Faculdade de Pindamonhangaba

Carlos Eduardo Pereira Fernandes
Fabio de Oliveira Marcilio
Fabricio Correa Marques

COLETA DE DADOS PARA CONTROLE DE ESTOQUE UTILIZANDO DISPOSITIVOS MÓVEIS

PINDAMONHANGABA – SP
2014

Faculdade de Pindamonhangaba

Carlos Eduardo Pereira Fernandes
Fabio de Oliveira Marcilio
Fabricio Correa Marques

COLETA DE DADOS PARA CONTROLE DE ESTOQUE UTILIZANDO DISPOSITIVOS MÓVEIS

Trabalho de conclusão de curso apresenta como parte dos requisitos para obtenção do Diploma de Bacharel pelo curso Sistemas de Informação da Faculdade de Pindamonhangaba.
Orientador: Prof. Reuel Adimar Lopes.

PINDAMONHANGABA – SP
2014

Fernandes, Carlos Eduardo; Correa Marques, Fabricio; Oliveira Marcilio, Fabio

Coleta de Dados para Controle de Estoque Utilizando Dispositivos Móveis / Fernandes, Carlos Eduardo; Correa Marques, Fabricio; Oliveira Marcilio, Fabio / Pindamonhangaba : FAPI Faculdade de Pindamonhangaba, 2014.

56f. : il.

Trabalho de Conclusão de Curso (Graduação em Sistemas da Informação) FAPI-SP.

Orientador: Prof. Reuel Adimar Lopes.

1.mobile. 2.coleta de dados. 3.Android

I Coleta de Dados para Controle de Estoque Utilizando Dispositivos Móveis II Fernandes, Carlos Eduardo; Correa Marques, Fabricio; Oliveira Marcilio, Fabio

Carlos Eduardo Pereira Fernandes
Fabio de Oliveira Marcilio
Fabricio Correa Marques

COLETA DE DADOS PARA CONTROLE DE ESTOQUE UTILIZANDO DISPOSITIVOS MÓVEIS

Trabalho de conclusão de curso apresenta como parte dos requisitos para obtenção do Diploma de Bacharel pelo curso Sistemas de Informação da Faculdade de Pindamonhangaba.
Orientador: Prof. Reuel Adimar Lopes.

Data:_____
Resultado:_____

BANCA EXAMINADORA

Prof._____Faculdade de Pindamonhangaba
Assinatura_____

Prof._____Faculdade de Pindamonhangaba
Assinatura_____

Prof._____ Faculdade de Pindamonhangaba
Assinatura_____

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, e não somente nestes anos como universitários, mas em todos os momentos de nossas vidas, pois ele é o maior mestre que alguém pode conhecer.

Às nossas famílias, por terem nos apoiado dentro das possibilidades.

Ao nosso orientador Reuel Adimar Lopes, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos na realização desse trabalho.

Aos professores do curso de Sistemas de Informação da FAPI pelos ricos ensinamentos transferidos ao longo desses 4 anos, que somados compõem os conhecimentos indispensáveis para a nossa formação.

À Profª. Alindacir Grassi pela disposição em ajudar sempre quando preciso, e ao nosso Coordenador Prof. Rogério Oliveira de Paula pelo apoio incondicional não só nesse trabalho mas no decorrer de todo o curso.

A todos os nossos grandes amigos que fizemos na turma de graduação, sem exceção, cujo o companheirismo não poderia deixar de agradecer.

A todos que direta e indiretamente fizeram parte de nossa formação, o nosso muito obrigado.

Dedico este trabalho primeiramente a Deus. Aos meus pais e irmãos pela força e oração, sem eles eu não conseguiria. Dedico também aos meus colegas de sala, pela companhia e amizade por esses 4 anos.

(Carlos Fernandes)

Dedico esse trabalho primeiramente a Deus, pois sem ele nada seria possível, meus pais que sempre me apoiaram e me deram a base para tudo que sou hoje, minha filha Bruna e esposa Roberta pelo amor, paciência e por estarem sempre ao meu lado em todos os momentos.

(Fabio Marcilio)

Dedico esse trabalho primeiramente a Deus por mais esse sonho concretizado, a todos os professores que participaram dessa realização, a minha família, e amigos que incentivaram a manter foco para não desistir dos meus ideais.

(Fabricio Marques)

" Quem avança confiante na direção de seus sonhos e se empenha em viver a vida que imaginou para si, encontra um sucesso inesperado em seu dia-a-dia".

(Henry Ford)

RESUMO

O Mercado atual está em plena expansão e com grande desenvolvimento na área de automação comercial, com fluxo intenso de informações importantes a serem trabalhadas pelos programas ERP. As informações são requisitadas e processadas com o menor tempo possível para o bom desenvolvimento dos trabalhos dos colaboradores e o acompanhamento da gerencia. Com esse cenário é extremamente importante o controle total do estoque das empresas com a maior confiabilidade possível. Com o estudo do mercado de ERP, conclui-se que há um grande benefício no uso de um dispositivo coletor de dados no auxílio da alimentação de dados do estoque, trazendo agilidade e confiabilidade. O grande revés é a viabilidades de aquisição de um dispositivo coletor de dados para pequenas e médias empresas, o valor do investimento inibe o uso dessa ferramenta facilitadora. Tendo em vista a grande expansão dos dispositivos mobile nos dias de hoje, surgiu a oportunidade de desenvolvimento da ferramenta de coleta de dados utilizando dispositivos móveis com a plataforma *Android*. Utilizando como dispositivo de entrada a câmera do aparelho, os códigos são lidos e interpretados para posteriormente serem trabalhados conforme a necessidade do software de automação. Com a proposta traz-se a vantagem de se usar equipamentos mais acessíveis que coletores de dados convencionais com o mesmo resultado.

Palavras-chave: mobile, coleta de dados, *Android*

ABSTRACT

The current market is booming (or expanding) and with rapid development of commercial automation, with intensive flows of important information to be developed by the programs ERP. Information is requested and processed with the shortest possible time for the proper development of the work of employees monitoring the management. With that scenario it's extremely important total control of the company stocks with more reliably as possible. According to the study of the market survey of ERP, it is concluded that there is a great benefit in the use of collecting device of stock data, providing agility and reliability. The major setback it's the acquisition feasibility of collecting device of data for small and medium-sized enterprises, the value of a fund's investment inhibits the use of facilitating tool. Bearing in mind the big expansion of mobile devices nowadays, an opportunity for development of the data collection tool using mobile devices with android-based. Using the camera as a supplemental input device, the codes are read and interpreted to subsequently be developed the automation software as needed. With the proposal brings the decisive advantage of using equipment more accessible than data collectors conventional with the same indefensible result.

