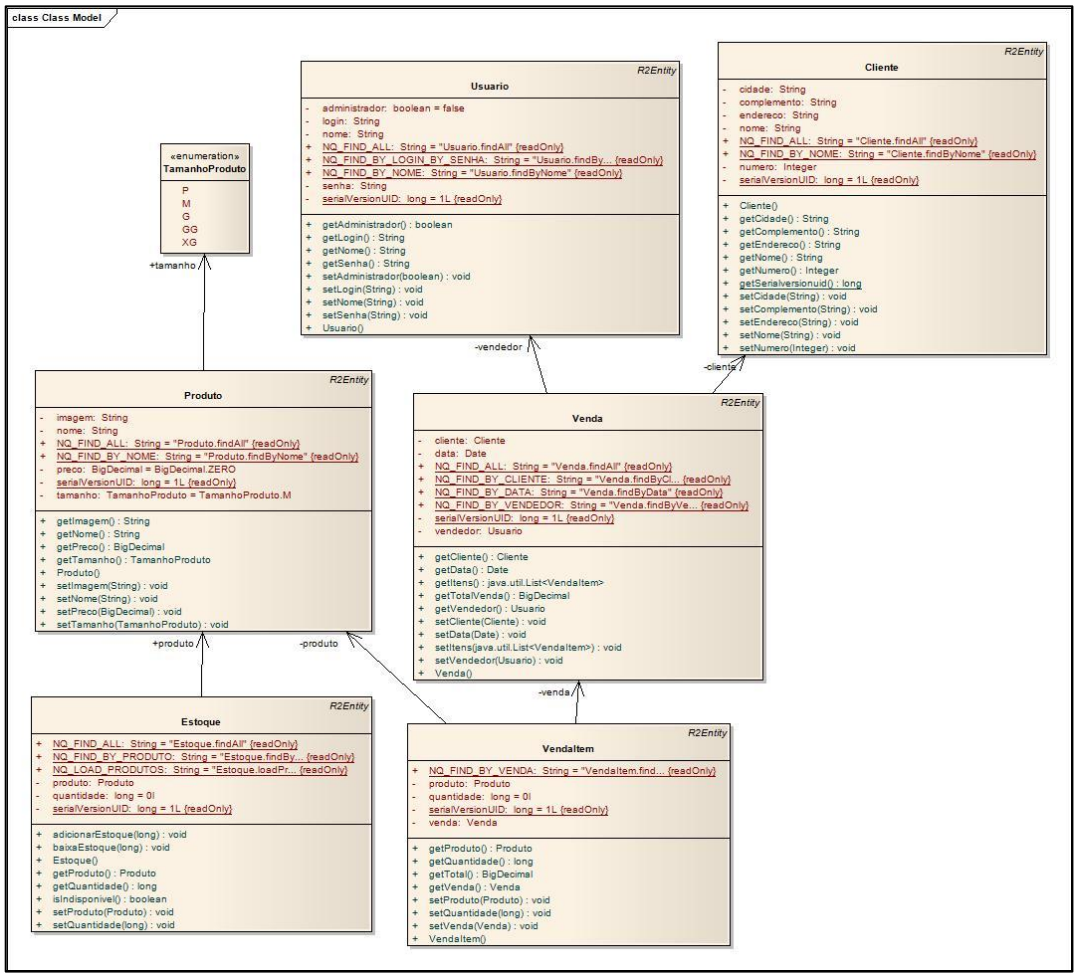


Figura 13 - Classes do servidor



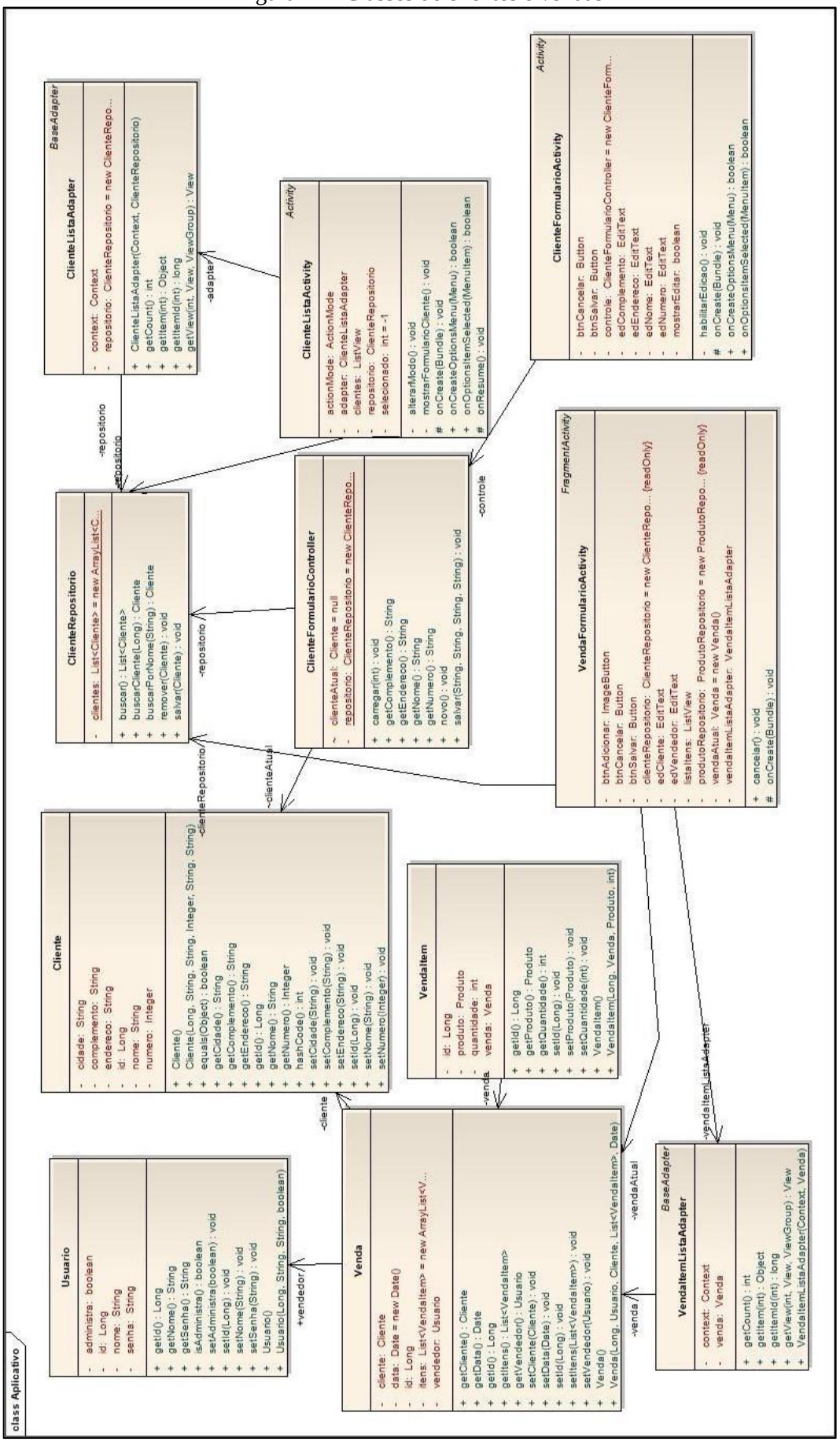
A seguir é apresentada uma breve descrição das classes criadas para o servidor:

a) Cliente: classe *Plain Old Java Object* (POJO) que representa um cliente no sistema;
b) ClienteRepositorio: responsável por realizar pesquisas que dizem respeito a Clientes;
c) ClienteServico: classe que contém a regra de negócio para persistência de clientes;
d) ClienteWS: classe que representa o *web service* de clientes;
e) CoreRepositorio: superclasse que contém as funcionalidades básicas para todas as outras classes de repositório;
f) CoreServico: superclasse que contém as funcionalidades básicas para todas as outras classes de serviços;
g) CoreUtils: classe utilitária;
h) CoreWS: superclasse de todas as classes que representam um *web service*;
i) Estoque: classe POJO que representa um estoque no sistema;

j) EstoqueRepositorio: classe que representa o repositório de estoques;
k) EstoqueServico: classe que contém a regra de negócio para persistência de estoques;
l) EstoqueWS: representa o *web service* de estoques;
m) Produto: classe POJO que representa um produto no sistema;
n) ProdutoRepositorio: classe que representa o repositório de produtos;
o) ProdutoServico: classe que contém a regra de negócio para persistência de produtos;
p) ProdutoWS: representa o *web service* de produtos;
q) TamanhoProduto: *enum* que representa os tamanhos dos produtos;
r) Usuario: classe POJO que representa um usuário no sistema;
s) UsuarioRepositorio: classe que representa o repositório de estoques;
t) UsuarioServico: classe que contém a regra de negócio para persistência de usuários;
u) UsuarioWS: representa o *web service* de usuário;
v) Utils: classe utilitária com métodos para conversões;
w) VendaRepositorio: classe que representa o repositório de vendas, responsável por realizar pesquisas que dizem respeito a Venda;
x) VendaServico: classe que contém a regra de negócio para persistência de vendas;
y) VendaWS: representa o *web service* de vendas;
z) VendaItem: Classe POJO que representa um item da venda no sistema.

A seguir, são apresentados os diagramas das classes utilizadas pelo aplicativo. Na Figura 14 é apresentado o diagrama das classes de vendas e clientes do aplicativo. Na Figura 15 é apresentado o diagrama das classes de estoque e produtos.

