UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

JAIR RICARDO CELLA

APLICATIVO PARA COLETA DE ITENS DE ESTOQUE POR MEIO DE DISPOSITIVOS MÓVEIS

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2015

JAIR RICARDO CELLA

APLICATIVO PARA COLETA DE ITENS DE ESTOQUE POR MEIO DE DISPOSITIVOS MÓVEIS

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

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2015

APLICATIVO PARA COLETA DE ITENS DE ESTOQUE POR MEIO DE DISPOSITIVOS MÓVEIS

Por

Jair Ricardo Cella

Esta monografia foi apresentada às 13h30 do dia 3 de novembro de 2015 como requisito parcial para a obtenção do título de ESPECIALISTA, no III Curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composto pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

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

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

Prof. Msc. Vinicius Pegorini
Banca
UTFPR – Campus Pato Branco

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

RESUMO

CELLA, Jair Ricardo. Aplicativo para coleta de itens de estoque por meio de dispositivos móveis. 2015 33 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

A separação de itens de estoque para um pedido e a conferência de estoque é uma atividade que se realizada manualmente é bastante trabalhosa. A separação de itens para pedido é realizada, por exemplo, em comércio eletrônico em que a partir de uma listagem de produtos os mesmos devem ser separados para envio ao cliente. A conferência de estoque é uma atividade que pode ter o objetivo de compatibilizar o estoque físico com o registrado. A inclusão de novos itens em estoque pode ser utilizada para contabilizar produtos adquiridos. O uso de dispositivos móveis para fazer a identificação desses produtos e baixá-los de estoque (no caso da separação de itens e atualização de estoque) ou incluí-los no estoque (inclusão de itens ou ajuste de estoque) facilita o trabalho pela não necessidade de anotar em planilhas e essas informações e depois ser necessário transferi-las para um sistema. Porém, se a identificação dos itens puder ser feita por meio de código de barras e com leitura pela câmera de um dispositivo móvel, por exemplo, o trabalho fica bem mais fácil. Considerando essa possibilidade por meio deste trabalho um aplicativo para dispositivos móveis Android foi implementado para que a identificação de itens de estoque, visando separar itens para pedido e atualizar estoque, pudesse ser realizada por meio de código de barras.

Palavras-chave: Plataforma Android. Dispositivos móveis. Leitura de código de barras por câmera de celular.

ABSTRACT

CELLA, Jair Ricardo. Application to collect inventory items through mobile devices. 2015 33 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

Separation of inventory items for an order and inventory conference is an activity that is performed manually is a laborious task. The separation of items to order is performed, for example, in electronic commerce in which a product from the same list should be sorted for delivery to the customer. The stock conference is an activity that can have in order to reconcile the physical inventory with registered. The inclusion of new items in stock can be used to account for purchased products. The use of mobile devices to make the identification of such products and download them in stock (in the case of the separation of items and inventory update) or include them in stock (including items or inventory adjustment) makes the job easier by not need to write down this information into spreadsheets and then have to transfer them to a system. However, if the identification of items can be made by barcode read by the camera and a mobile device, for example, the work becomes much easier. Considering this possibility by means of this work an Android application was implemented for mobile devices that identification of inventory items, to separate items to order and update inventory, could be accomplished by means of bar code.

Palavras-chave: Android platform. Mobile devices. Bar code scanning by camera phone.

LISTA DE FIGURAS

Figura 1 – Visão geral do funcionamento do aplicativo.....19
Figura 2 – Visão esquemática do funcionamento do servidor Java e cliente Android 20
Figura 3 – Protótipo das telas do módulo para separação de itens.....20
Figura 4 – Protótipo de telas do módulo para conferência de estoque.....21
Figura 5 – Protótipo de tela de configurações.....21
Figura 6 – Tela inicial do aplicativo.....22
Figura 7 – Tela para separação dos itens de estoque 22
Figura 8 – Tela para baixa de itens de estoque.....23
Figura 9 – Tela de inclusão de itens 24
Figura 10 – Estrutura de diretório da aplicação 25

LISTA DE QUADROS

Quadro 1 – Tecnologias e ferramentas pra modelagem e implementação do sistema.....17

LISTAGENS DE CÓDIGOS

Listagem 1 – Implementação da construção e atualização do banco de dados 26
Listagem 2 – Código do EstoqueDAO 26
Listagem 3 – Método construtor 27
Listagem 4 – Método buscaEstoqueCodBar 28
Listagem 5 – Método adicionaEstoque 28
Listagem 6 – Código de ligação entre Android e Java 29
Listagem 7 - Método inserir do webservice 30

LISTA DE SIGLAS

CSS	*Cascade Style Sheet*
GPS	*Global Position System*
HTML	*Hypertext markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 10
1.1 CONSIDERAÇÕES INICIAIS 10
1.2 OBJETIVOS 11
1.2.1 Objetivo Geral 11
1.2.2 Objetivos Específicos 11
1.3 JUSTIFICATIVA 11
1.4 ESTRUTURA DO TRABALHO 12
2 REFERENCIAL TEÓRICO 13
2.1 CONTEXTO 13
2.2 CLASSIFICAÇÃO DE APLICAÇÕES MÓVEIS 13
2.3 APLICAÇÕES MÓVEIS EM NEGÓCIOS 15
3 MATERIAIS E MÉTODO 17
3.1 MATERIAIS 17
3.2 MÉTODO 17
4 RESULTADO 19
4.1 ESCOPO DO SISTEMA 19
4.2 MODELAGEM DO SISTEMA 20
4.3 APRESENTAÇÃO DO SISTEMA 21
4.4 IMPLEMENTAÇÃO DO SISTEMA 24
5 CONCLUSÃO 31
REFERÊNCIAS 32

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, os seus objetivos e a justificativa. O texto é finalizado com a apresentação dos capítulos subsequentes que compõem o texto.

1.1 CONSIDERAÇÕES INICIAIS

A separação de itens de estoque ou expostos para venda para compor pedidos de clientes e repor mercadorias é uma atividade que pode ser auxiliada pelo uso de aplicativos computacionais. Uma espécie de listagem desses produtos, que representam os itens de composição de uma nota fiscal ou de um pedido, por exemplo, podem ser apresentados no aplicativo e os mesmos serem separados para compor o referido pedido.