Keywords: mobile, data collector, Android

Lista de Figura

Lista de Tabelas

Lista de Siglas

SDK: Software Developement Kit

IDE: Integrated Development Environment

API: Aplications Programming Interface

ERP: Enterprise Resource Planning

USB: Universal Serial Bus

PME: Pequenas e medias empresas

JUG: Java User Group

GUJ: Grupo de usuários do JAVA

IBM: International Business Machines

JDK: Java Development Kit

JRE: JAVA Runtime Enviroment

XML: eXtensible Markup Language

HTML: HyperText Markup Language

PHP: Hypertext Preprocessor

JVM: JAVA Virtual Machine

GPS: Global Positioning System

MV: Máquina virtual

WVGA: Wide Video Graphics Array

Compilador JTI: just in time

PPI: Pixel per inch

NFC Near Field Communication:

VPN: Virtual Private Network

EAN: Europe Article Number

URL: Uniform Resource Locator

UI: User Interface

IOS: iPhone Operation System

SUMÁRIO

1. INTRODUÇÃO

Com o passar do tempo os telefones celulares foram evoluindo, ganhando cada vez mais recursos e se tornando um item quase indispensável na vida das pessoas. Mas não foi apenas isso que mudou. Também houve uma mudança significativa para nós, os desenvolvedores de *software*. (MONTEIRO, 2013).

Com o grande avanço da tecnologia, os dispositivos móveis estão se tornando mais poderosos com relação às suas capacidades de armazenamento, de processamento e de comunicação, e mais acessíveis aos consumidores, ao mesmo tempo. (AQUINO, 2007).

O desenvolvimento do aplicativo será baseado nas tecnologias vivenciadas ao longo do curso, usaremos as ferramentas *Android* SDK e a IDE *Eclipse*, que adota a base de programação e funcionamento da linguagem JAVA. Em sua funcionalidade o aplicativo ira auxiliar o cliente a fazer a manutenção e controle do estoque de sua empresa, sendo funcional em qualquer dispositivo *Android* (tablet, smartphone, etc.), utilizando a API *Zxing* responsável pela leitura do código de barras utilizando o recurso de câmera do dispositivo incorporada ao programa, o dispositivo consegue por meio da entrada da câmera a leitura do código de barras do produto, após a leitura o usuário entra com a quantidade, assim o aplicativo gera um arquivo no formato texto que alimentara o software ERP que fará o tratamento das informações, assim atualizando o estoque.

1.1. Objetivos

1.1.1. OBJETIVO GERAL

Desenvolver uma aplicação para dispositivos móveis utilizando a plataforma *Android* que faça a coleta de dados auxiliando pequenas e médias empresas de varejo, que mesmo com pouco investimento possam controlar seu estoque com mais facilidade e agilidade.

1.1.2. OBJETIVOS ESPECÍFICOS

Criar um aplicativo de fácil manuseio e de custo acessível, que faça a coleta dos dados utilizando a câmera do dispositivo e disponibilizando os dados coletados em formato de texto de modo que possam ser utilizados (importado) nos sistemas ERP.

Os dados coletados ficarão disponíveis para serem enviados via e-mail e compartilhado via aplicativo *Dropbox*.

A iniciativa em utilizar dispositivos móveis vem da necessidade de tornar mais acessível a contagem de estoque utilizando dispositivos para auxiliar o trabalho, vendo que um coletor de dados profissional requer um investimento grande, às vezes apenas um aparelho não atende à demanda da empresa, sendo necessários vários aparelhos devido ao porte do estoque, assim se torna inviável a compra de diversos aparelhos de custo elevado.

A maioria dos coletores de dados específicos não faz a alimentação em tempo real das informações já coletadas, nos intervalos das contagens é necessário transferir os dados via cabo serial ou USB, gerando um retrabalho para o operador. Com a aplicação móbile o envio pode ser feito via internet tanto por e-mail ou então utilizando o *Dropbox*.

1.2. Justificativa

Devido à grande demanda de PMEs por soluções ERP para controle de vendas e estoque, surge à necessidade de controle dos produtos, assim o software ERP pode proporcionar um controle total sobre compras e estimativas de vendas. Para buscar agilidade e confiabilidade na contagem de estoque, as empresas investem em equipamentos para coleta de dados, são scanners de mão que na sua maioria fazem a coleta e gera um arquivo para alimentar o software.

O investimento em equipamentos de coleta de dados é relativamente alto para empresas de pequeno e médio porte, às vezes se faz necessário de mais de uma unidade devido à grande quantidade de produtos, assim essa possibilidade acaba ficando inviável devido ao custo benefício. A proposta do aplicativo de coleta de dados para *Android* é de levar a mesma solução, mas por um custo acessível para pequenas e médias empresas.

De acordo com o Instituto de Desenvolvimento Cultural (IDC), o mercado brasileiro de ERP (Enterprise Resource Planning) ou Sistemas Integrados de Gestão Empresarial (SIGE ou SIG) seguirá com um crescimento anual médio de 11%. Os motivos deste avanço são muitos, sobre tudo por conta da atuação de pequenas e médias empresas. No caso das PMEs, a redução de custos e melhoria na qualidade das informações e do atendimento a exigências legais e de normas de grandes empresas que poderão vir a atender são as prerrogativas.

Para as grandes corporações a utilização de sistemas de gestão integrados está madura e bem explorada. O seu crescimento, no entanto, poderá ser impulsionado por um momento de troca ou aperfeiçoamento das soluções já obtidas. A insatisfação por um produto também pode abrir espaço para a mudança de fornecedores, consultorias ou especialistas em reestruturações de operações. Entre as tendências deste setor, a mobilidade, a computação em nuvem e a segmentação, aliadas a uma lógica de atuação unificada entre todas as cadeias de negócio é cada vez maior. (BRANDAO, 2014).

1.3. Estrutura Do Trabalho

Este trabalho está dividido em capítulos, estruturados de forma clara e organizada, propiciando ao leitor um claro entendimento dos conceitos abordados, e metodologias utilizadas para sua realização.

Capítulo 1 – Introdução

O primeiro capítulo aborda os objetivos, justificativas e a estrutura do trabalho.

Capítulo 2 – Revisão da Literatura

O segundo capítulo apresenta o embasamento teórico, assim como a descrição das tecnologias utilizadas para o desenvolvimento da aplicação.

Capítulo 3 – Metodologia

O terceiro capítulo apresenta as especificações do aplicativo e os diagramas da análise: diagrama de caso de uso, especificação de caso de uso, diagrama de classes, diagrama de atividades e o uso da biblioteca Zxing.

Capítulo 4 – Resultados e discussões

O quarto capítulo apresenta a organização do projeto, e o detalhamento de suas classes. Interface do aplicativo contendo imagens e propostas futuras.

Capítulo 5 – Conclusão

O quinto capitulo apresenta as considerações finais sobre o desenvolvimento da aplicação expondo os resultados obtido.

Capítulo 6 – Referencias

O sexto capitulo descreve as fontes de pesquisas para embasar na elaboração do projeto.

Capítulo 7 – Anexos

O sétimo capitulo disponibilizará um CD-ROM com os códigos-fontes e aplicativos do projeto.