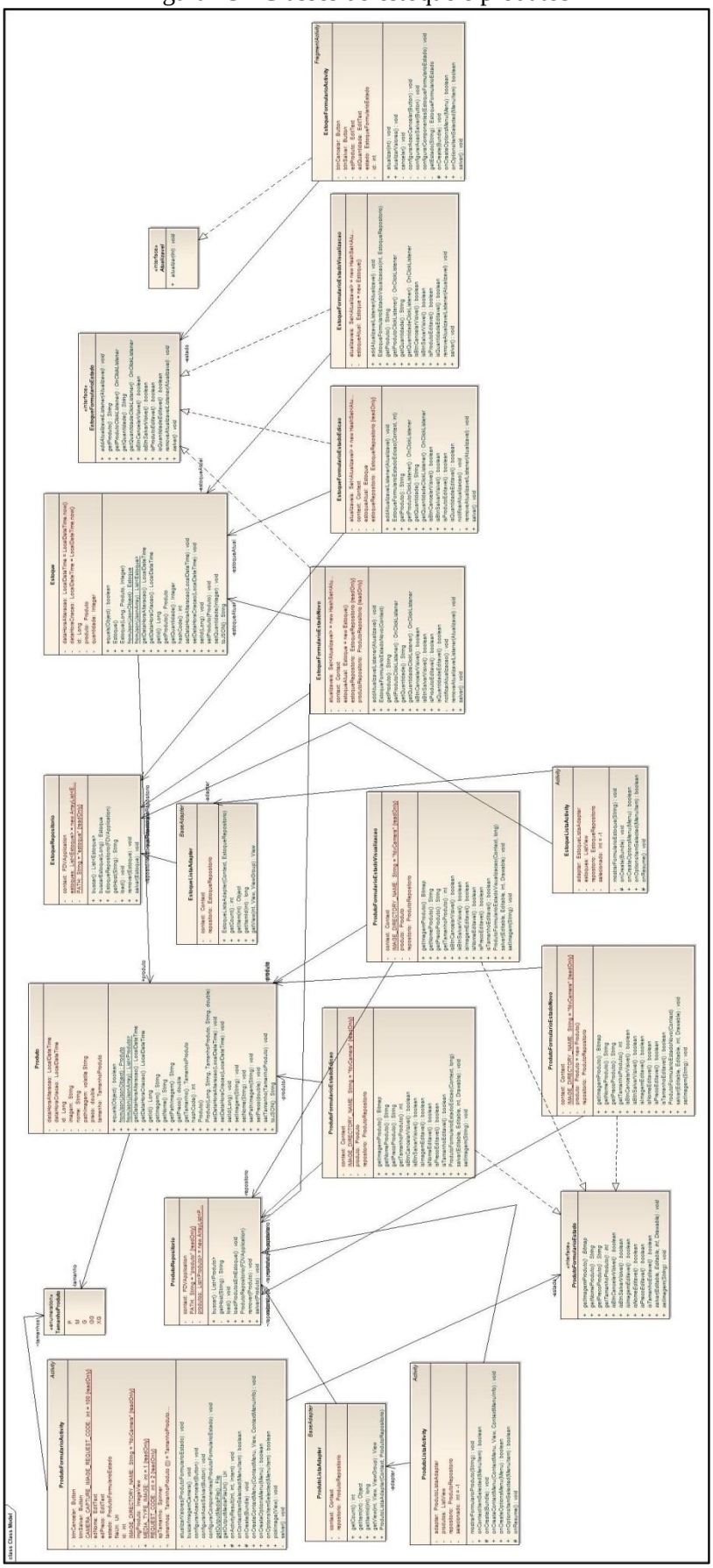
Figura 14 - Classes de clientes e vendas



A seguir é apresentada uma breve descrição das classes de cliente e vendas do aplicativo:

a) Cliente: classe que representa o cliente;
b) ClienteFormularioController: controlador do formulário de clientes;
c) ClienteFormularioActivity: classe com os métodos do formulário de clientes;
d) ClienteListaActivity: classe que apresenta a listagem de clientes cadastrados;
e) ClienteListaAdapter: classe que carrega os clientes que serão apresentados no formulário de clientes;
f) ClienteRepositorio: classe utilitária responsável por gerenciar a carga dos dados de clientes;
g) Venda: classe que representa uma venda no aplicativo;
h) VendaFormularioActivity: classe com os métodos do formulário de vendas;
i) VendaItem: classe que representa o item de venda;
j) VendaListaAdapter: classe que carrega os itens que serão apresentados no formulário de vendas;
k) Usuário: classe que representa o usuário.

Figura 15 - Classes do estoque e produtos



A seguir é apresentada uma breve descrição das classes de cliente e vendas do aplicativo:

a) Atualizavel: interface para indicação de classes que podem receber atualização externa.
b) Estoque: classe que representa o estoque;
c) EstoqueFormularioActivity: classe que controla a manipulação do formulário de estoques;
d) EstoqueFormularioEstado: interface para representação do estado do formulário de estoque;
e) EstoqueFormularioEstadoEdicao: classe que configura e controla os controles do formulário de estoque durante a edição;
f) EstoqueFormularioEstadoNovo: classe que configura e controla os controles do formulário de estoque durante a criação de um novo estoque;
g) EstoqueFormularioEstadoVisualizacao: classe que configura e controla os controles do formulário de estoque durante a visualização de estoque;
h) EstoqueListaActivity: classe que apresenta a tela de listagem de estoque;
i) EstoqueListaAdapter: controla os itens que serão apresentados na listagem de estoque;
j) EstoqueRepositorio: classe responsável por gerenciar a carga dos dados do estoque;
k) PreferenciasFormActivity: classe que representa a tela de preferencias;
l) Produto: classe que representa o produto;
m) ProdutoFormularioActivity: classe com métodos do formulário de produto;
n) ProdutoFormularioEstado: interface que representa o estado do formulário de produtos;
o) ProdutoFormularioEstadoEdicao: classe que controla os componentes do formulário de cadastro de produto durante a edição de um produto;
p) ProdutoFormularioEstadoNovo: classe que controla os componentes do formulário de cadastro de produto durante a criação de um novo produto;
q) ProdutoFormularioEstadoVisualizacao: classe que controla o estado dos componentes de formulário de produtos durante a visualização de um produto;
r) ProdutoListaActivity: classe que realiza a apresentação da listagem de produtos;
s) ProdutoListaAdapter: classe que realiza o controle dos itens que serão apresentados na lista de produtos;

t) ProdutoRepositorio: classe utilitária responsável por gerenciar a carga dos dados dos produtos;
u) TamanhoProduto: *enum* com os tamanhos de produtos.

3.3 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

A implementação do sistema foi dividida em dois projetos: servidor e aplicativo. Em cada projeto foram utilizados padrões e técnicas diferentes, porém ambos utilizando a linguagem Java e o Eclipse. A seguir, é detalhado o desenvolvimento de cada projeto bem como as técnicas utilizadas.

3.3.1.1 Servidor

No desenvolvimento do servidor, foi utilizado um *framework* para persistência dos dados, chamado de *Java Persistence API* (JPA). O JPA é um *framework* baseado em *Plain Old Java Objects* (POJOS) para persistir dados em Java que possui as seguintes funcionalidades:

a) capacidade mapeamento objeto-relacional, realizado através de anotações no código ou através de um XML definido externamente;
b) realização de consultas através do Java Persistence Query Language (JPQL), uma linguagem de consulta que é derivada do EJB QL e transformada depois para SQL;
c) integração e teste (MEDEIROS, 2014).

No desenvolvimento, o JPA foi utilizado para realização de consultas e identificação do modelo objeto-relacional, ou seja, para realizar o mapeamento dos atributos conforme o que está configurado nas entidades do bando de dados. A implementação foi realizada sem o

arquivo XML externo, portanto as declarações ficam dentro das próprias classes. Na Figura 16 pode ser vista a implementação da classe de Produto utilizando JPA.

Figura 16 - Código utilizando JPA

```
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import r2jpa.domainentity.R2Entity;

@Table(name="produtos")
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
@NamedQueries({
    @NamedQuery(name = "Produto.findAll",
                query = "SELECT o FROM Produto o ORDER BY o.nome asc"),
    @NamedQuery(name = "Produto.findByNome",
                query = "SELECT o FROM Produto o WHERE UPPER(o.nome) LIKE :nome ORDER BY o.nome asc")
})

public class Produto extends R2Entity {

    private static final long serialVersionUID = 1L;
    public final static String NQ_FIND_ALL = "Produto.findAll";
    public final static String NQ_FIND_BY_NOME = "Produto.findByNome";

    @NotNull(message = "O nome do produto é obrigatório")
    @Size(max = 100, message = "O tamanho máximo para o nome do produto é de 100 caracteres")
    @Column(name = "nome", length = 100, nullable = false)
    private String nome;

    @Column(name = "preco", precision = 11, scale = 2)
    private BigDecimal preco = BigDecimal.ZERO;

    @NotNull(message = "O tamanho do produto é obrigatório")
    @Column(name = "tamanho", nullable = false, length = 2)
    @Enumerated(EnumType.STRING)
```

No desenvolvimento pensou-se em otimizar a implementação do código no que se refere às validações. Para isso, foi utilizado o *Beans Validation* que é uma especificação Java que permite conforme Bernard (2014):

a) expressar restrições sobre modelos de objetos através de anotações;
b) escrever restrições personalizadas de forma extensível;
c) fornecer as APIs para validar objetos e gráficos de objetos;
d) fornecer as APIs para validar os parâmetros e valores de retorno de métodos e construtores;
e) relatar o conjunto de violações (localizado) executado em Java SE, integrado no Java EE (6 e 7).