Essa atividade pode ser realizada em estoques ou mesmo de produtos expostos. Exemplos de aplicações desse tipo de atividade são: a separação de itens para composição itens para atender pedidos de compras *online* de supermercados e nesse caso os itens podem vir diretamente das prateiras ou gôndolas do supermercado; a separação de itens para um pedido de cliente em uma loja de materiais de construção em que os itens vêm diretamente do estoque; e a separação de itens para repor mercadorias para venda para clientes, no caso de comércio *online*.

De forma manual a separação pode ser realizada por meio de uma listagem impressa (que pode ser inclusive cópia de nota fiscal, pedido ou outro) na qual o funcionário pode marcar os itens à medida que os vai separando. Com o auxilio de um aplicativo computacional, o funcionário tem acesso à listagem, indica os itens efetivamente separados e pode ser automaticamente realizada a baixa no estoque, se for o caso.

Além de realizar a separação de itens, a conferência de estoques é uma atividade que pode beneficiar-se de um aplicativo computacional que, por meio da câmera, permita ler o código de barras e incluir ou baixar o referido produto do estoque.

Por meio deste trabalho é proposta uma solução de aplicativo para identificação de itens no sentido de separá-los e para conferência de estoques. Visando facilitar e agilizar o trabalho, a solução proposta é para dispositivos móveis e contará com o reconhecimento de código de barras por meio da câmera do dispositivo. Assim, o reconhecimento do item será facilitado, sem a necessidade de digitação. O aplicativo será desenvolvido para a plataforma

Android, com o banco de dados SQLite e a biblioteca Zxing para a leitura e identificação do código de barras.

1.2 OBJETIVOS

A seguir são apresentados os objetivos deste trabalho.

1.2.1 Objetivo Geral

Implementar um aplicativo para dispositivos móveis para identificação de produtos pela leitura do código de barras realizada pela câmera do aplicativo.

1.2.2 Objetivos Específicos

- Fornecer uma ferramenta computacional para dispositivos móveis visando facilitar a conferência que é realizada pelo departamento de estoque pelo auxílio na separação de itens que compõem notas fiscais de saída e de entrada ou outro documento.
- Facilitar a tarefa de conferência de estoque, pela identificação dos itens por meio da leitura do seu código de barras com um dispositivo móvel.
- Permitir a identificação de itens de estoque que compõem uma nota fiscal ou um pedido, por exemplo, para inventário de estoque, atualização de entradas e saídas de estoque por meio da câmera de celular (ou outro dispositivo móvel) com Android.
- Auxiliar na conferência de estoque (entrada ou baixa de itens) por meio da leitura do código de barras realizada com auxílio da câmera do dispositivo móvel.

1.3 JUSTIFICATIVA

A justificativa de uso das tecnologias e no tipo de aplicativo desenvolvido está fundamentada na praticidade que o uso de dispositivos móveis, como os telefones celulares,

podem trazer para identificar itens de um estoque por meio do uso da câmera para identificação do código de barras dos produtos.

O aplicativo desenvolvido interagirá com um sistema utilizado na empresa para a identificação dos itens a serem separados. Outra justificativa do aplicativo é a identificação fácil e rápida de itens de estoque. Ao ler o código de barras os dados do produto são buscados no banco de dados da aplicação e apresentados para consulta na tela do dispositivo.

1.4 ESTRUTURA DO TRABALHO

O restante deste texto está organizado em capítulos. No Capítulo 2 está o referencial teórico centrado no desenvolvimento de aplicativos para dispositivos móveis. O Capítulo 3 apresenta os materiais e o método utilizados no desenvolvimento do trabalho. Os resultados são apresentados no Capítulo 4. E as considerações finais estão no Capítulo 5, seguidas pelas referências utilizadas no texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho que está centrado em desenvolvimento para dispositivos móveis.

2.1 CONTEXTO

Em 2013, o número assinaturas de serviços de celulares móveis alcançou 6.9 bilhões, o que corresponde a 95,5% da população mundial, de acordo com dados do relatório global da ITU Statistics (GLOBAL ICT DEVELOPMENTS, 2015). O aumento da receita com a venda de *smartphones* e *tablets* e a existência de conexões de dados rápidas fornecidas pelos operadores móveis criou uma explosão de aplicações móveis (DELOITTE, 2012). E segundo esses autores, com a introdução de *design* responsivo os sites *web* se tornaram flexíveis o suficiente para trabalhar bem com uma variedade de resoluções que podem ser controladas e ajustar-se a tela de telefones, *tablets* e *laptops* (KARADIMCE; BOGATINOSKA, 2014). Os *smartphones* atuais, com telas maiores e processadores mais rápidos, estarão ocupando o lugar dos *desktops* em áreas como educação, negócio e ambiente doméstico (NAGESH; CAICEDO, 2012).

Aplicativos desenvolvidos para plataformas móveis como, por exemplo, iOS, Android, Blackberry, tem tido um intenso foco de interesse técnico e pela área de negócio. Aplicativos móveis, destinados ao uso pessoal e de empresas, tem, primeiramente, focado na migração de aplicações já utilizadas pelos usuários de redes (RADIA *et al*, 2012). Esses autores ressaltam que poucas aplicações, por exemplo, aplicações baseadas em localização e serviços, não têm correspondentes na rede e esse conjunto de aplicações exclusivamente móvel tem constituído uma pequena fração do universo de aplicações desenvolvidas para o mercado de dispositivos móveis.

2.2 CLASSIFICAÇÃO DE APLICAÇÕES MÓVEIS

Nagesh e Caicedo (2012) categorizam as aplicações móveis podem ser genericamente classificadas em três categorias:

a) Aplicações móveis nativas – como as aplicações móveis nativas são construídas para hardware e sistema operacional específico elas podem obter vantagem dos recursos tecnológicos mais recentes desenvolvidos para os respectivos dispositivos móveis e podem integrar-se com aplicativos *on-board* como calendário, agenda e *email* (LIONBRIDGE, 2012).

b) Aplicações móveis baseadas na *web* – essas aplicações são acessadas por meio de um navegador *web* para dispositivos móveis (LIONBRIDGE, 2012). Para esse autor, como uma aplicação *web* tradicional, uma aplicação *web* móvel é construída com três tecnologias essenciais: *Hypertext markup Language* (HTML) que define os textos estáticos e as imagens; *Cascade Style Sheet* (CSS) que define estilos e apresentação e JavaScript que define as interações e animações.

c) Aplicações móveis híbridas – essas aplicações são desenvolvidas utilizando bibliotecas de código fonte aberto, mas também possuem acesso a alguns dos recursos nativos dos dispositivos como câmera, *Global Position System* (GPS), acelerômetro, sistemas de arquivos, entre outros (NAGESH; CAICEDO, 2012).