2. REVISÃO DA LITERATURA

2.1. Java

A linguagem Java surgiu em 1991 quando cientistas da Sun, liderados por Patrick Naughton e James Gosling, projetaram uma linguagem pequena para ser utilizada em eletrodomésticos em geral. O projeto foi nomeado como “Green”. Com o crescimento da internet, os criadores notaram a importância de um browser, a partir disso foi desenvolvido o *HotJava*, este browser reconhecia applets e, por isso, tinha uma máquina virtual Java embutida em si. Em 1995, a Netscape decidiu tornar a próxima versão do seu browser (versão 2.0) apta a reconhecer applets Java. A partir de então, todas as versões posteriores do Netscape, inclusive do Internet Explorer, reconheciam Java. (JAQUES,2007)

A linguagem Java foi projetada para ser utilizada no desenvolvimento de aplicações que consumam o mínimo de recursos do sistema e que possam ser executadas em diferentes plataformas de hardware e software. Os aplicativos em Java são compilados em um código de *bytes* independentes de arquitetura. (MARURI,2004).

A figura abaixo ilustra o processo de compilação e interpretação de programas Java. Pode-se compilar os programas Java em qualquer plataforma que possua um compilador Java. O produto da compilação pode então ser executado em qualquer implementação da Máquina Virtual Java. Portanto, o mesmo programa Java pode ser executado em qualquer sistema operacional sem a necessidade de recompilação. (RIBEIRO, 2003).

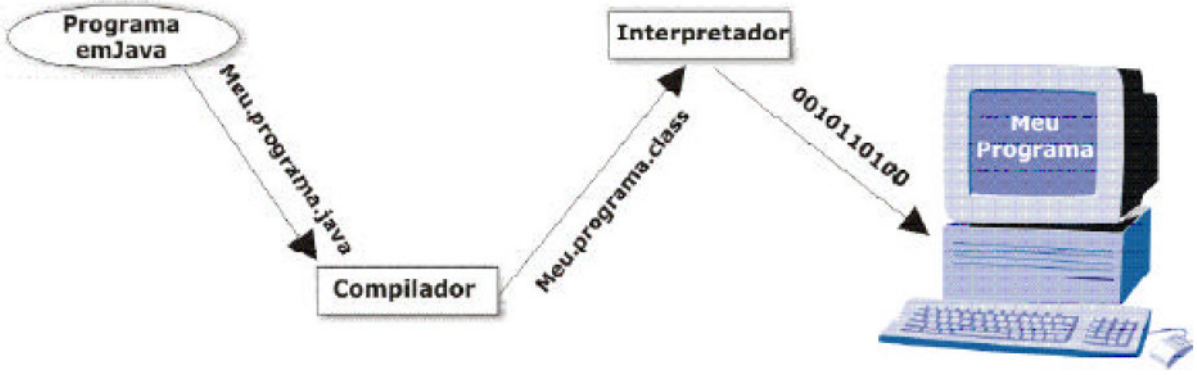

Figura 1: Compilação e Interpretação de Programas Java

"O aspecto da utilização do Java em multiplataforma é muito importante, porque os programadores não necessitam ficar preocupados em saber em qual máquina o programa será executado, uma vez que um mesmo programa pode ser usado num PC, num Mac ou em um computador de grande porte. É muito melhor para uma empresa desenvolver um software que possa ser executado em qualquer lugar independente da máquina do cliente". (FURGERI, 2002).

A comunidade de desenvolvedores Java reúne-se em grupo denominados JUGs (Java User Groups). No Brasil o movimento de grupos de usuários expandiu-se bastante e tem formado alguns dos maiores grupos de usuários Java do mundo, como por exemplo o PortalJava, GUJ e o JavaFree. (VELLOSO, 2011).

A Oracle adquiriu a Sun Microsystems por US$ 7,4 bilhões, após várias tentativas da IBM que comprar. A Oracle disse em comunicado que espera aumentar seu lucro por ação. A empresa disse ainda que estima que a compra da Sun vai contribuir com mais de US$ 1,5 bilhão para o lucro operacional da Oracle e para mais de US$ 2 bilhões no segundo ano. (MOREIRA, 2009).

2.1.1. Eclipse

O Projeto Eclipse foi originalmente criado pela IBM em Novembro de 2001 e apoiado por um consórcio de fornecedores de software. A Eclipse Foundation foi criada em janeiro de 2004 como uma corporação sem fins lucrativos independente para atuar como o mordomo da comunidade Eclipse. A corporação independente sem fins lucrativos foi criada para permitir um fornecedor neutro e aberto, transparente comunidade a ser estabelecida em torno de Eclipse. Hoje, a comunidade Eclipse é composto por indivíduos e organizações de uma seção transversal da indústria de software. (ECLIPSE,2014).

Eclipse é um IDE para desenvolvimento Java, porém suporta várias outras linguagens a partir de plugins como C/C++, PHP, ColdFusion, Python, Scala e plataforma *Android*. Ele foi feito em Java e segue o modelo open source de desenvolvimento de software. Atualmente faz parte do kit de desenvolvimento de software recomendado para desenvolvedores *Android*. (ECLIPSE,2014)

O projeto *Eclipse* foi iniciado na IBM que desenvolveu a primeira versão do produto e doou-o como software livre para a comunidade. O gasto inicial da IBM no produto foi de

mais de 40 milhões de dólares. Hoje, o *Eclipse* é o IDE Java mais utilizado no mundo. (JUNNARKAR,2001)

O *Eclipse* é um ambiente de desenvolvimento de software integrado, do inglês Integrated Development Environnement (IDE). Como o próprio nome indica é um ambiente de trabalho, composto por várias ferramentas integradas, pensado e elaborado para desenvolver qualquer tipo de aplicação. (ECLIPSE,2014)

Uma das maiores vantagens do *Eclipse* é o poder de integrar diversos *plug-ins* que são gerenciados por um núcleo interno, fazendo com que outros desenvolvedores possam criar seus próprios *plug-ins* e utiliza-los no *Eclipse*. (FLAUSINO, 2007)

2.1.2. NETBEANS

O *NetBeans* foi iniciado em 1996 por dois estudantes tchecos na Universidade de Charles, em Praga, quando a linguagem de programação Java ainda não era tão popular como atualmente. Primeiramente o nome do projeto era Xelfi, em alusão ao Delphi, pois a pretensão deste projeto era ter funcionalidades semelhantes aos IDEs então populares do Delphi que eram mais atrativas por serem ferramentas visuais e mais fáceis de usar, porém com o intuito de ser totalmente desenvolvido em Java. (NETBEANS,2014).

Em 1999 o projeto já havia evoluído para uma IDE proprietário, com o nome de *NetBeans* DeveloperX2, nome que veio da ideia de reutilização de componentes que era a base do Java. Nessa época a empresa Sun Microsystems havia desistido de sua IDE Java Workshop e, procurando por novas iniciativas, adquiriu o projeto *NetBeans* DeveloperX2 incorporando-o a sua linha de softwares. (NETBEANS,2014).