Na Figura 17, pode ser visto a classe de Cliente utilizando *Beans Validation* para validações do que será informado ao realizar um cadastro de cliente.

Figura 17 - Validações com *Beans Validation*

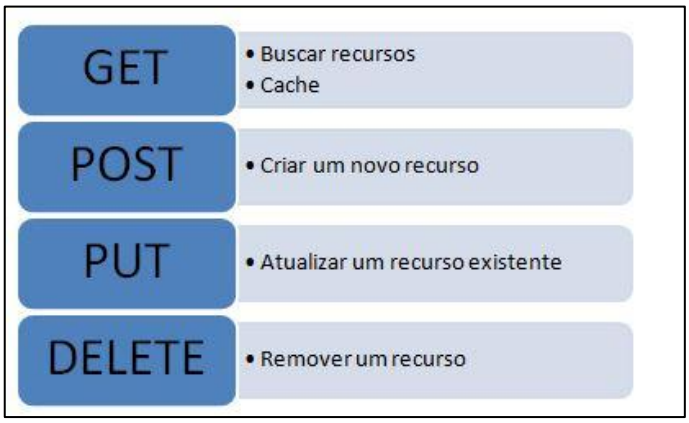
```
public class Cliente extends R2Entity {

    private static final long serialVersionUID = 1L;
    public final static String NQ_FIND_ALL = "Cliente.findAll";
    public final static String NQ_FIND_BY_NOME = "Cliente.findByNome";

    @NotNull(message = "O nome do cliente é obrigatório")
    @Size(max = 100, message = "O tamanho máximo para o nome do cliente é de 100 caracteres")
    @Column(name = "nome", length = 100, nullable = false)
    private String nome;

    @Size(max = 50, message = "O tamanho máximo para o endereço é de 50 caracteres")
    @Column(name = "endereco", length = 50)
    private String endereco;

    @Column(name = "numero")
    private Integer numero;

    @Size(max = 30, message = "O tamanho máximo para o complemento do endereço é de 30 caracteres")
    @Column(name = "complemento", length = 30)
    private String complemento;
```

Para realizar a integração entre o servidor e o aplicativo, optou-se pela implementação de *web services*. Os *web services* utilizam o estilo arquitetural conhecido como *Representational State Transfer* (REST). Os serviços que seguem este estilo arquitetural são conhecidos como serviços RESTful e se baseiam no funcionamento da web e do protocolo HTTP. Este funcionamento utiliza a estrutura de recursos (GODINHO, 2009).

O funcionamento dos *web services* é baseado no mesmo princípio da *web* de página e devem ser baseados em um protocolo universal, que para web pode-se considerar o HTTP, um protocolo aberto e amplamente conhecido. Os serviços devem também expor recursos que possam ser associados através de URIs, característica que faz parte das raízes do funcionamento da web. As chamadas às URIs são realizadas através de verbos HTTP (GODINHO, 2009).

Na Figura 18 pode ser visto os verbos HTTP e suas utilizações. Ambos utilizados no desenvolvimento dos *web services*.

Figura 18 - Verbos HTTP



No projeto de servidor, foi criada uma classe genérica que implementa os métodos HTTP em comum para todos os *web services*, chamada de CoreWS. Na Figura 19 pode ser vista a classe CoreWS.java.

Figura 19 - Classe que métodos HTTP

```
public abstract class CoreWS<T extends R2Entity> {

    protected abstract CoreServico<T> getServico();

    @GET
    public java.util.List<T> search(@DefaultValue("") @QueryParam("filtro") String filtro){
         return  getServico().pesquisarPor(filtro);
    }

    @GET
    @Path("/{id}")
    @Produces(value = MediaType.APPLICATION_JSON)
    public T carregarPorID(@PathParam("id") Long id) {
        return null;
    }

    @POST
    public void save(T entity) {
        process(entity);
    }

    @POST
    @Path("/{id}")
    public void update(@PathParam("id") Long id, T entity) {
        process(entity);
    }

    private void process(T entity) {
        getServico().salvar(entity);
    }

    @DELETE
    @Consumes(value={MediaType.TEXT_PLAIN, MediaType.APPLICATION_XML})
    @Path("/{id}")
    public void delete(@PathParam("id") Long id) {
        getServico().deletarPorID(id);
    }
}
```

A classe CoreWS é herdada por classes filhas que implementam os *web services* e tem

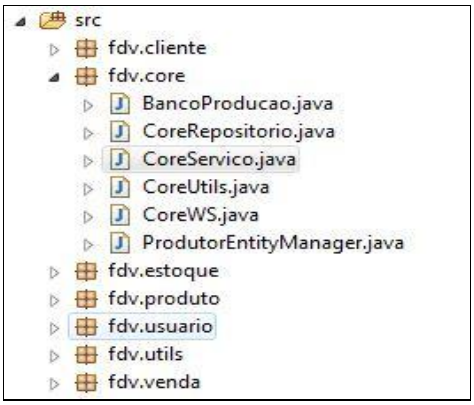
seus métodos sobrescritos quando necessário. Um exemplo disto é a classe que implementa o serviço de vendas, que pode ser vista na Figura 20.

Figura 20 - Implementação do serviço de vendas

```
import javax.inject.Inject;
import javax.ws.rs.Consumes;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import fdv.core.CoreServico;
import fdv.core.CoreWS;
import fdv.estoque.ProdutoSemEstoqueException;
import fdv.estoque.QuantidadeInsuficienteEstoqueException;

@Path("/venda")
@Produces(value = MediaType.APPLICATION_JSON)
@Consumes(value = MediaType.APPLICATION_JSON)
public class VendaWS extends CoreWS<Venda> {

    @Inject
    private VendaServico servico;

    @Override
    protected CoreServico<Venda> getServico() {
        return servico;
    }

    @Override
    public void save(Venda entity) {
        try {
            servico.efetivaVenda(entity);
        } catch (QuantidadeInsuficienteEstoqueException
                | ProdutoSemEstoqueException e) {
            e.printStackTrace();
        }
    }
}
```

No desenvolvimento do projeto foi realizada a criação de classes abstratas genéricas que implementam métodos em comum e que podem ser sobrescritos quando necessário. O pacote “Core” possui todas as classes genéricas que são utilizadas nas demais classes da implementação do lado servidor. Na Figura 21 pode ser visto o pacote Core com as classes que compõem o pacote.

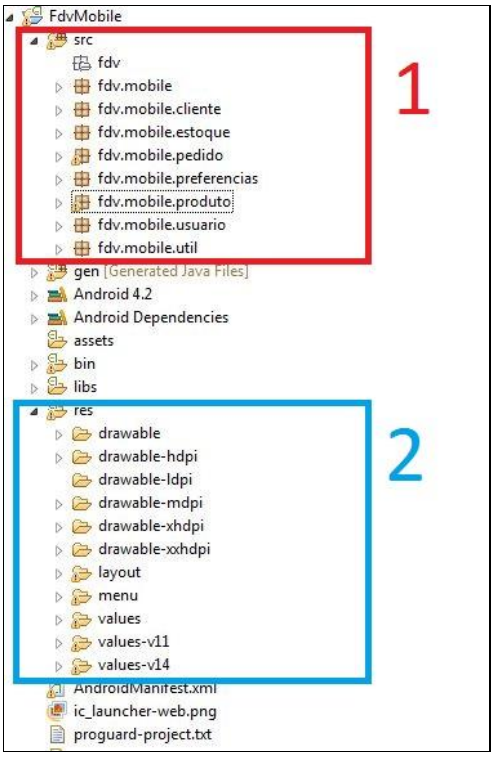
Figura 21 - Pacote "Core" - classes genéricas



3.3.1.2 Aplicativo

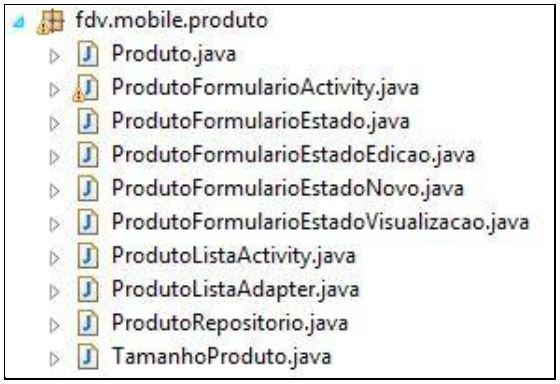
O desenvolvimento do aplicativo foi realizado em um projeto separado do desenvolvimento do servidor. Este projeto utiliza o *plugin* ADT, que conforme mencionado na seção 2.4, facilita o desenvolvimento de aplicativos para Android. O projeto do aplicativo está subdividido conforme pode ser visto na Figura 22.

Figura 22 - Projeto Aplicativo



O item 1 da Figura 21 representa os pacotes com classes. A divisão dos pacotes foi definida conforme funcionalidades, por exemplo, o pacote “fdv.mobile.produto” trata das funcionalidades do cadastro de produto, conforme pode ser visto na Figura 23.