Já Unhelkar e Murugesan (2010) classificam as aplicações móveis em cinco categorias:

a) *m-broadcast* - broadcast móvel – aplicações nesta categoria transmitem diferentes tipos de conteúdos para um grande grupo de usuários.

b) *m-information* - informação móvel – apresentam informações solicitadas pelo usuário e o fluxo é unidirecional, no sentido da não iteração do usuário com o aplicativo, ou seja, somente do aplicativo para o usuário.

c) m-transaction - transação móvel - facilita a execução de transações. Por meio desses aplicativos os usuários compram e vendem bens e serviços e permitem rastrear compras e vendas realizadas.

d) *m-oppeation* - operação móvel – suportam aspectos operacionais de um negócio que não envolve interação direta com clientes ou compradores. Elas oferecem acesso conveniente para informações, tais como informações de estoque, agenda de produção, informações de folha de pagamento e balanços.

e) *m-collaboration* colaboração móvel.- mantém colaboração entre empregados e unidades funcionais em uma empresa. Além disso, facilitam a colaboração com outras empresas de interesse ou parceiras de negócio. Aplicações de redes sociais podem auxiliar a criar e gerenciar grupos de pessoas (empregados, clientes, fornecedores) também pertencem a essa categoria.

Para Unhelkar e Murugesan (2010) as aplicações das três últimas categorias são ricas e complexas e necessitam de requisitos e desafios diferentes dos necessários às aplicações das duas primeiras categorias.

Karadimce e Bogatinoska (2014) ressaltam que considerando as diferenças inerentes nessas tecnologias, cada abordagem possui benefícios e desvantagens. Para esses autores as aplicações móveis nativas são construídas para dispositivos móveis específicos e seu respectivo sistema operacional. As aplicações *web* móveis são, geralmente, obtidas por *dowload* de um servidor *web* a cada vez que são executadas, embora aplicações construídas com HTML5 também possam ser executadas no dispositivo móvel de forma *offline* (LIONBRIDGE, 2012). O grau de acesso dos recursos dos dispositivos não é comparável com as aplicações móveis nativas, mas é melhor que o acesso das aplicações *web* móveis (NAGESH; CAICEDO, 2012).

2.3 APLICAÇÕES MÓVEIS EM NEGÓCIOS

As funções básicas de uma empresa estão envolvidas com logística e cadeia de suprimentos, vendas e mercado, serviços, operações e relacionamentos (RADIA *et al*., 2012). Essas atividades funcionais são suportadas por um conjunto de funcionalidades que incluem: infraestrutura, recursos humanos, tecnologia e desenvolvimento de produto, gerenciamento e aquisição (*procurement*) (BASOLE, 2008; TARASEWICH *et al*., 2008).

Aplicações empresariais de sucesso para dispositivos móveis são provavelmente as que permitem aos seus clientes obterem ganhos significativos quando medidos no contexto das seguintes métricas (RADIA *et al*, 2012):

a) Transformação de negócio – essas transformações são obtidas pela automação de tarefas, compartilhamento e rede de informações e pessoas, transformação de processos e relacionamentos e criação de novas oportunidades de receitas.
b) Eficiência – eficiência é obtida por meio de ganhos de produtividade e redução de custos, primeiramente por meio de automação de processos e compartilhamento de informações. A habilidade de tomar decisões baseadas na alta disponibilidade de informação é também baseada na adoção de abordagens móveis como componente da infraestrutura de tecnologia da empresa.
c) Efetividade – efetividade está relacionada com a percepção da melhoria da realização dos objetivos e metas de negócio.

Para Radia *et al.* (2012) similarmente, no contexto dos consumidores, aplicações móveis de sucesso são as que podem positivamente impactar áreas relacionadas da vida e trabalho, incluindo: produtividade (*email*, documentos disponibilizados na nuvem, serviços disponíveis); serviços de utilidade (navegação, comunicação, serviços de alerta, serviços de bilhetagens); entretenimento (jogos, música, vídeo, televisão); compilação de informações (navegação na *web*) e redes sociais.

O sucesso de aplicações móveis é melhor avaliado tendo como base em quão bem essas aplicações permitem realizar tarefas de valor em termos de realizar as suas funcionalidades, isto é, de serem úteis e em alinhamento com objetivos de negócio ou de atendimento a clientes. E quão bem a tecnologia é utilizada para prover alta qualidade e bom desempenho. E, ainda, quão bem é aceito pelos usuários em termos de usabilidade, segurança, funcionalidades e satisfação de uso ao mesmo tempo em que provê um modelo relativamente seguro para uso (RADIA *et al.*, 2012).

Para atender aos requisitos de sucesso dos aplicativos móveis, os desenvolvedores devem escolher as características adequadas para interatividade entre os usuários e os recursos (dados, informações e outros). Exemplos dessas interatividades de controle são (TARASEWICH *et al.*, 2008):

a) Estruturação de interações móveis – a comunicação móvel pode ser tratada como mediada ou situada. A interação mediada permite comunicação entre quaisquer dois pontos com uma localização espacial definida e/ou tempo definido, como ocorre em uma transação de um banco com seus clientes, que pode ser manipulada durante determinadas horas localmente e outras horas por meio de *call center*. Ao passo que uma comunicação situada requer o suporte para certa localização no espaço e tempo.
b) Estilo de controle – os requisitos se o controle sobre a organização ou usuários é centralizado ou distribuído precisa ser integrado em termos de mobilidade e a estratégia da aplicação móvel desde o início do projeto.
c) Estilo de colaboração – os requisitos de negócio devem ser integrados no início do projeto de forma a atender a estratégia de negócio da empresa.
d) Estilo de comunicação – os requisitos que cada evento de comunicação possui de ser uma transação (no qual não há memória prévia de comunicação ou sem estado (*statelass*)) ou um relacionamento (há memória de comunicação, isto é, o estado é mantido (*state-full*)) é um fator importante que deve ser incluído na estratégia de projeto de aplicações móveis pela empresa.