A IDE *NetBeans* auxilia programadores a escrever, compilar, debugar e instalar aplicações, e foi arquitetada em forma de uma estrutura reutilizável que visa simplificar o desenvolvimento e aumentar a produtividade, pois reúne em uma única aplicação todas estas funcionalidades. (NETBEANS,2014).

Totalmente escrita em Java, mas que pode suportar qualquer outra linguagem de programação que desenvolva com Swing, como C, C++, Ruby e PHP. Também suporta linguagens de marcação como XML e HTML. (NETBEANS,2014).

2.1.3. JDK e JRE

Java Runtime Environment (JRE) significa Ambiente de Tempo de Execução Java, e é utilizado para executar as aplicações da plataforma Java. É composto por bibliotecas (APIs) e pela Máquina virtual Java (JVM). (CIPOLI, 2012).

Máquina virtual Java (do inglês Java Virtual Machine - JVM) é um programa que carrega e executa os aplicativos Java, convertendo os bytecodes em código executável de máquina. A JVM é responsável pelo gerenciamento dos aplicativos, à medida que são executados. (LAUREANO,2008).

Development Kit (JDK) significa Kit de Desenvolvimento Java. Android SDK é o software utilizado para desenvolver aplicações no Android, que possui um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações. (LECHETA,2010).

2.2. Android

Hoje, há 1,5 bilhão de televisores em uso ao redor do mundo. Um bilhão de pessoas estão na Internet. Mas quase três bilhões de pessoas têm um telefone celular, tornando-o um dos produtos de consumo do mundo de maior sucesso. Dessa forma, construir um aparelho celular superior enriqueceria e melhoraria as vidas de inúmeras pessoas (OHA, 2011). Em um contexto mundial, esta é a missão do Open Handset Alliance™, grupo formado pelas empresas líderes de mercado em tecnologia móvel.

Como objetivo, o grupo tende a definir uma plataforma única e aberta para celulares para deixar os consumidores mais satisfeitos com o produto final. Também criar uma plataforma aberta e flexível para o desenvolvimento de aplicações corporativas (LECHETA,2010).

Desenvolvido especialmente para dispositivos moveis como aparelhos celulares e tablets, o *Android* e uma plataforma composta de um sistema operacional, middlewares e um conjunto de aplicativos principais como os Contatos, Navegador de Internet e o Telefone propriamente dito. Além disso, existe o *Android* SDK que e um conjunto de ferramentas e APIs para o desenvolvimento de aplicativos para a plataforma, utilizando a linguagem Java (MONTEIRO,2013).

2.2.1. HISTORIA

Em outubro de 2003 na cidade de Palo Alto, Califórnia, Andy Rubinera, Nick Sears e Chris White fundaram a empresa Android Inc. No início a empresa desenvolvia tecnologia totalmente independente de outras empresas. Dentre os seus principais objetivos, estava o desenvolvimento de um sistema operacional avançado para câmaras digitais. Algum tempo depois, percebeu-se que o mercado para tais dispositivos não era grande suficiente, então a equipe desviou os seus esforços para produzir um sistema operacional para smartphones, rivalizando assim com outros sistemas da categoria, como por exemplo, o Symbian. (GUIMARAES, 2013).

Desde 1997 os celulares ainda não tinham se tornado multiuso, não tinham ferramentas multifuncionais pessoais. Ninguém ainda via a necessidade de navegação na Internet, tocar MP3, ou qualquer uma das muitas funções que estamos acostumados a usar em um celular (DIMARZIO, 2008). No entanto, mesmo se a necessidade estava presente na época, a falta de memória do dispositivo e capacidade de armazenamento era um obstáculo ainda maior para superar, ou seja, os celulares até meados 2002 serviram apenas para executar e receber chamadas, acompanhar seus contatos, e, eventualmente, enviar e receber mensagens de texto curtas mensagens, pois até câmeras em celulares não eram comumente encontrados nas mãos dos consumidores (DIMARZIO, 2008). Com o passar do tempo os telefones celulares foram evoluindo, ganhando cada vez mais recursos e se tornando um item quase indispensável na vida das pessoas.

Antes, o mercado de desenvolvimento para celulares era praticamente restrito aos fabricantes e operadoras que controlavam a criação e inclusão dos aplicativos em seus aparelhos. A liberação, por parte dos fabricantes, de um kit de desenvolvimento de software (SDK) para suas plataformas e a criação de lojas para a distribuição de aplicativos viabilizou a abertura deste mercado para praticamente qualquer empresa ou desenvolvedor, criando assim novas oportunidades de negócio.

A plataforma *Android* desfruta hoje de um papel de destaque no mercado, tanto pela quantidade significativa de dispositivos produzidos como também por oferecer uma API rica, disponibilizando fácil acesso a vários recursos de hardware, tais como Wi-Fi e GPS, além de boas ferramentas para o desenvolvedor. A facilidade de desenvolver utilizando uma linguagem de programação (Java) bastante disseminada, a simplicidade e baixo custo para a publicação de aplicativos na loja Google Play e a quantidade de dispositivos *Android* em uso no mundo só fazem aumentar a popularidade da plataforma (MONTEIRO,2013).

2.2.2. ESTRUTURA

O *Android* é um sistema operacional baseado no kernel do Linux. Apesar de ter sido desenvolvido inicialmente para smartphones, hoje é usado em diversas outras aplicações como tablets, netbooks, relógios, etc. Existe pouca coisa em comum com distribuições Linux convencionais (embarcadas ou não), lembrando que um sistema embarcado (ou sistema embutido) é um sistema micro processado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. (GOMES,2012).

Nele não há distinção entre aplicativos que são nativos e os demais. Isso possibilita uma grande customização do sistema operacional, permitindo a substituição completa de aplicativos nativos por outros, criados por terceiros. Além disto, todos os aplicativos têm acesso as mesmas funcionalidades.

2.2.3. ARQUITETURA

Sua arquitetura possui basicamente 4 camadas conforme figura abaixo, em seguida uma breve descrição das mesmas:

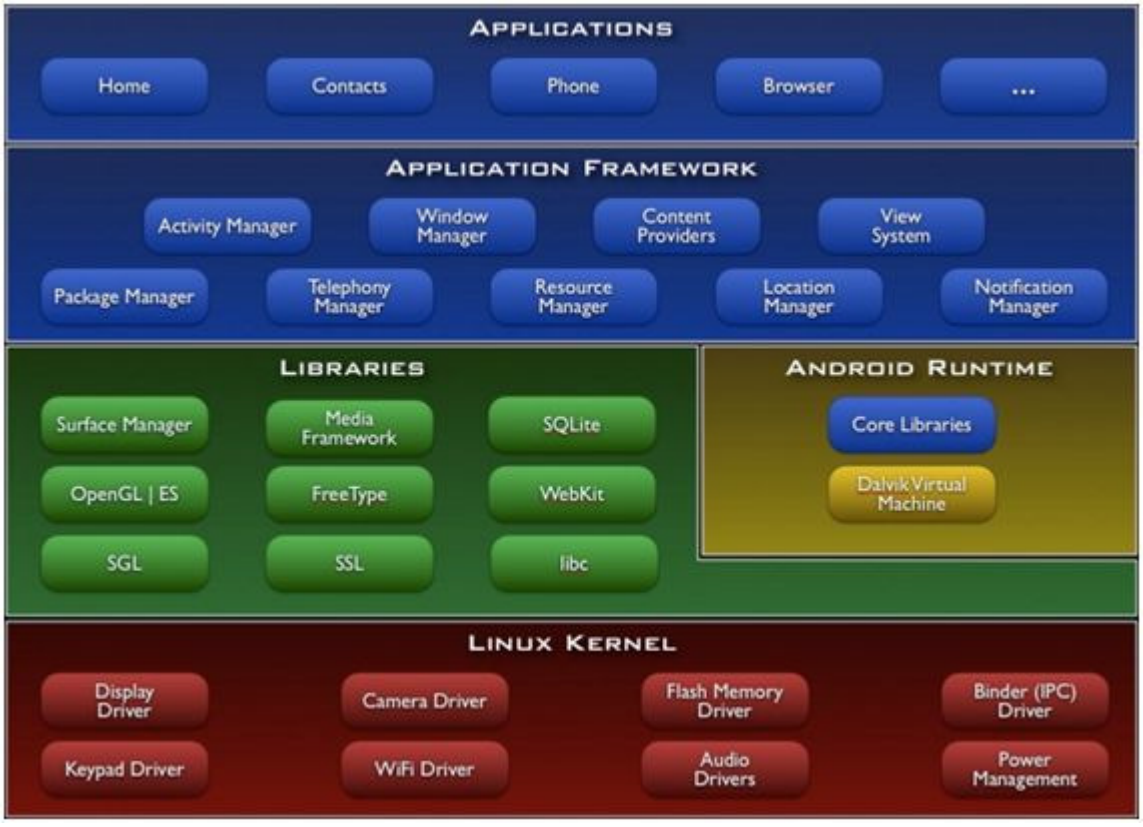

Figura 2: Arquitetura do sistema operacional Android

Fonte: Gomes, R

- Aplicações: A camada de aplicativos é a que está no topo da pirâmide da arquitetura do sistema operacional *Android*, composta pelo conjunto de aplicações nativas do mesmo. Dentre estes pode–se citar: cliente de e-mail, despertador, calendário, jogos, mapas, browser e internet, etc.
- Framework: A camada de framework nativo disponibiliza aos desenvolvedores as mesmas Applications Programming Interface (APIs) – Interface de Programação de Aplicativos utilizados para a criação de aplicações originais do sistema operacional *Android*. Este framework permite que o programador tenha o mesmo acesso ao sistema que os aplicativos da camada de aplicativos possuem. Este framework foi criado para abstrair a complexidade e simplificar a reutilização de procedimentos. Essa camada funciona como um link com a camada de bibliotecas do sistema operacional que serão acessadas através de APIs contidas no framework.
- Bibliotecas e serviços: Essas bibliotecas são responsáveis por fornecer funcionalidades para manipular o áudio, vídeo, gráficos, banco de dados e browser. Algumas bibliotecas são a Bionic, aOpenGL/ES para trabalhar com interface gráfica, e a SQLite para trabalhar com banco dedados. Aqui também estão os serviços usados em camadas superiores, como máquina virtual Java Dalvik. A maior parte destas bibliotecas e serviços estão desenvolvidos em C e C++.
- O *Android* Runtime: Permite que cada thread rode sua própria instância da MV (máquina virtual). Embora no desenvolvimento de aplicativos seja utilizada a linguagem Java, as aplicações não são executadas em uma máquina virtual Java tradicional, e sim em outra chama de Dalvik. Essa máquina virtual é otimizada especialmente para dispositivos móveis. A plataforma Google *Android* permite o desenvolvimento de aplicativos na linguagem Java. Essa máquina virtual foi construída pelos engenheiros da Google, para obter um consumo mínimo de memória e isolamento de processos. Ela permite que as aplicações escritas em linguagem Java sejam executadas normalmente.
- Kernel Linux: A camada do kernel é baseada em um sistema do sistema operacional Linux versão 2.6. Esta camada atua também como responsável pela abstração entre o hardware e os aplicativos e é responsável pelos serviços principais do sistema operacional *Android*, como o gerenciamento de memória e de processos. Várias funções do kernel são utilizadas diretamente pelo *Android*, mas muitas modificações foram feitas para otimizar memória e tempo de processamento das aplicações. Essas modificações incluem

novos dispositivos de drivers, adições no sistema de gerenciamento de energia e um sistema que possibilita terminar processos de maneira criteriosa quando há pouca memória disponível.

2.2.4. FUNCIONAMENTO DE UM APLICATIVO ANDROID

O diagrama abaixo é de fundamental importância para o correto entendimento do funcionamento de uma aplicação *Android*.