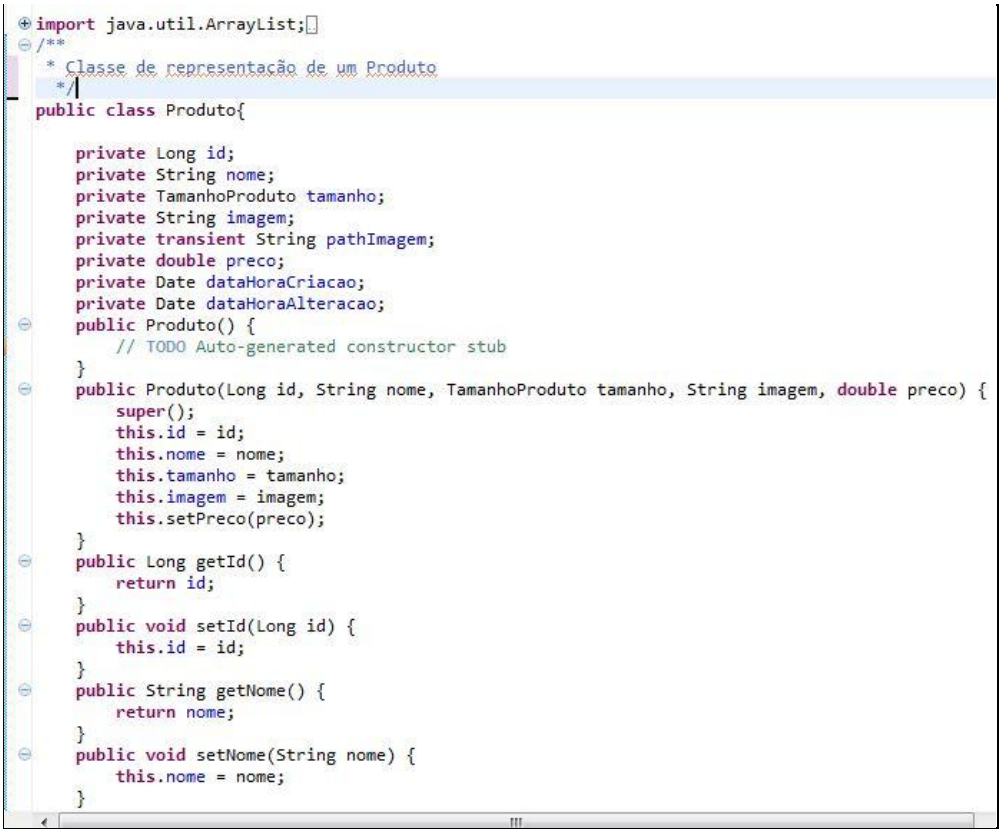
Figura 23 - Pacote produto



Todos os pacotes possuem basicamente a mesma estrutura: possuem uma classe que representa o objeto. Possui classe *Activity*, classes de listas, classes de repositório e neste pacote em específico possuem também classes para controle de estado. A seguir será detalhado cada tipo de classe existente nos pacotes.

As classes que representam os objetos contêm em si atributos e métodos de cada objeto, que serão utilizados posteriormente ao obter ou preencher valores na tela. Na Figura 24 pode ser vista a classe que representa o objeto Produto.

Figura 24 - Classe de produto



```
import java.util.ArrayList;
/**
 * Classe de representação de um Produto
 */
public class Produto{

    private Long id;
    private String nome;
    private TamanhoProduto tamanho;
    private String imagem;
    private transient String pathImagem;
    private double preco;
    private Date dataHoraCriacao;
    private Date dataHoraAlteracao;
    public Produto() {
        // TODO Auto-generated constructor stub
    }
    public Produto(Long id, String nome, TamanhoProduto tamanho, String imagem, double preco) {
        super();
        this.id = id;
        this.nome = nome;
        this.tamanho = tamanho;
        this.imagem = imagem;
        this.setPreco(preco);
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
```

A classe chamada de *Activity*, segundo Amorim (2013), é responsável por definir a tela, controlar sua máquina de estados, passagem de parâmetro de uma tela para outra, entre outros. Esta classe contém basicamente os métodos acionados a partir da tela, bem como seus parâmetros, conforme pode ser visto na Figura 25.

Figura 25 - Classe ProdutoActivity

```
public class ProdutoFormularioActivity extends Activity {
    TamanhoProduto[] tamanhos = TamanhoProduto.values();
    public static final int MEDIA_TYPE_IMAGE = 1;
    private static final int CAMERA_CAPTURE_IMAGE_REQUEST_CODE = 100;
    private static final String IMAGE_DIRECTORY_NAME = "fdvCamera";
    private static final int REQUEST_CODE = 2;
    private Spinner spTamanho;
    private EditText edNome;
    private EditText edPreco;
    private ImageView imgProduto;
    private Button btnCancelar;
    private Button btnSalvar;
    private ProdutoFormularioEstado estado;
    private int id;
    private Uri fileUri;

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

        ArrayAdapter<TamanhoProduto> adapter = new ArrayAdapter(this,
                android.R.layout.simple_spinner_item, tamanhos);

        spTamanho = (Spinner) findViewById(R.id.produtoFormSpTamanhoProduto);
        spTamanho.setAdapter(adapter);
        edNome = (EditText) findViewById(R.id.produtoFormEdNomeProduto);
        edPreco = (EditText) findViewById(R.id.produtoFormEdPrecoProduto);
        imgProduto = (ImageView) findViewById(R.id.produtoFormImgProduto);
        btnCancelar = (Button) findViewById(R.id.produtoFormBtnCancelar);
        btnSalvar = (Button) findViewById(R.id.produtoFormBtnSalvar);

        Intent intent = getIntent();
        String estado = intent.getStringExtra("estado");
        id = intent.getIntExtra("id", 0);
        if ("VISUALIZAR".equals(estado)) {
            this.estado = new ProdutoFormularioEstadoVisualizacao(
                    (Context) this, id);
        } else if ("EDITAR".equals(estado)) {
```

As classes de lista, chamadas de Lista *Adapter*, realizam o controle dos itens que serão apresentados na tela.

As classes de repositório são responsáveis por gerenciar a carga dos dados. Representa a camada de abstração entre a aplicação e o servidor. São responsáveis por chamar realizar a requisição dos *web services*, via REST. Na Figura 26, pode ser vista o método “salvar()” da classe “ProdutoRepositorio.java”.

Figura 26 - Método salvar

```
public void salvar(Produto produto) {
    try {
        if(produto.getId() == null){
            Rest.post("http://192.168.0.2:8080/teste/api-fdv/produto",
                    produto.toJSON());
        }else{
            Rest.put("http://192.168.0.2:8080/teste/api-fdv/produto/"+produto.getId(),
                    produto.toJSON());
        }

        produtos.remove(produto);
        produtos.add(produto);
    } catch (Throwable e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

As classes de repositório utilizam métodos da classe “Rest.java” que é responsável por realizar a comunicação entre aplicação e servidor utilizando o padrão REST, ou seja, todos os métodos de comunicação com o servidor estão implementados nesta classe.

No método “salvar” demostrado na Figura 26, foram utilizados os métodos “post” e “put” da classe de Rest. Na Figura 27, pode ser visto a implementação do método “post”, que conforme definido na seção 3.3.1.1, é responsável por criar um novo recurso.

Figura 27 - Método "post" da classe Rest.java

```
public static String post(String url, String json) throws Throwable {
    HttpParams httpParameters = new BasicHttpParams();

    int timeoutConnection = 6000;
    HttpConnectionParams.setConnectionTimeout(httpParameters,
            timeoutConnection);

    int timeoutSocket = 6500;
    HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);
    DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);
    HttpContext localContext = new BasicHttpContext();
    HttpPost postRequest = new HttpPost(url);

    StringEntity input = new StringEntity(json);
    input.setContentType("application/json");
    postRequest.setEntity(input);

    HttpResponse response = httpClient.execute(postRequest, localContext);

    if (response.getStatusLine().getStatusCode() != 201) {
        throw new RuntimeException("Falha : HTTP cod : "
                + response.getStatusLine().getStatusCode());
    }

    BufferedReader br = new BufferedReader(new InputStreamReader(
            (response.getEntity().getContent())));

    String output;
    StringBuilder resultado = new StringBuilder();
    while ((output = br.readLine()) != null) {
        resultado.append(output);
    }