3 MATERIAIS E MÉTODO

Neste Capítulo são apresentados os materiais e o método para a realização do trabalho. Os materiais se referem às ferramentas e tecnologias utilizadas no desenvolvimento do aplicativo. E o método contém as atividades realizadas no desenvolvimento do aplicativo.

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias utilizadas na modelagem e implementação do aplicativo desenvolvido como resultado deste trabalho.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
Android Studio	4.0	http://developer.android.com/tools/studio/index.html	Ambiente de desenvolvimento de aplicações Android
Eclipse luna	4.4.0	https://eclipse.org	Ambiente de desenvolvimento de aplicações
SQLite	3.9	https://www.sqlite.org/	Banco de Dados
Zxing		http://zxing.org/w/decode.jspx	Biblioteca para leitura de código de barras pela câmera do dispositivo
Java	5.6	https://www.java.com	Linguagem de programação
Axis2	1.6.3	http://axis.apache.org/axis2/java/core/	Web Service comunicação Soap
Ksoap	2.1	https://code.google.com/p/ksoap2-android/	Comunicação Soap para o Android

Quadro 1 – Tecnologias e ferramentas pra modelagem e implementação do sistema

3.2 MÉTODO

A seguir estão descritas as etapas realizadas para o desenvolvimento do aplicativo e as principais atividades desenvolvidas em cada uma dessas etapas.

a) Requisitos

O levantamento dos requisitos foi realizado tendo como base as necessidades de supermercados e distribuidoras de medicamentos. Esses requisitos foram obtidos de conversas informais com pessoas relacionadas à área de distribuição de medicamentos. Sendo, assim,

obtida uma visão geral de como o aplicativo deveria funcionar. Essa visão foi definida com base nos interesses e necessidades indicadas pelas pessoas consultadas. Contudo, embora definidas uma visão geral do negócio a partir de possíveis usuários, à medida que as implementações eram realizadas diversos ajustes e modificações e inclusão de outras tecnologias tiveram que serem feitas.

b) Análise e projeto do sistema

A modelagem dos requisitos visou gerar um protótipo do sistema. Uma versão prévia de como o sistema deve funcionar.

c) Implementação

A implementação foi realizada utilizando as ferramentas Android Studio e NetBeans como ambientes de desenvolvimento e as tecnologias indicadas no Quadro 1.

d) Testes

Os testes foram informais foram realizados à medida que as funcionalidades do sistema eram implementadas. Os testes foram informais e visando identificar erros de codificação a atendimentos às funcionalidades definidas.

4 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho que é o desenvolvimento de um aplicativo para dispositivos móveis com Android para identificação de itens de estoque, auxiliando na separação de itens e conferência de estoque.

4.1 ESCOPO DO SISTEMA

O aplicativo desenvolvido tem duas funcionalidades de negócio básicas:

a) Separação de itens que compõem notas fiscais e pedidos - o coletor de dados receberá dados com o nome do produto, o código de barras do produto, a quantidade e o respectivo valor.

b) Conferência de estoque – o coletor receberá o código de barras do produto e a quantidade exportando os dados em um arquivo *eXtensible Markup Language* (XML) para que a aplicação efetue a importação do arquivo para atualização de dados.

A Figura 1 apresenta a visão geral de funcionamento do aplicativo.

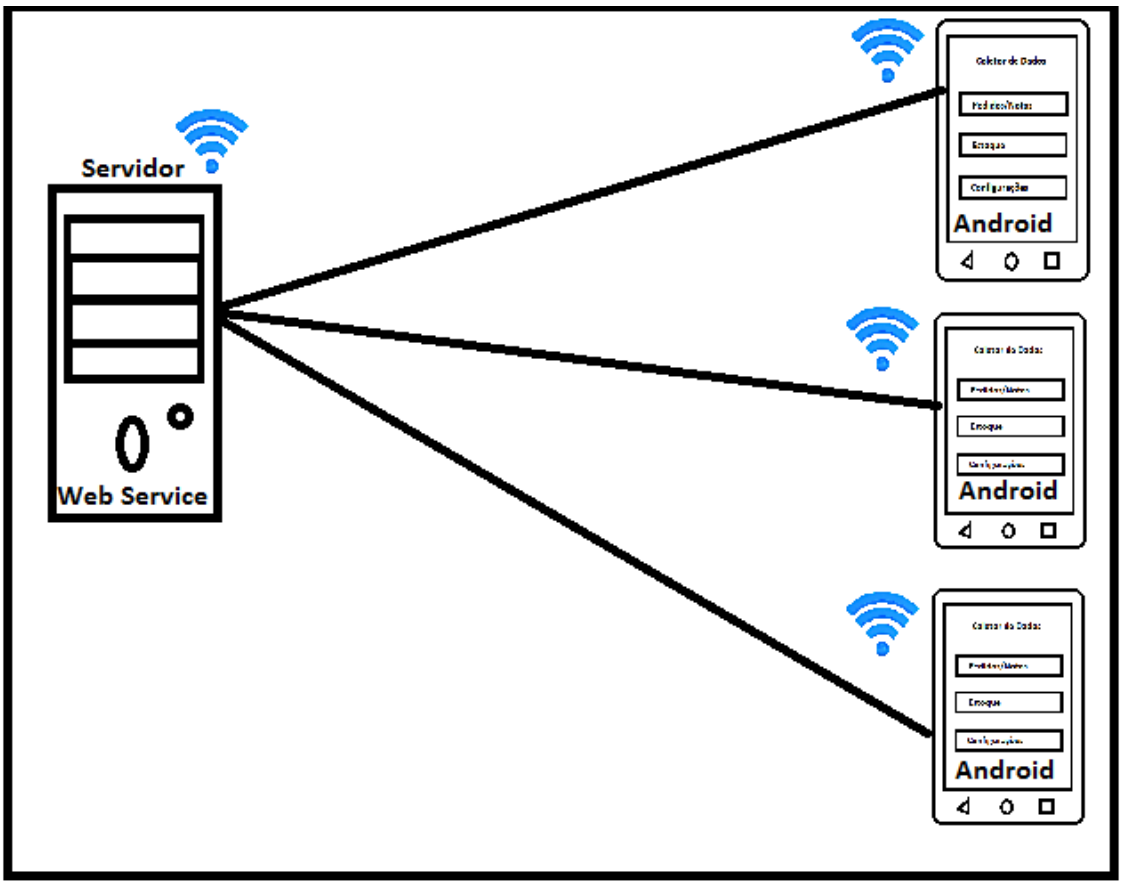

Figura 1 – Visão geral do funcionamento do aplicativo

4.2 MODELAGEM DO SISTEMA

A Figura 2 apresenta o escopo de funcionamento do aplicativo, com as funcionalidades do servidor Java e do cliente Android.