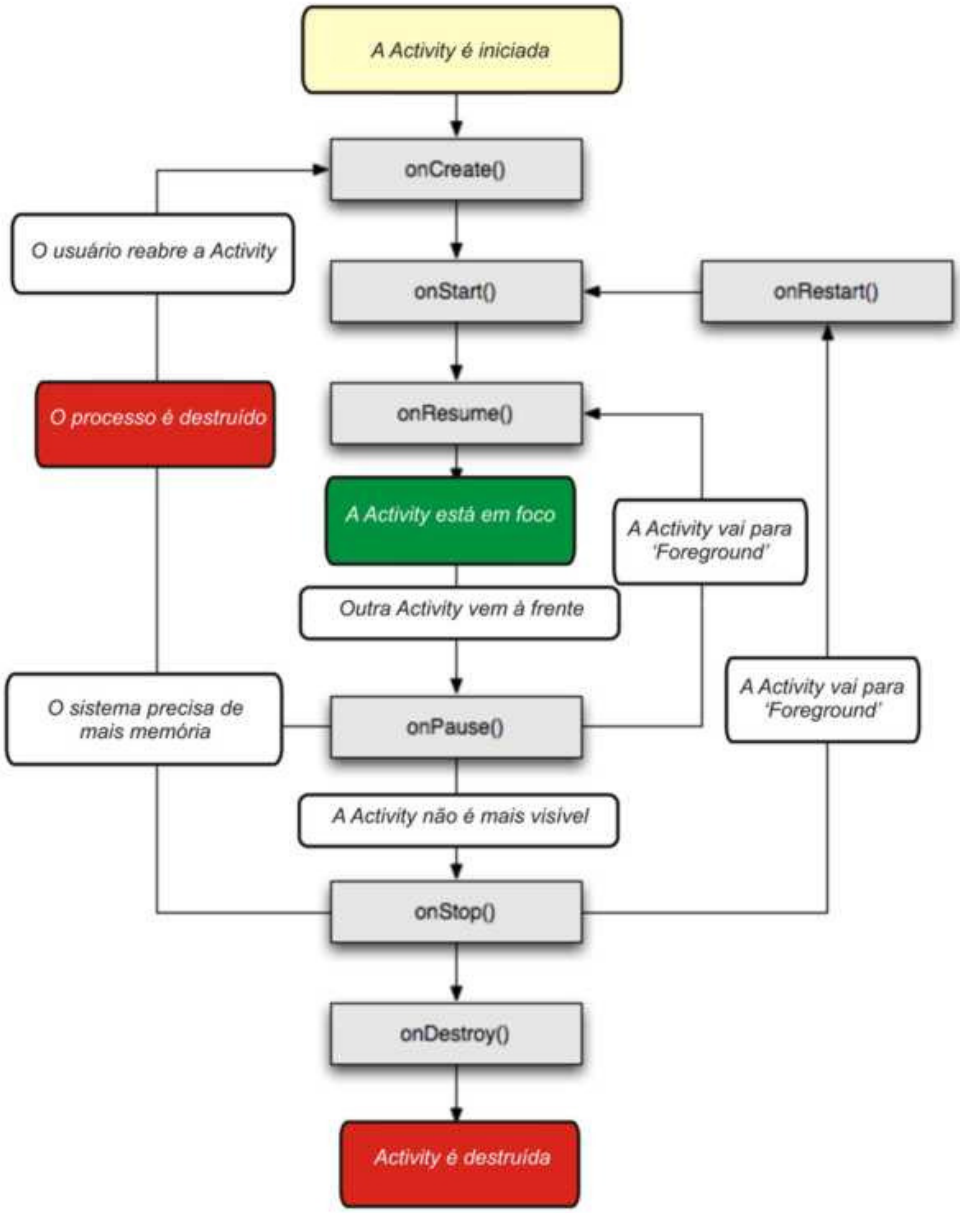

Figura 3: Ciclo de vida de uma aplicação

Fonte: LECHETA, R.

Funções da Activity:

onCreate() É a primeira função a ser executada quando uma Activity é lançada. Geralmente é a responsável por carregar os layouts XML e outras operações de inicialização. É executada somente uma vez durante a "vida útil" da Activity.

onStart() É chamada imediatamente após a onCreate() – e também quando uma Activity que estava em background volta a ter foco.

onResume() Assim como a onStart(), é chamada na inicialização da Activity (logo após a própria onStart()) e também quando uma Activity volta a ter foco. Qual a diferença entre as duas? A onStart() só é chamada quando a Activity não estava mais visível na tela e volta a ter o foco, enquanto a onResume() sempre é chamada nas "retomadas de foco".

onPause() É a primeira função a ser invocada quando a Activity perde o foco (ou seja, uma outra Activity vem à frente).

onStop() – Análoga à onPause(), só é chamada quando a Activity fica completamente encoberta por outra Activity (não é mais visível).

onDestroy() A última função a ser executada. Depois dela, a Activity é considerada "morta" – ou seja, não pode mais ser relançada. Se o usuário voltar a requisitar essa Activity, outro objeto será construído.

onRestart() Chamada imediatamente antes da onStart(), quando uma Activity volta a ter o foco depois de estar em background.

2.2.5. VERSÕES DO ANDROID

Android 1.0(Astro): Lançado em fevereiro de 2009. Esta versão tinha como principais características suporte a navegadores HTML, reprodução de vídeos via YouTube, além de outros serviços disponibilizados pela Google. Na primeira versão foi implantado a janela de notificações, já incluso também o *Android* Market para realização de downloads e atualização. (BARROS, 2013).

Android 1.5(cupcake): Lançada em abril de 2009, foi lançada em grande escala comercialmente. A atualização trouxe correções de falhas e diversas funcionalidades ao sistema. Além disso, o Cupcake foi o primeiro telefone touch-screen com a plataforma. (BARROS, 2013).

Android 1.6(Donut): O lançamento do HTC Hero com Android 1.6 foi um dos marcos na história, e foi aí que as pessoas começaram a ver o potencial, e que a Apple e a Microsoft começaram a se ligar que não estavam sozinhas no mercado. O Hero era um celular muito bom para a época, e a maiora das pessoas queria tê-lo. (LUCIEL, 2010).

Android 2.0 e 2.1(Eclair): Lançado em outubro de 2009 e atualizado em janeiro de 2010. Esta versão, trouxe o suporte à sincronização de contas permitindo a possibilidade de adicionar diversas contas no mesmo dispositivo com isso sincronizando e-mails e contatos.

Foi adicionado também Bluetooth 2.1, detalhes dos contatos ao clicar no mesmo e recursos inéditos para câmera. (BARROS, 2013).

Android 2.2(Froyo): Lançado em 2010, esta versão ficou conhecido por ser a mais rápida do *Android*. Contendo o USB Tethering e a possibilidade de usar o smartphone como um hotspot Wi-Fi.

Foi adicionado também conteúdo Flash, desempenho aprimorados, chamadas por voz, opção para desativar o uso de pacote de dados, troca rápida de idiomas via teclado. Além disso contemplando o suporte aos displays de 320 e 720 ppi. (BARROS, 2013).

Android 2.3(Gingerbread): Lançado em dezembro de 2010, mantendo as características visuais da versão anterior, com apenas pequenas diferenças, como na interface das ligações.

Foi adicionado também um novo gerenciador de downloads, suporte para diversas câmeras, melhora na autonomia de bateria e compatibilidade com tecnologia NFC. (BARROS, 2013).

Android 3.0(Honeycomb): Lançado em fevereiro de 2011, ganhou nova interface e suporte a tablets. Foi adicionada as barras de sistema possibilitando o acesso rápido a notificações.

Foi adicionado também teclado redesenhado, simplificação das multitarefas e navegação na Internet em abas. (BARROS, 2013).

Android 4.0(Ice Cream Sandwich): Lançado em outubro de 2011 e atualizado em março de 2012, permitiu customizar o launcher, com opção de acrescentar widgets e apps simplesmente arrastando-os para a tela principal.

Foi adicionado também suporte a pastas, gerenciador de tarefas com screenshots dos aplicativos, recurso de deslizar para fechar, além do Google Chrome embutido. (BARROS, 2013).

Android 4.1, 4.2 e 4.3(Jelly Bean): Lançada em julho de 2012 com melhoria nas opções de acessibilidade, expansões das notificações, opção de desativar alertas para cada app, organização de atalhos e widgets automaticamente, transferência de dados via Bluetooth para o *Android* Beam, melhorias na câmera e na busca por voz, além da adição do auxiliar Google Now. A versão 4,2 apresentou mais recursos como: fotos panorâmicas, digitação através de gestos, tela de bloqueio melhorada, multiusuários, VPN funcionando em segundo plano, além das falhas corrigidas. (BARROS, 2013).