    httpClient.getConnectionManager().shutdown();
    return resultado.toString();
}
```

No pacote de produto existe ainda a interface de estado da tela, ou seja, representa se os campos devem ficar ou não habilitados, durante a criação ou edição do cadastro de produtos. Na Figura 28, pode ser vista a interface “ProdutoFormularioEstado.java” que é implementada pelas classes “ProdutoFormularioEstadoEdicao.java” e “ProdutoFormularioEstadoNovo.java” para controle da tela de produtos.

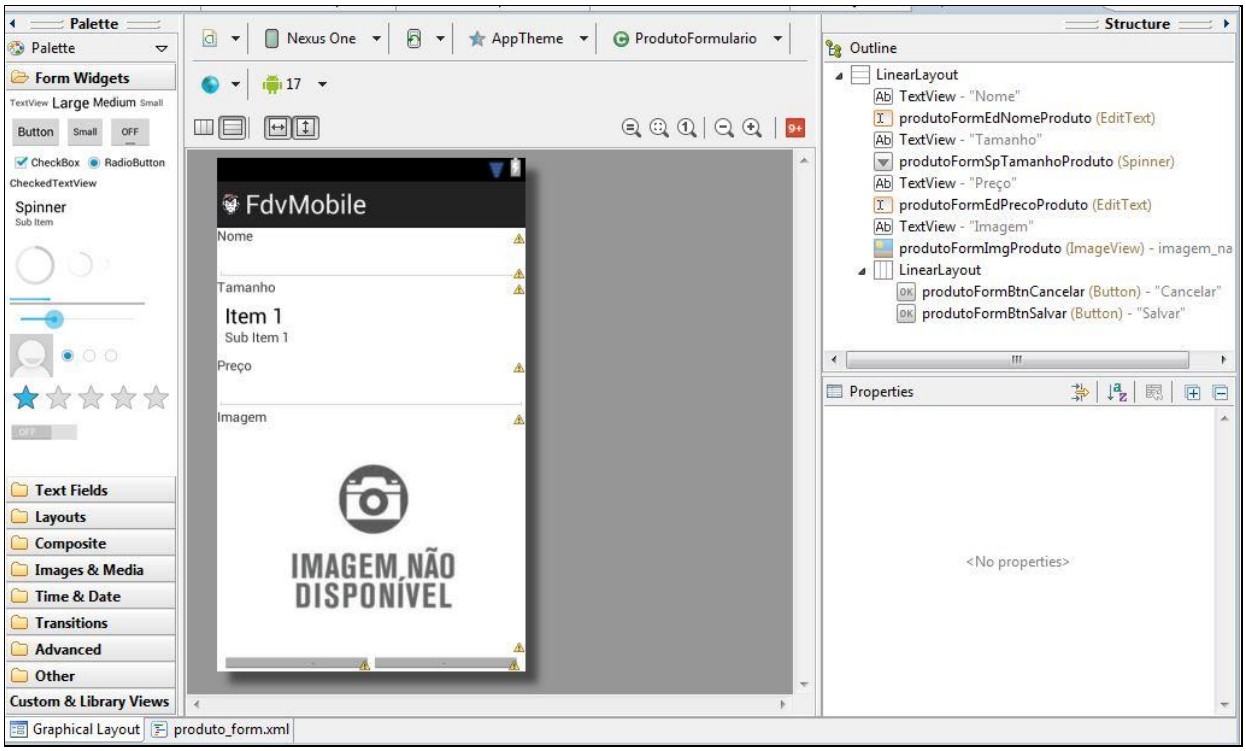
Figura 28 - Interface de controle de estados do produto

```
import android.graphics.Bitmap;

/**
 * Representação do Estado da tela do Cadastro de Produto
 *
 */
public interface ProdutoFormularioEstado {

    boolean isTamanhoEditavel();

    boolean isNomeEditavel();

    boolean isPrecoEditavel();

    boolean isImagemEditavel();

    boolean isBtnCancelarVisivel();

    boolean isBtnSalvarVisivel();

    void salvar(Editable text, Editable text2, int selectedItemPosition,
            Drawable drawable);

    String getNomeProduto();

    String getPrecoProduto();

    int getTamanhoProduto();

    Bitmap getImagemProduto();

    void setImagem(String path);

}
```

A seguir será apresentada uma breve descrição sobre o item 2 do projeto, mostrado na Figura 22. A pasta “res” possui todos os recursos do aplicativo, ou seja, é o local onde são armazenadas as imagens utilizadas nas telas, o *layout* e XML de cada tela, o *menu*, bem como a definição dos valores: cores, dimensões, entre outros.

Na Figura 29, pode ser vista a definição do *layout* da tela de produto, com os recursos utilizados na criação da tela.

Figura 29 - Layout da tela de produtos



Na Figura 30, pode ser visto o XML com as definições da tela de produto, onde são definidos os componentes utilizados, bem como o tamanho de cada item.

Figura 30 - XML da tela de produto

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Nome"/>
    <EditText
        android:id="@+id/produtoFormEdNomeProduto"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Tamanho"/>
    <Spinner
        android:id="@+id/produtoFormSpTamanhoProduto"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Preco"/>
    <EditText
        android:id="@+id/produtoFormEdPrecoProduto"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
         android:inputType="numberDecimal" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Imagem"/>
    <ImageView
        android:id="@+id/produtoFormImgProduto"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
Graphical Layout  produto_form.xml
```

3.3.2.1 Operacionalidade do aplicativo

Para realizar o acesso ao aplicativo é necessário já possuir usuário e senha previamente cadastrados. Este cadastro é realizado pelo administrador do sistema. Para realizar o acesso ao sistema, é necessário fazer *login* no sistema. Na Figura 31, pode-se visualizar a tela de *login*.

Figura 31 - Tela de *login*

Após realizar *o login,* é aberto o *menu* onde o usuário seleciona a operação que deseja realizar, conforme pode ser visto na Figura 32.

Figura 32 - Menu do aplicativo

Para cadastrar um novo produto, o usuário seleciona o item “Produto” no *menu,* então o aplicativo mostra para o usuário uma lista de produtos já cadastrados e a opção de cadastrar um novo produto, conforme pode ser visto na Figura 33.

Figura 33 - Lista de produtos

Ao selecionar um produto na lista ou clicar no botão novo, o aplicativo abre a tela de cadastro de produtos. Esta tela é utilizada toda vez que chegam novos produtos na empresa ou quando é necessário alterar algum produto já existente. Na Figura 34 pode ser vista a tela de cadastro e edição de produtos.

Figura 34 - Tela de produto

Nesta tela, ao selecionar o ícone de imagem, é possível inserir ou alterar uma imagem do produto cadastrado. O usuário pode selecionar uma imagem existente na galeria ou tirar uma nova foto, conforme pode ser visto na Figura 35.

Figura 35 - Tela para selecionar imagem

Após realizar o cadastro do produto, o usuário deve atualizar o controle de estoque desse produto. Ao entrar na tela do controle de estoque, os produtos cadastrados são automaticamente carregados para que o usuário possa selecionar o produto desejado. Na Figura 36, pode ser visualizada a seleção do produto na tela de controle de estoques.

Figura 36 - Seleção de produto no estoque

Após informar o produto, deve-se informar a quantidade de produtos e então salvar as alterações. Na Figura 37 pode ser vista a seleção da quantidade de produtos e na Figura 38 pode ser vista a tela com as informações do estoque.

Figura 37 - Seleção da quantidade de produto

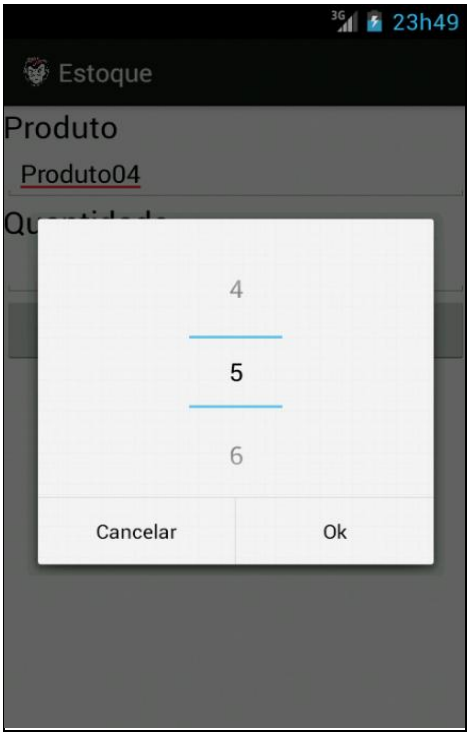
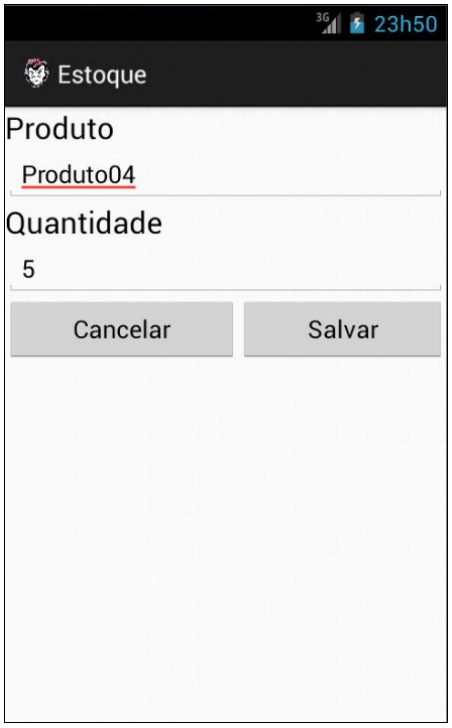
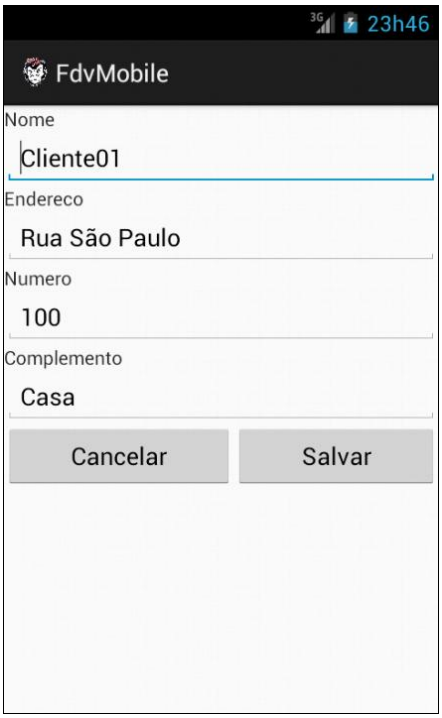


Figura 38 - Tela de estoque



Outro cadastro que é possível realizar através do aplicativo é o cadastro de clientes. Na tela de cadastro de cliente são informados alguns dados básicos utilizados a fim de ter a informação de para quem se está vendendo. Na Figura 39 pode ser vista a tela de cadastro de cliente.