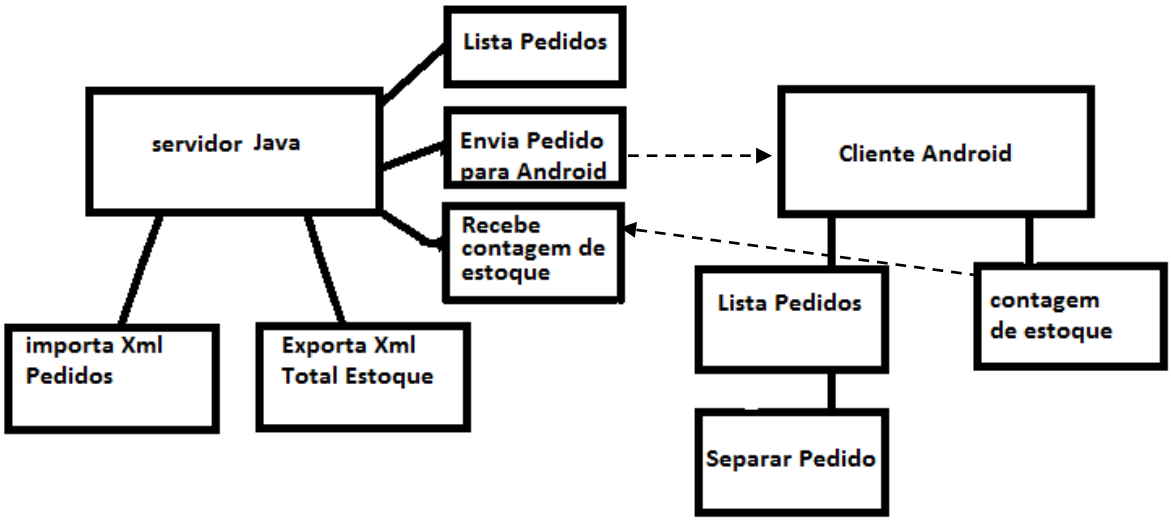

Figura 2 – Visão esquemática do funcionamento do servidor Java e cliente Android

De acordo com a representação da Figura 2, o servidor Java é o responsável importar os pedidos por meio de um XML. Desses dados é gerada a lista de pedidos que é enviada para o cliente Android. O cliente Android a partir desses pedidos pode disponibilizá-los para que seja realizada a contagem de estoque e enviar esses dados para o servidor Java. De posse desses dados, o servidor faz a exportação do XML.

A Figura 3 apresenta o protótipo da interface para a funcionalidade do aplicativo que tem o objetivo de auxiliar na separação de itens de estoque.

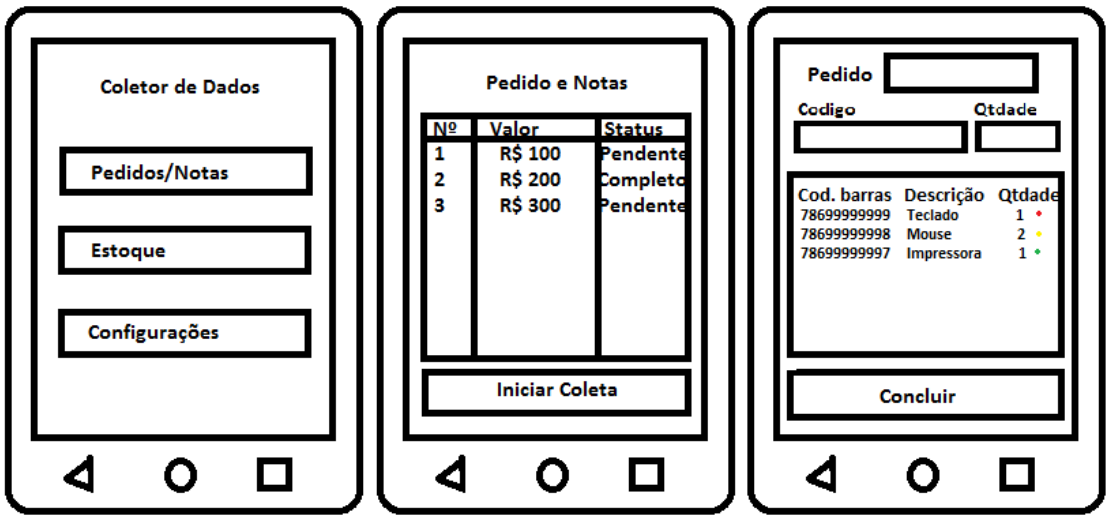

Figura 3 – Protótipo das telas do módulo para separação de itens

Na Figura 4 estão os protótipos das telas para a conferência e identificação de itens de estoque.

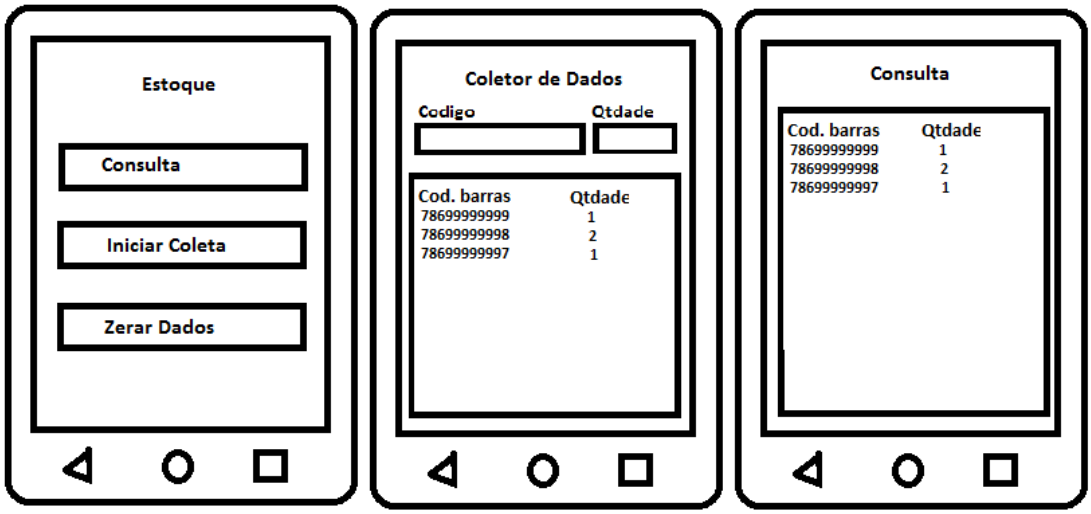

Figura 4 – Protótipo de telas do módulo para conferência de estoque

O protótipo da tela de configurações do aplicativo é apresentado na Figura 5.