Android 4.4 (Kit Kat): Lançada em setembro de 2013. Possui refinamentos visuais e foi otimizado para funcionar melhor com smartphones de baixo custo. Houve pequenas melhorias em algumas partes do sistema, como no discador. Houve melhorias de desempenho para tornar o KitKat mais suave e responsivo nos smartphones de baixo custo, com 512 MB de RAM. Além da otimização no consumo de memória. (HIGA, 2013)

Android 5.0 (Lolipop): Última versão do Android, na qual virá nos smartphones da nova geração.

2.2.6. SDK DO ANDROID

É a sigla de Software Development Kit, ou seja, Kit de Desenvolvimento de Software ou Kit de Desenvolvimento de Aplicativos. Várias tecnologias vêm com SDK para programadores criarem um software de interação entre o hardware e o usuário.

No caso do *Android*, é possível realizar o download desse SDK através do site: https://developer.*Android*.com/sdk/index.html.

2.3. Desenvolvimento Mobile no Brasil

O domínio do Google no mercado de smartphones atingiu um novo patamar, com o *Android* atingindo recorde de participação dos embarques globais de smartphones, com 84,6% de acordo com pesquisa realizada pela Strategy Analytics.

O crescimento do mercado de smartphones *Android* durante o segundo trimestre reduziu um pouco as fatias do BlackBerry, dos iPhones e dos Windows Phone, da Microsoft.

"O crescimento mundial do *Android* está sendo impulsionado pela forte demanda por smartphones de baixo custo em todas as grandes regiões, como a China, Índia e África", disse Neil Mawston, diretor executivo da Strategy Analytics.

A Quaddro realizou sua 1ª Pesquisa do mercado profissional de mobilidade no Brasil, com o intuito de mapear tanto o perfil das empresas e programadores quanto dos clientes de projetos mobile.

Durante 3 meses foram coletadas informações de 150 profissionais que trabalham diretamente com produção mobile no Brasil.

2.3.1. LOCALIZAÇÃO

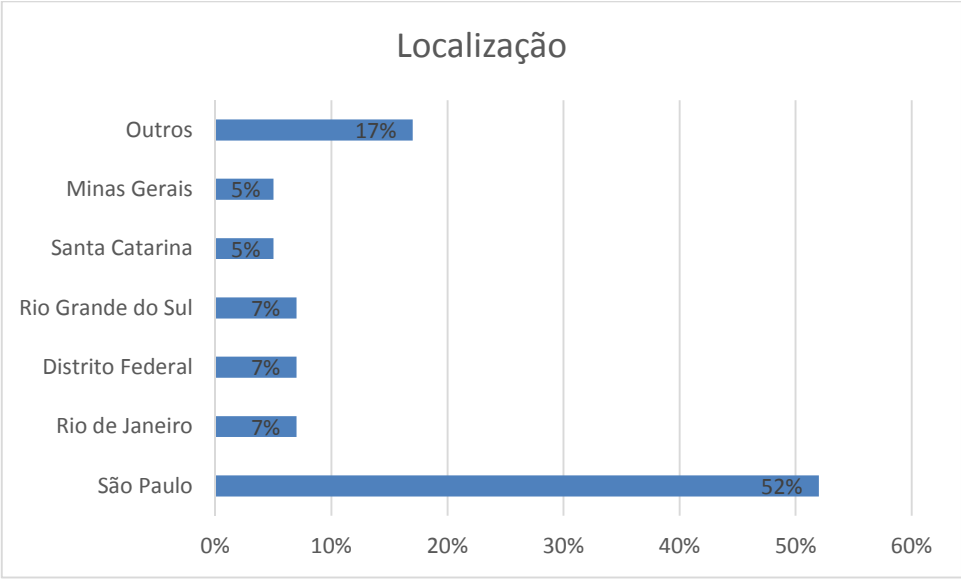

Figura 4: Gráfico de Localização

Fonte: Site Quaddro

Se analisarmos os resultados por região, fica claro o domínio absoluto das regiões Sul e Sudeste. Separando por estados os 4 maiores mercados são: São Paulo (52%), Rio de Janeiro (7%), Distrito Federal (7%) e Rio Grande do Sul (7%).

2.3.2. PAGAMENTO

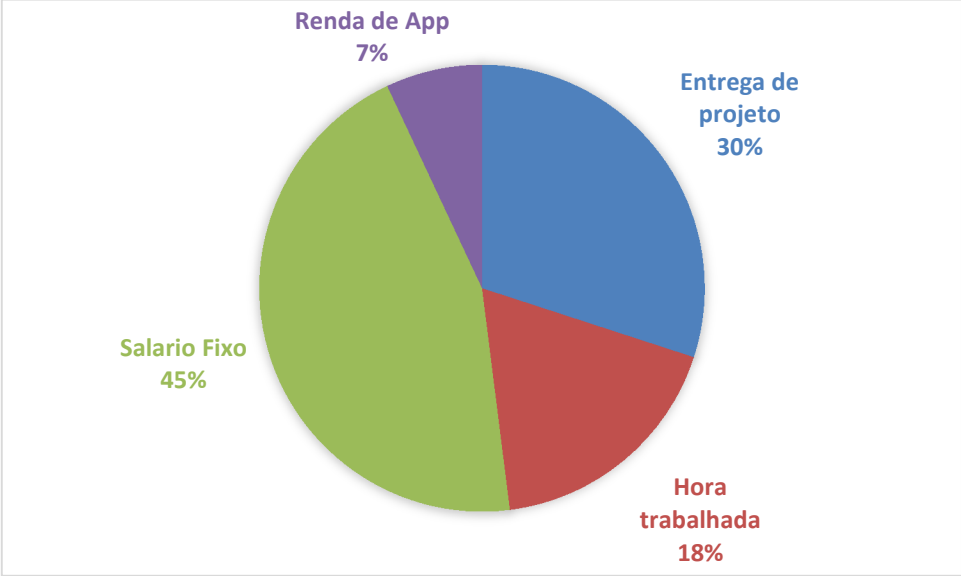

Figura 5: Gráfico de Valores

Fonte: Site Quaddro

Conforme figura acima 45% dos profissionais afirmaram que recebem salário fixo, enquanto 30% recebem por entrega de projeto. Outros 18% ganham por hora trabalhada e 7% obtêm sua renda diretamente dos aplicativos. Esse é um cenário bem heterogêneo com espaço para o salário fixo, mas também para as formas de receita para profissionais autônomos.

Desses profissionais autônomos que possuem receitas de seus próprios projetos, as proporções de receitas são:

41% > Recebem pela venda direta na App Store
41% > Recebem a partir de patrocínio de outras empresas
10% > Recebem a partir de publicidade
08% > Recebem a partir de vendas dentro do app (InApp Purchase)

Sobre o custo médio dos projetos da empresa os valores foram reveladores.