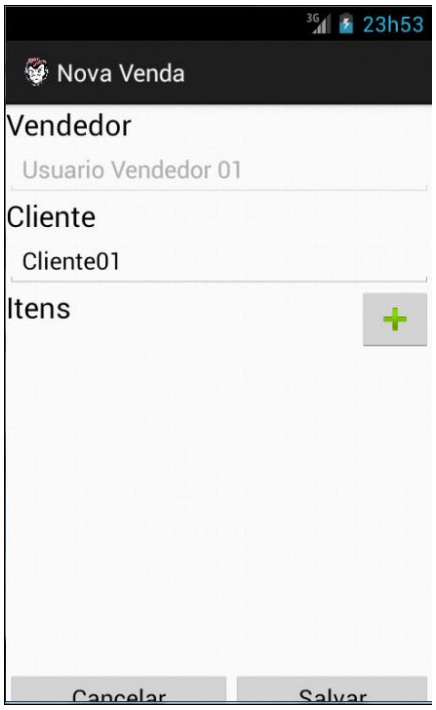
Figura 39 - Cadastro de cliente



A funcionalidade considerada mais importante do aplicativo é a realização da venda, utilizada quando o representante vender algum produto. Para realizar uma venda é necessário que todos os cadastros descritos anteriormente já tenham sido realizados.

Ao abrir a tela de venda, a informação de quem está realizando a venda vem preenchida automaticamente, com base no usuário logado, conforme pode ser visto na Figura 40.

Figura 40 - Tela de vendas

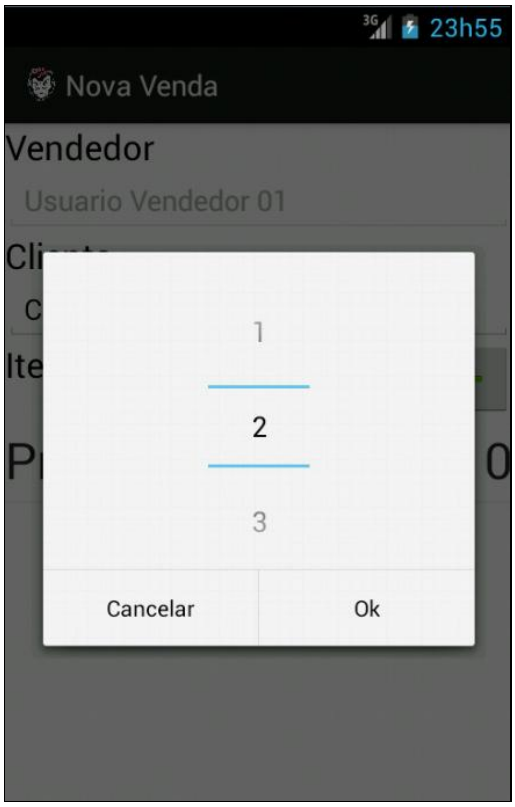


Ao realizar uma venda, o representante deve informar o cliente que está comprando, bem como os itens vendidos. O aplicativo carrega para seleção do usuário todas as informações previamente cadastradas.

Ao informar um item da venda o representante deve informar também a quantidade vendida para aquele item. Na Figura 41, pode ser vista a seleção do item vendido e na Figura 42 pode ser vista a seleção da quantidade por item.

Figura 41 - Seleção de item de venda

Figura 42 - Seleção da quantidade de item vendido



O processo de vendas através do aplicativo pode ser considerado intuitivo. A venda pode ser realizada de qualquer lugar através do uso de um dispositivo móvel, tornando o dia-a-dia do representante mais produtivo do que quando realizava a venda no papel.

Na Figura 43 pode ser vista a tela de realização de venda.

Figura 43 - Tela de realização de venda

3.4 RESULTADOS E DISCUSSÃO

Um dos principais objetivos deste trabalho era desenvolver um aplicativo para Android que possibilitasse a realização do processo de vendas de produtos, desde o cadastro de clientes até a realização da venda. Este objetivo foi atingido a partir da criação do protótipo de aplicativo para força de vendas que realiza a integração via *web service* com o servidor onde está alocado o banco de dados. Desta forma, o segundo objetivo do trabalho que era disponibilizar um *web service* para integração de informações entre o dispositivo móvel e o servidor também foi realizado. Com o desenvolvimento do aplicativo foi possível realizar comparações entre o aplicativo desenvolvido e os correlatos apresentados na seção 2.5 deste

trabalho.

Comparando o aplicativo desenvolvido neste trabalho com o aplicativo desenvolvido por Machado (2007), pode-se notar que embora os dois trabalhos sejam aplicativos para dispositivos móveis, há grandes diferenças entre os aplicativos. O sistema operacional é uma das grandes diferenças, pois o aplicativo desenvolvido por Machado (2007) roda em PALM OS, um sistema operacional desenvolvido em 1996 para Assistentes Digitais Pessoais (PDAS) e posteriormente estendido para suportar *smartphone* (MOBILEINFOCENTER, 2014). Já o aplicativo deste trabalho, foi desenvolvido para Android, o sistema operacional que domina cerca de 85% do mercado de celulares brasileiro (LANDIM, 2014). Outra diferença que se pode destacar também é o foco dos trabalhos, no trabalho de Machado o foco é utilização do aplicativo para uma vasta gama de empresas, diferentemente deste trabalho que tem como objetivo atender uma empresa específica. No Quadro 3 pode ser vista a comparação das características de cada um dos trabalhos.

Quadro 3 - Comparação com aplicativo de Machado (2007)

Características	Aplicativo Machado (2007)	Aplicativo deste trabalho
Linguagem de programação	Visual Basic	Java
Banco de dados	Utiliza arquivo de banco de dados PDB	PostgreeSql
Ambiente (*web/desktop/mobile*)	*Mobile*	*Mobile*
Sistema operacional	PALM OS	Android
Foco do trabalho	Aplicação voltada para PALM OS para ser utilizada por diversas empresas	Aplicativo para Android para atender a empresa Sold Out
Ambiente de desenvolvimento	Basic NS	Eclipse

Ao comparar o aplicativo desenvolvido com o aplicativo AppVenda$, não existe diferença entre o sistema operacional. A principal diferença é que o AppVenda$ é um aplicativo grátis que pode ser utilizado por qualquer pessoa ou empresa. Já o aplicativo desenvolvido neste trabalho tem como foco apenas uma empresa e não será comercializado para outras empresas. As principais funcionalidades em si, são basicamente iguais, no entanto o AppVenda$ tem algumas funcionalidades a mais que o aplicativo desenvolvido, sendo alguns deles: a) controle de pagamentos; b) relatório de vendas; c) Backup e Restauração dos Dados. No Quadro 4 pode ser vista a comparação entre os aplicativos.

Quadro 4 - Comparação com aplicativo AppVenda$

Características	Aplicativo AppVenda$	Aplicativo deste trabalho
Linguagem de programação	Java	Java
Ambiente (*web/desktop/mobile*)	*Mobile*	*Mobile*
Sistema operacional	Android	Android
Foco do trabalho	Aplicativo com objetivo de controlar vendas diretas	Aplicativo para Android para atender a empresa Sold Out
Ambiente de desenvolvimento	Eclipse	Eclipse

4 CONCLUSÕES

Este trabalho apresenta um protótipo de aplicativo para força de vendas que será utilizado pela empresa Sold Out. A grande vantagem é que o aplicativo permite que seja alterado o processo de vendas. Atualmente na empresa as vendas são realizadas utilizando o conceito de consignação, ou seja, os representantes pegam os produtos na empresa e vendem conforme demanda. A partir da utilização do aplicativo, as vendas não serão mais realizadas com consignação. O representante irá utilizar o aplicativo para realizar todo o processo de vendas. No próprio aplicativo o representante irá realizar a demonstração do produto e em seguida a realização da venda. Outra vantagem da utilização do aplicativo é que o estoque ficará sempre atualizado, o que não acontecia no processo anterior de consignação.

Além do aplicativo, foi desenvolvida uma integração entre o aplicativo e o servidor. A comunicação entre aplicativo e servidor é realizada via *web service*. O aplicativo utilizado por cada representante se comunica com o servidor, onde todos os procedimentos realizados são armazenados ou atualizados no banco de dados.

Para o desenvolvimento foi escolhida a linguagem Java tanto para o desenvolvimento do aplicativo quanto do servidor. Optou-se por utilizar a mesma linguagem para o desenvolvimento das duas camadas, para que todos utilizem o mesmo ambiente de desenvolvimento, além das tecnologias utilizadas serem *Open Source* e rodarem em diversas plataformas diferentes.

Para o aplicativo, foi escolhido o sistema operacional Android. A escolha foi baseada no mercado atual, já que o Android predomina a maior parte do mercado de dispositivos móveis no Brasil. Além disto, os usuários já possuem dispositivos com Android e estão familiarizados com o sistema operacional.

O protótipo de aplicativo possui algumas limitações, ou seja, algumas funcionalidades não previstas nesse aplicativo, mas que futuramente poderão ser implementadas. Dentre elas, pode-se destacar a criação de uma tela para visualização das vendas realizadas, geração de relatórios e criação de controle de metas por representante.

Embora existam algumas limitações no protótipo, pode-se considerar que os resultados obtidos no desenvolvimento deste trabalho são satisfatórios, visto que os objetivos principais foram atingidos e principalmente que através da utilização do aplicativo, o dia-a-dia da empresa Sold Out se tornará mais prático e haverá mais controle sobre os processos de venda e estoque da empresa.