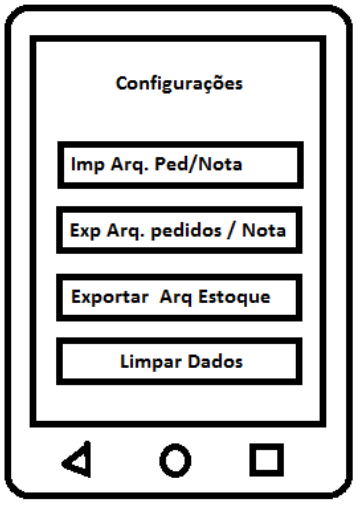

Figura 5 – Protótipo de tela de configurações

4.3 APRESENTAÇÃO DO SISTEMA

Aplicação consiste em duas partes: o servidor Java e o cliente Android. O servidor enviará e receberá informações sempre que solicitado pelo cliente Android. A tela principal do aplicativo Android tem as opções de Pedidos/Notas, Estoque e Configurações. Essa tela é apresentada na Figura 6.

Figura 6 – Tela inicial do aplicativo

Acessando a tela de Pedidos/Notas é possível visualizar os pedidos pendentes que estão no servidor que estão aguardando ou que já foram separados pelo usuário por meio do campo situação do pedido (Figura 7).

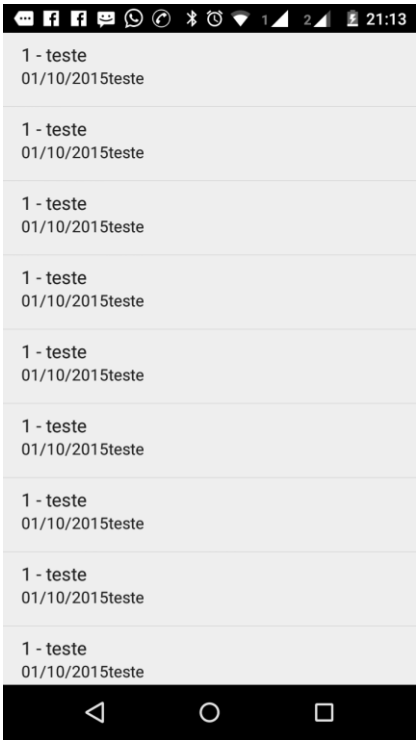

Figura 7 – Tela para separação dos itens de estoque

Ao selecionar um pedido, a aplicação é direcionada para a tela de separação. Nessa tela, apresentada na Figura 8, podem ser observadas as informações gerais do produto além da quantidade, quantidade separada e valor. A baixa da quantidade do produto pode ser realizada de duas maneiras:

a) Quando selecionar um produto o usuário informará a quantidade separada.

b) Podendo também clicar no botão baixar. Essa opção ativará a captura do código de barras pela câmera, realizando, assim, a baixa de uma unidade por código de barras encontrado.

Figura 8 – Tela para baixa de itens de estoque

A aplicação verifica a cada baixa, se o pedido já foi separado completamente e atualizando o *status* de "Pendente" para "Concluído".

Na tela Principal, apresentada na Figura 9, ao ser selecionada a opção "Estoque", o aplicativo permite "zerar dados". Essa operação limpa todos os dados referente ao estoque para que comece a contagem a partir do botão “Iniciar Coleta". Por meio dessa opção, as quantidades por código de barras são somadas de duas formas:

a) Informando o código de barras e a quantidade - o aplicativo procurará o código de barras e caso o encontre acrescentará à quantidade existente a quantidade informada.

b) Por meio do botão captura - o sistema acionará a câmera digital para a captura do código de barras e acrescentará uma unidade.

Figura 9 – Tela de inclusão de itens

Utilizando o Botão "Configurações" é possível importar os pedidos para separação, como também enviar os dados da contagem de estoque para o servidor e posteriormente transformar em arquivo XML para utilizar em aplicações comerciais para o acerto de estoque.

4.4 IMPLEMENTAÇÃO DO SISTEMA

O sistema foi implementado utilizando-se os *Integrated Develpment Environmet* (IDE) NetBeans e o Android Studio.

Cada diretório da estrutura tem um propósito definido, para que a aplicação esteja separada, por pacotes e arquivos contendo código Java. A estrutura de diretório é apresentada na Figura 10.