4.1 EXTENSÕES

Para trabalhos futuros, surge a oportunidade de complementar a solução agregando novas funcionalidades aos sistemas. Como sugestão, pode-se implementar uma integração do cadastro de usuário com o sistema de Recursos Humanos (RH), para que o *login* dos representantes seja único para a empresa toda.

Outra sugestão seria expandir o sistema para que ele se torne mais completo e possa ser utilizado em diversos departamentos da empresa, por exemplo, criar um controle de produção, Ordem de Produção (OP) entre outras funcionalidades.

O sistema poderia também ser complementado com um módulo de relatórios e gráficos, para facilitar a gestão de vendas da empresa, bem como o acompanhamento de resultados.

Seria interessante também, disponibilizar o aplicativo para outros sistemas operacionais, como IOS e Windows Phone.

REFERÊNCIAS

ANDROID DEVELOPERS. **Home page**, [Califórnia], 2004. Disponível em: <http://developer.android.com/index.html>. Acesso em: 27 ago. 2013.

AMORIM, Vicente**. Computação Móvel**: Conceitos Básicos do Android. [Ouro Preto], 2013. Disponível em: <http://www.decom.ufop.br/vicente/disciplinas/2013_2/comp_movel/material/conceitos_basicos_android.pdf>. Acesso em: 22 jun. 2014.

BERNARD, Emmanuel. **Bean Validation:** What is Bean Validation [S.l], 2014. Disponível em: <http://beanvalidation.org/>. Acesso em: 08 jun. 2014.

CADENHEAD, Rogers; LEMAY, Laura. **Aprenda em 21 dias Java 2**. Rio de Janeiro: Elsevier, 2005.

FRANÇA, Renato. **Mercado do Mundo de Dispositivos Móveis**. [Recife], 2011. Disponível em: <https://www.google.com/url?q=http://www.cin.ufpe.br/~pet/wordpress/wp-content/uploads/2011/09/Mercado-do-Mundo-de-Dispositivos-M%25C3%25B3veis1.pptx&sa=U&ei=66I3UqiBAsfP2wWx-YGwCw&ved=0CAcQFjAA&client=internal-uds-cse&usg=AFQjCNE9MqSXIaY7jnhGyTnohvX7jTPaew>. Acesso em: 16 set. 2013.

FUTRELL, Charles M. **Vendas:** Fundamentos e novas práticas de gestão. São Paulo: Saraiva, 2003.

GODINHO, Rafael. **Criando serviços REST com WCF**. [S.l], 2009. Disponível em: <http://msdn.microsoft.com/pt-br/library/dd941696.aspx#item2>. Acesso em: 08 jun. 2014.

GOLDBERG, Claudio. **A estratégia e objetivo da força de vendas**. [São Paulo], 2005. Disponível em: <http://www.institutomvc.com.br/costacurta/artCG03Estrategias_Objetivos.htm>. Acesso em: 25 ago. 2013.

GOOGLE PLAY. **AppVenda$ - Controle de Vendas**. [S.l], 2014. Disponível em: <https://play.google.com/store/apps/details?id=air.AppVendas&hl=pt_BR>. Acesso em: 10 maio. 2014.

JOHNSON, Thienne M. **JAVA para Dispositivos Móveis**: desenvolvendo aplicações com J2me. São Paulo: Novatec, 2007.

LANDIM, Wikerson. **Android domina 85,1% do mercado de celulares no Brasil.** [S.l], 2014. Disponível em: <http://www.tecmundo.com.br/celular/50306-android-domina-85-1-do-mercado-de-celulares-no-brasil.htm>. Acesso em: 20 jun. 2014.

LECHETA, Ricardo R. **Google Android**: aprenda a criar aplicações para dispositivos móveis com Android SD. São Paulo: Novatec, 2009.

MACHADO, Tiago. **Sistema gerencial para automação de força de vendas usando dispositivos móveis baseados em Palm OS**. 2007. 81 f. Trabalho de conclusão de curso (Bacharelado Ciências da Computação) – Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.inf.furb.br/~pericas/orientacoes/VendasPalm2007.pdf>. Acesso em: 18 ago. 2013.

MEDEIROS, Higor. **Introdução à JPA - Java Persistence API.** [S.l], 2014. Disponível em: <http://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173>. Acesso em: 08 jun. 2014.

MOBILEINFOCENTER. **Palm Os**. [S.l], 2014. Disponível em: <http://www.palminfocenter.com/palm-os/>. Acesso em: 20 jun. 2014.

PAMPLONA, Vitor Fernando. **Web Services. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME**. [S.l], 2010. Disponível em: <http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-disponibilizando-e-acessando-Web-Services-via-J2SE-e-J2ME.html>. Acesso em: 17 set. 2013.

PAULILO, Gustavo. **Como construir uma força de vendas:** O que faz uma força de vendas se diferenciar de simplesmente uma equipe de vendas? [S.l], 2014. Disponível em: <http://www.agendor.com.br/blog/como-construir-uma-forca-de-vendas/>. Acesso em: 31 mar. 2014

PEPPERS, D.; ROGERS, M. **CRM marketing 1 to 1**: um guia executivo para entender e implantar estratégias de customer relationship management. São Paulo: Peppers and Rogers Group do Brasil, 3. ed. 2004.

POSSER, Lucas S. **Computação móvel e mlearning**: estudo e construção de um protótipo para Smartphone. 2006. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas da Informação) – Centro Universitário Ritter dos Reis, Porto Alegre. Disponível em: <http://www.uniritter.edu.br/graduacao/informatica/sistemas/downloads/Computacao_Movel_e_M-Learning.pdf>. Acesso em: 25 ago. 2013.

RENATO, Flávio. **A história dos telefones celulares.** [S.l], 2014. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/06/historia-dos-telefones-celulares.html>. Acesso em: 30 mai. 2014.

SOLD OUT. Figth Fitness Wear. **Documento entregue ao representante**. Blumenau, 2014.

TOOLSWEB. **Home page**, [S.l], 2011. Dispovível em: <http://www.toolsweb.com.br/?webservice>. Acesso em: 25 ago. 2013.

W3SCHOOLS. **Home page**, [S.l], 2013. Disponível em: <http://www.w3schools.com/webservices/ws_intro.asp>. Acesso em: 27 ago. 2013.

APÊNDICE A – Descrição dos casos de uso

Este Apêndice apresenta a descrição dos casos de uso conforme previstos no(s) diagrama apresentado na seção 3.2.1, conforme Quadro 5.

Quadro 5 – Descrição dos casos de uso

UC01 Cadastrar Produto

Permite ao proprietário informar os dados de um novo produto, como também alterar ou excluir um produto já cadastrado.

Constraints

Pré-condição: O usuário logado no protótipo deve ter a permissão de proprietário.

Pós-condição: Um produto foi incluído, alterado ou excluído do protótipo.

Pós-condição: Produto foi consultado.

Cenários

Consulta produto {Principal}

1. Proprietário solicita cadastro de produto
2. Protótipo apresenta tela com lista de produtos
3. Proprietário opta por localizar um produto
4. Proprietário digita a descrição do produto
5. Protótipo faz uma requisição *web service* para pesquisar produto com a descrição informada
6. Protótipo apresenta os produtos com a descrição informada
7. Proprietário seleciona o produto desejado
8. Protótipo apresenta os dados do produto

Incluir produto {Alternativo}

No passo 3, o proprietário opta por incluir um novo produto.

3.1. Proprietário informa os dados de um novo produto e seleciona Incluir

3.2. Protótipo faz uma requisição *web service* para gravar as informações

3.3. Protótipo do servidor recebe a requisição e insere registro no banco de dados

Alterar produto {Alternativo}

No passo 7, o proprietário seleciona o produto desejado.

7.1. Protótipo apresenta os dados para alteração

7.2. Proprietário edita os dados e seleciona Alterar

7.3. Protótipo faz uma requisição *web service* para gravar as informações

7.4. Aplicação do servidor recebe a requisição e altera registro no banco de dados

Excluir produto {Alternativo}

No passo 7, o proprietário seleciona um produto pressionando o registro por mais tempo.

7.1. Protótipo apresenta o ícone de exclusão

7.2. Proprietário seleciona Excluir

7.3. Protótipo faz uma requisição *web service* para excluir as informações

7.4. Servidor recebe a requisição e exclui o registro no banco de dados

UC02 Manter Clientes

Permite ao usuário informar os dados de um novo cliente, como também alterar ou excluir um cliente já cadastrado.

Constraints

Pré-condição: O usuário logado no protótipo.

Pós-condição: Um cliente foi incluído, alterado ou excluído do protótipo.

Pós-condição: Cliente foi consultado

Cenários

Consulta produto {Principal}

1. Usuário solicita cadastro de cliente
2. Protótipo apresenta tela com lista de clientes
3. Usuário opta por localizar um cliente
4. Usuário digita a nome do cliente
5. Protótipo faz uma requisição *web service* para pesquisar clientes com o nome informado
6. Protótipo apresenta os clientes com o nome informado
7. Usuário seleciona o cliente desejado
8. Protótipo apresenta os dados do cliente

Incluir produto {Alternativo}

No passo 3, o Usuário opta por incluir um novo cliente.

3.1. Usuário informa os dados de um novo cliente e seleciona Incluir

3.2. Protótipo faz uma requisição *web service* para gravar as informações

3.3. Servidor recebe requisição e insere registro no banco de dados

Alterar produto {Alternativo}

No passo 7, o usuário seleciona um cliente.