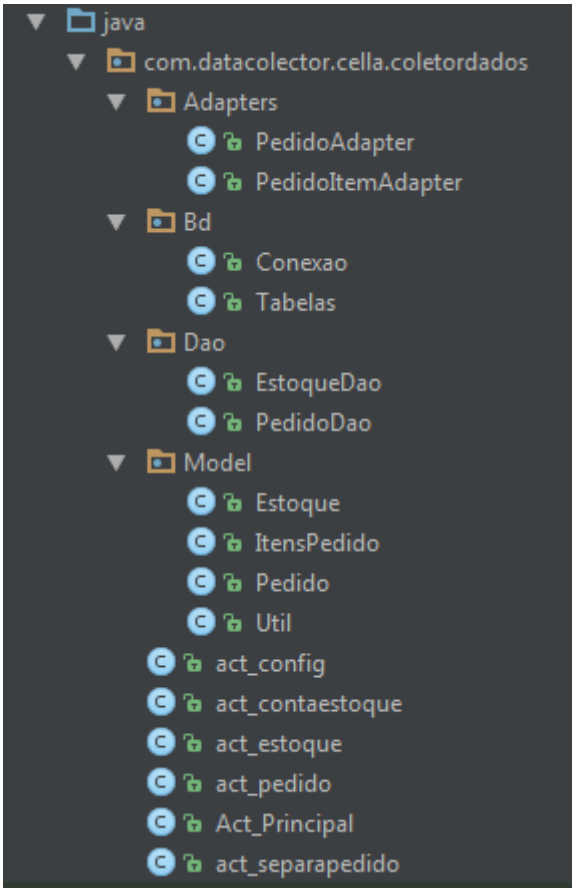

Figura 10 – Estrutura de diretório da aplicação

Na Figura 10 bd\Conexao contém o arquivo de código apresentado na Listagem 1. Esse código é responsável pela construção e atualização do banco de dados. O método onCreate cria o banco de dados formado pelas tabelas que estão adicionadas no bd\Tabelas. O método onUpgrade é executado toda vez que o oldVersion(Versão do Banco antiga) é diferente do newVersion(nova versão do Banco). Nesse trabalho toda vez que o banco de dados é alterado são excluídas todas as tabelas e elas são adicionados novamente.

```
package com.datacolector.cella.coletordados.Bd;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

/**
 * Created by Ricardo Cella on 13/09/2015.
 */
public class Conexao extends SQLiteOpenHelper {
    public static final String nomebanco = "ColetorDados";
    public static final int versao = 4;
    private Tabelas t;
    private String criaTabela;
    private String deletaTabela;
            /*private static String DeletaTabelaEstoque =
```

```
                    private static String DeletaTabelaPedido =
*/
    public Conexao(Context context, String nomebanco, int versao,
                   String scriptCreate, String scriptDelete) {
        super(context, nomebanco, null,versao);
        this.criaTabela   = scriptCreate;
        this.deletaTabela = scriptDelete;
        t = new Tabelas();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        ArrayList<String> a = t.getTabelasAdiciona();
        for (int i = 0; i < a.size() ; i++) {
            db.execSQL(a.get(i));
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        ArrayList<String> d = t.getTabelasExclui();
        for (int i = 0; i < d.size() ; i++) {
            db.execSQL(d.get(i));
        }
        onCreate(db);
    }
}
```

Listagem 1 – Implementação da construção e atualização do banco de dados

O código do *dao\EstoqueDao*, exibido na Figura 10, é apresentado na Listagem 2. Esse código é responsável por incluir e alterar a quantidade de códigos de barra para a contabilização da contagem do estoque.

```
public EstoqueDao(Context ctx) {
    try {
        dbConexao = new Conexao(ctx, Conexao.nomebanco, Conexao.versao);
        db = dbConexao.getWritableDatabase();
        listaEstoque = new ArrayList<Estoque>();

    } catch (Exception e) {
        Log.e("Erro : ", e.getMessage());
    }
}
```

Listagem 2 – Código do EstoqueDAO

Para o método construtor (Listagem 2) é passado o leiaute do arquivo formulário e é criada a conexão com o banco de dados SQLite.

Na Listagem 3 são apresentados três métodos: atribuiValorEstoque, inserirEstoque e AlterarEstoque. Os métodos InserirEstoque e AlterarEstoque passam informações do modeloestoque (bean de entidade) para o AtribuiValorEstoque transformar em um conteúdo de valores para ser adicionado ou alterado no Banco de Dados do Android.

```
public ContentValues atribuiValorEstoque(Estoque est) {
    ContentValues values = new ContentValues();
    values.put("CODBAR", est.getCodbarras());
    values.put("QUANTIDADE", est.getQuantidade());
    return values;
}

public long inserirEstoque(Estoque est) {
    long id = 0;
    try {
        ContentValues values = atribuiValorEstoque(est);
        id = db.insert(nomeTabela, null, values);
    } catch (Exception e) {
        Log.e("Erro : ", e.getMessage());
    } finally {
        return id;
    }
}
public boolean AlterarEstoque(Estoque est) {
    int num = 0;
    try {

        String where = "ID_ESTOQUE=?";
        String[] args = new String[]{String.valueOf(est.getId())};
        num = db.update(nomeTabela, atribuiValorEstoque(est), where,
args);
    } catch (Exception e) {
        Log.e("Erro : ", e.getMessage());
    }
    return (num == 0);

}
```

Listagem 3 – Método construtor

Na Listagem 4, para o método buscaEstoqueCodBar é passada uma *string* com o código de barras que por meio do db.query executa uma consulta no banco de dados. O objeto estoque com as informações da consulta é retornado com o cursor.

```
public Estoque buscaEstoqueCodBar(String codigoBarras) {
    Cursor cursor = null;
    Estoque est = new Estoque();
    String where = "CODBAR=?";
    String[] args = new String[]{codigoBarras};
    try {
        cursor = db.query(nomeTabela, getColunasEstoque(),where, args,
null, null, "ID_ESTOQUE DESC", null);
        if (cursor.getCount() > 0) {
            while (cursor.moveToNext()) {

est.setId(cursor.getInt(cursor.getColumnIndex("ID_ESTOQUE")));

est.setCodbarras(cursor.getString(cursor.getColumnIndex("CODBAR")));

est.setQuantidade(cursor.getInt(cursor.getColumnIndex("QUANTIDADE")));
            }
        }
    } catch (Exception e) {
        Log.e("Erro :", e.getMessage());
    } finally {
```

```
        if (!cursor.isClosed()) {
            cursor.close();
        }
    }
    return est;
}
```

Listagem 4 – Método buscaEstoqueCodBar

O método adicionaEstoque recebe um objeto estoque e tenta buscar o mesmo pelo método da Listagem 5. Se o código de barras já está armazenado no banco ocorre uma alteração do registro existente, caso não exista um novo registro é incluído.

```
public long adicionaEstoque(Estoque est) {
    long  id = 0;
    try {

        Estoque estBusca = buscaEstoqueCodBar(est.getCodbarras());
        if (estBusca.getCodbarras() != null){
            estBusca.setQuantidade((estBusca.getQuantidade() +
est.getQuantidade()));
            AlterarEstoque(estBusca);
            id = estBusca.getId();
        }
        else{
            id =  insertEstoque(est);
        }

    } catch (Exception e) {
        Log.e("Erro : ", e.getMessage());
    } finally {
        return id;
    }
}
```