7.1. Protótipo apresenta os dados para alteração

7.2. Usuário edita os dados e seleciona Alterar

7.3. Protótipo faz uma requisição *web service* para gravar as informações

7.4 Servidor recebe requisição e altera registro no banco de dados

Excluir produto {Alternativo}

No passo 7, o usuário seleciona um cliente pressionando o registro por mais tempo.

7.1. Protótipo apresenta o ícone de exclusão

7.2. Usuário seleciona Excluir

7.3. Protótipo faz uma requisição *web service* para excluir o registro

7.4. Aplicação do servidor recebe a requisição e exclui o registro no banco de dados

UC03 Controlar Estoque

Permite ao proprietário visualizar a situação atual do estoque dos produtos, como também fazer entradas ou saídas de estoque

Constraints

Pré-condição: O usuário logado no protótipo deve ter a permissão de proprietário

Pós-condição: Foi feita uma entrada ou uma saída de estoque

Pós-condição: Estoque foi consultado

Cenários

Consulta estoque {Principal}

1. Proprietário solicita estoque

2. Protótipo faz uma requisição *web service* para atualizar os estoques

3. Protótipo apresenta uma de produtos com a quantidade em estoque

4. Proprietário informa descrição do produto

5. Protótipo faz uma requisição *web service* passando a descrição do produto

6. Proprietário visualiza estoque atual dos produtos pesquisados

Entrada de estoque {Alternativo}

No passo 3, o proprietário opta por dar entrada no estoque de um produto

3.1. Proprietário informa uma quantidade e seleciona a opção (+)

Saída de estoque {Alternativo}

No passo 3, o proprietário opta por dar saída no estoque de um produto

3.1. Proprietário informa uma quantidade e seleciona a opção (-)

UC04 Demonstrar produto

Permite ao representante demonstrar os produtos aos clientes

Constraints

Pré-condição: O usuário logado no protótipo deve ter a permissão de representante

Pré-condição: Foi realizado o cadastro de um produto

Pós-condição: Produto foi demonstrado

Cenários

Consultar produto {Principal}

1. Representante solicita produto
2. Protótipo apresenta tela com lista de produtos
3. Representante opta por localizar um produto
4. Representante digita a descrição do produto
5. Protótipo faz uma requisição *web service* para pesquisar produto com a descrição informada
6. Protótipo apresenta os produtos com a descrição informada
7. Representante seleciona o produto desejado
8. Protótipo apresenta os dados do produto

Exibir imagens do produto {Alternativo}.

No passo 7, o representante opta por visualizar a imagem do produto em tela cheia

7.1. Representante seleciona a imagem do produto

7.2. Protótipo exibe a imagem e tala cheia

UC05 Efetuar Vendas

Permitir ao usuário vender os produtos aos clientes

Constraints

Pré-condição: O usuário logado no protótipo.

Pré-condição: Foi realizado o cadastro de um produto

Pré-condição: Foi realizado o cadastro de um cliente

Pós-condição: Produto foi vendido

Cenários

Vender produto {Principal}

1. Usuário solicita venda
2. Protótipo apresenta tela de vendas
3. Usuário seleciona o cliente
4. Usuário seleciona a opção produtos
5. Protótipo apresenta uma lista com os produtos
6. Usuário seleciona o produto desejado
7. Protótipo adiciona o produto na venda
8. Usuário finaliza venda
9. Protótipo faz uma requisição *web service* para gravar as informações de venda

Venda de vários itens {Alternativo}

Para a venda de dois ou mais itens basta repetir os passos 4, 5 e 6 antes de executar o passo 8. Desta forma no passo 7 o protótipo irá se comportar da seguinte maneira

7.1 Protótipo soma os valor do produto adicionado com o total da venta e atualiza na tela

UC06 Sincronizar Dados

Permitir ao protótipo efetuar requisições *web service* ao sistema do servidor para consultas, incluir ou excluir informações no banco de dados.

Constraints

Pré-condição: O usuário logado no protótipo deve ter a permissão de representante ou proprietário.

Pré-condição: Ter conexão *Wi-Fi* ou 3G

Pós-condição: Dado foi integrado

Cenários

Sincronizam dados {Principal}

1. Protótipo faz uma requisição *web service* solicitando as informações
2. Servidor faz a consulta no banco de dados e retorna para o protótipo
3. Protótipo interpreta as informações retornadas via *web service* e exibe na tela do dispositivo móvel

Erro na sincronização {Alternativo}

No passo 1 o protótipo identifica um erro na chamada do serviço

3.1. Protótipo apresenta uma mensagem de erro na sincronização

UC07 Consultar Estoque

Permite ao representante visualizar a situação atual do estoque dos produtos.

Constraints

Pré-condição: O usuário logado no protótipo deve ter a permissão de representante

Pós-condição: Foi feita uma entrada ou uma saída de estoque

Pós-condição: Estoque foi consultado

Cenários

Consulta estoque {Principal}

1. Representante solicita estoque
2. Protótipo faz uma requisição *web service* para atualizar os estoques
3. Protótipo apresenta uma de produtos com a quantidade em estoque
4. Representante informa descrição do produto
5. Protótipo faz uma requisição *web service* passando a descrição do produto
6. Representante visualiza estoque atual dos produtos pesquisados

APÊNDICE B – Descrição do Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados apresentadas na seção de especificação deste trabalho. Nos Quadros de 6 a 11 estão o dicionário de dados das tabelas do sistema. Os tipos de dados utilizados nos atributos são:

a) *int*: armazena números inteiros;
b) *varchar*: armazena caracteres alfanuméricos*;*
c) *datetime*: armazena data e hora;
d) *decimal*: armazena números de precisão e escala fixos;
e) *numeric*: equivalente ao *decimal*, armazena números de precisão e escala fixos.
f) *char:* corresponde a caracteres.

Quadro 6 - Tabela de usuários

Usuarios					
Controle desta tabela será realizado através do administrador do banco.					
Campo	Descrição	Tipo	Tamanho	Chave primária	Chave estrangeira
id	Id da tabela	*bigint*		Sim	Não
data_hora_alteracao	Data da alteração do registro	*timestamp*		Não	Não
data_hora_criacao	Data da criação do registro	*timestamp*		Não	Não
administrador	Indicativo se o usuário é administrador	*boolean*		Não	Não
login	*Login* do usuário	*character*	50	Não	Não
nome	Nome do usuário	*character*	100	Não	Não
senha	Senha do usuário	*character*	50	Não	Não

Quadro 7 - Tabela de clientes

Clientes					
Armazena o cadastro de Clientes.					
Campo	Descrição	Tipo	Tamanho	Chave primária	Chave estrangeira
id	Id da tabela	*Bigint*		Sim	Não
data_hora_alteracao	Data da alteração do registro	*Timestamp*		Não	Não
data_hora_criacao	Data da criação do registro	*Timestamp*		Não	Não
complemento	Complemento do endereço do cliente	*Character*	30	Não	Não
endereco	Endereço do clietne	*Character*	50	Não	Não
nome	Nome do cliente	*Character*	100	Não	Não
numero	Numero do endereço do cliente	*integer*		Não	Não
cidade	Cidade do endereço do cliente	*Character*	50	Não	Não

Quadro 8 - Tabela de produtos

Produtos					
Armazena o cadastro de Produtos.					
Campo	Descrição	Tipo	Tamanho	Chave primária	Chave estrangeira
id	Id da tabela	*Bigint*		Sim	Não
data_hora_alteracao	Data da alteração do registro	*Timestamp*		Não	Não
data_hora_criacao	Data da criação do registro	*Timestamp*		Não	Não
imagem	Imagem do produto	*text*		Não	Não
preco	Preço do produto	*numeric*	11,2	Não	Não
nome	Nome do produto	*Character*	100	Não	Não

tamanho	Tamanho do produto	*character*	2	Não	Não

Quadro 9 - Tabela de estoques

Estoques					
Armazena o saldo de estoque dos produtos.					
Campo	Descrição	Tipo	Tamanho	Chave primária	Chave estrangeira
id	Id da tabela	*bigint*		Sim	Não
data_hora_alteracao	Data da alteração do registro	*timestamp*		Não	Não
data_hora_criacao	Data da criação do registro	*timestamp*		Não	Não
quantidade	Quantidade em estoque do produto	*bigint*		Não	Não
id_produto	Id do cadastro do produto	*bigint*		Não	Sim

Quadro 10 - Tabela de vendas

Vendas					
Armazena as vendas realizadas pelos usuários.					
Campo	Descrição	Tipo	Tamanho	Chave primária	Chave estrangeira
id	Id da tabela	*bigint*		Sim	Não
data_hora_alteracao	Data da alteração do registro	*timestamp*		Não	Não
data_hora_criacao	Data da criação do registro	*timestamp*		Não	Não
data	Data da venda	*timestamp*		Não	Não
id_cliente	Id do cadastro do cliente da venda	*bigint*		Não	Sim
id_vendedor	Id do cadastro do usuário	*bigint*		Não	Sim

Quadro 11 - Tabela de itens de venda

venda_itens					
Armazena dos itens das vendas.					
Campo	Descrição	Tipo	Tamanho	Chave primária	Chave estrangeira
id	Id da tabela	*bigint*		Sim	Não
data_hora_alteracao	Data da alteração do registro	*timestamp*		Não	Não
data_hora_criacao	Data da criação do registro	*timestamp*		Não	Não
quantidade	Quantidade vendida	*bigint*		Não	Não
id_produto	Id do cadastro do produto vendido	*bigint*		Não	Sim
id_venda	Id da venda	*bigint*		Não	Sim