Listagem 5 – Método adicionaEstoque

O código da Listagem 6 é responsável pela ligação entre o Android e Java.

```
public class act_contaestoque extends ActionBarActivity {
    Button btnIncluir;
    EditText editCodBar, editQuantidade;
    ListView listViewEstoque;
    Estoque est;
    List<Estoque> listaEstoque;
    EstoqueDao dao;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.act_contaestoque);
        editCodBar = (EditText) findViewById(R.id.editCodBar);
        editQuantidade = (EditText) findViewById(R.id.editQuantidade);
        listViewEstoque = (ListView) findViewById(R.id.listEstoque);
        btnIncluir = (Button) findViewById(R.id.btnIncluir);

        dao = new EstoqueDao(this);
        atualizaLista();
        btnIncluir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                btAdicionar();
                atualizaLista();
            }
        });
    }
    public void atualizaLista(){
        Cursor registro = dao.listar();

        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_2, registro,
                new String[]{"CODBAR","QUANTIDADE"}, new
int[]{android.R.id.text1,android.R.id.text2},0);

        listViewEstoque.setAdapter(adapter);

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        return super.onOptionsItemSelected(item);

    }
    private void btAdicionar() {
        //TODO fazer valida
ção dos campos

        est = new Estoque();
        est.setCodbarras(editCodBar.getText().toString());

est.setQuantidade(Long.parseLong(editQuantidade.getText().toString()));

        dao.adicionaEstoque(est);

        Toast.makeText(this, getString(R.string.sucesso),
Toast.LENGTH_LONG).show();
    }
}
```

Listagem 6 – Código de ligação entre Android e Java

No evento *Oncreate* (Listagem 6) é efetuada a ligação dos componentes do Android para que a linguagem Java entenda os mesmos. São criados, também, os eventos dos botões e outros componentes. Já na Listagem 7 é efetuado a inserção no banco Postgres que está no servidor por meio do WebService.

http://localhost:8080/ColetorDadosWs/services/EstoqueDao?wsdl | EstoqueDao.java

```
1  package br.com.coletordados.Dao;
2
3  import java.sql.PreparedStatement;
8
9  public class EstoqueDao {
10
11     public boolean inserirEstoque(Estoque est) {
12         try {
13             Conexao.getConexao();
14             String queryInserir = "INSERT INTO estoque(codbarras, quantidade) VALU
15
16             PreparedStatement pst = Conexao.getPreparedStatement(queryInserir);
17             pst.setString(1, est.getCodbarras());
18             pst.setLong(2, est.getQuantidade());
19             pst.executeUpdate();
20             Conexao.fechaConexao();
21         } catch (Exception e) {
22             e.printStackTrace();
23             return false;
24         }
25
26         return true;
27     }
```

Listagem 7 - Método inserir do webservice

5 CONCLUSÃO

O objetivo deste trabalho foi implementar um aplicativo *mobile* para registrar a contagem de estoque e separação itens para pedidos. O aplicativo foi desenvolvido utilizando as tecnologias Android e Java.

A biblioteca Java *Zxing* que foi utilizada no desenvolvimento do aplicativo facilitou a implementação do projeto do coletor de dados. A *Zxing* facilita o desenvolvimento de aplicações que necessitam de leitura de código de barras, pois sua configuração é realizada por meio de classes. É possível encontrar muita documentação e exemplos de utilização. Contudo, há que se ressaltar que essa biblioteca é de entendimento e uso difíceis, o que ampliou o tempo previsto para desenvolvimento do projeto.

O Android Studio, utilizado como IDE de desenvolvimento de aplicações, foi a ferramenta principal utilizada para o desenvolvimento do projeto Essa ferramenta facilita a construção de telas, como também a pré-implementação de métodos e possui uma boa aceitação pela comunidade. Isso pode ser constatado pelos fóruns e comentários disponibilizados na Internet.

Outra ferramenta importante que foi utilizada e que facilitou o desenvolvimento do *Webservice* foi a IDE Eclipse. Essa ferramenta tem suporte para o desenvolvimento de *Webservice*, além de gerar os serviços automaticamente, por meio Axis2 um biblioteca que permite a troca de mensagens via Soap.

Como trabalhos futuros, ressaltam-se realizar a implementação de algumas funcionalidades, como bloqueio de envio do mesmo pedido para dois dispositivos móveis distintos, garantindo que apenas uma pessoa realizará a separação de um determinado pedido.

REFERÊNCIAS

BASOLE, Raul C. **Applications, technologies, and strategies**, v. 2. Enterprise Mobility, 2008.

DELOITEE. **Observatório móvel Brasil 2012**. Disponível em: <http://www.gsma.com/publicpolicy/wp-content/uploads/2012/03/gsma_brazil_mobile_observatory_ptg.pdf>. Acesso em: 6 fev. 2015.

GLOBAL ICT DEVELOPMENTS. **Report on global mobile-cellular subscriptions by ITU statistics**. Disponível em: http:<ww.itu.int/ITUD/ict/statistics/explorer/index.html>. Acessado em: 05 mar. 2015.

KARADIMCE, Aleksandar; BOGATINOSKA, Dijana Capeska. **Using hybrid mobile applications for adaptive multimedia content delivery**. 37th International Convention Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014, p. 686 – 691.

LIONBRIDGE, **Mobile web apps vs mobile native apps: how to make the right choice.** 2012. Disponível em: <http://www.lionbridge.com/files/2012/11/Lionbridge-WP_MobileApps2.pdf>. Acesso em: 18 abr. 2015.

NAGESH, Anirudh; CAICEDO, Carlos E. **Cross-platform mobile application development**. In: 10th Annual Conference on Telecommunications and Information Technology. Disponível em: <http://www.academia.edu/4233075/Cross-Platform_Mobile_Application_Development >. Acesso em: 26 abr. 2015.

RADIA Nimish; ZHANG, Ying; TATIPAMULA, Mallik; MADISETTI, Vijay K. Next-Generation applications on cellular networks: trends, challenges, and solutions, v. 100, n. 4, April 2012, **IEEE**, p. 841-854.

TARASEWICH, Peter; GONG, Jung; NAH, Fiona F.; DEWESTER, David. Mobile interaction design: integrating individual and organizational perspectives. **Inf. Knowl. Syst. Manage.**, v. 7, p. 121–144, 2008.

UNHELKAR, Bhuvan, MURUGESAN, San. The enterprise mobile applications development framework. **Mobile Computing**, p. 33-39, 2010.