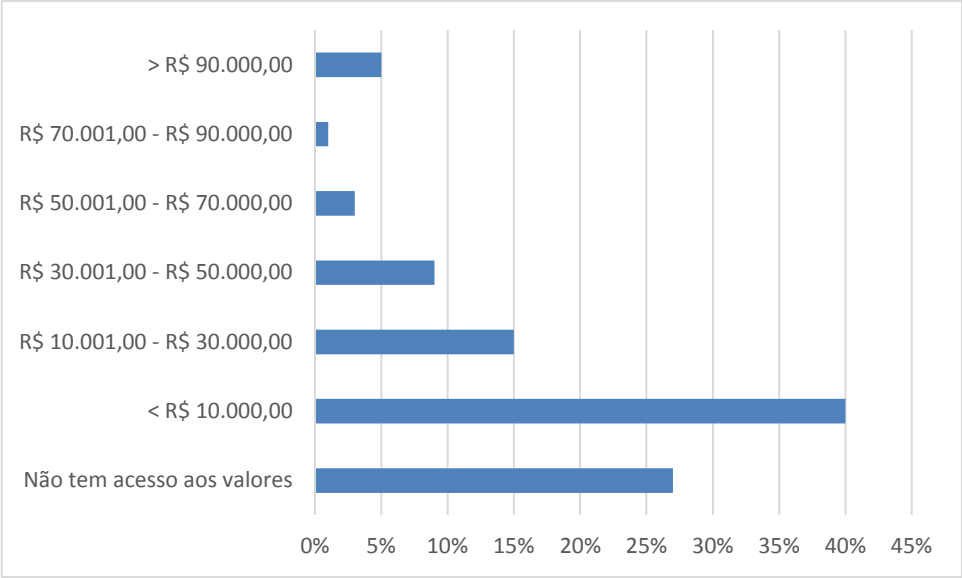

Figura 6: Gráfico de Salários

Fonte: Site Quaddro

Primeiramente é importante notar que 27% dos profissionais declararam que não tem acesso as informações de valores dos projetos nas empresas que trabalham, conforme verificado na figura acima. Daqueles que tem acesso a essa informação 40% disseram que a média de valores dos projetos de suas empresas é de até R$ 10.000. Esse é um valor baixo por toda a complexidade envolvida com a criação do aplicativo, mas pode ser explicado por projetos com recursos técnicos simples.

2.4. Mercado de Automação Comercial

Automação comercial é a aplicação de métodos e ferramentas para automatizar processos comerciais, isso é, mecanizar e agilizar processos manuais, alcançando total eficiência.

O processo de Automação Comercial no Brasil está ocorrendo em um ritmo muito acelerado e é inevitável. Primeiramente, deve-se a recessão econômica, que gera a necessidade reduzir custos, elevar o lucro e a produtividade. E também a necessidade do Órgão Administrador de Tributos Federais (Receita Federal) combater a Sonegação e garantir a Arrecadação devida.

A Associação Brasileira de Automação Comercial – EAN – define a automatização comercial como a informatização e a integração de todas as operações internas de uma empresa, bem como na integração desses processos como o mundo externo, compostos dos

fornecedores, governos, bancos, serviços de proteção ao crédito, operadores de cartão de crédito, entre outros, e, até mesmo com os consumidores. (TIERGARTEN 2007).

O desenvolvimento da automação, inicialmente, ocorre com a implantação dos equipamentos e a substituição dos procedimentos e rotinas manuais por procedimentos automatizados, até atingir a utilização de ferramentas que possibilitam um maior controle e uma melhor gestão dos negócios, gerando maior competitividade e rentabilidade. (Ângelo 1996).

3. METODOLOGIA

3.1. Diagramas

3.1.1. DIAGRAMA DE CASO DE USO

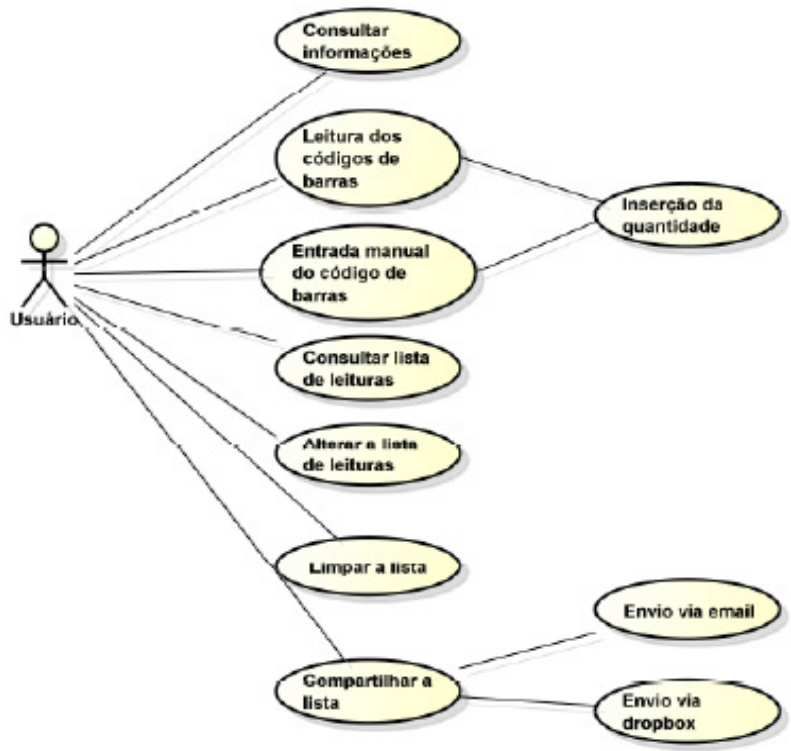

Figura 7: Diagrama de Caso de Uso

Fonte: Produção do próprio Autor

3.1.2. ESPECIFICAÇÃO DE CASO DE USO

- **Fazer contagem de estoque para atualizar o sistema ERP**

- **Objetivo**

Prover uma maneira para auxiliar na contagem e acerto do estoque com mais agilidade utilizando dispositivos móveis.

- **Tipo de Caso de Uso**

Caso de Uso Concreto.

- **Atores**

Nome Ator	Tipo	
	Primário	Secundário
Encarregado do estoque	X	
Gerente	X	

- **Pré-condições**

Todas as mercadorias precisam estar etiquetadas e conter código de barras.

- **Fluxo Principal**

  - Este caso de uso começa quando o ator deseja fazer a contagem do estoque.
  - O ator seleciona no menu principal a opção coletor de dados.
  - A aplicação apresenta uma tela com as opções Scan, Alterar, Leitura e Excluir.
  - O ator seleciona a opção Scan.
  - A aplicação abre a tela de leitura utilizando a câmera do dispositivo.

- O ator posiciona enquadrando o código de barras visualizando pela tela do dispositivo.
- A aplicação faz a leitura do código de barras e logo em seguida pede a quantidade do produto.
- O ator digita a quantidade do produto em estoque.
- A aplicação grava o código de barras e a quantidade no arquivo texto na memória do dispositivo e exibe o resultado em um campo de texto.
- O caso de uso termina.

- **Fluxos Alternativos**

- **Alterar**

O ator pode alterar a leitura mostrada na caixa de texto, tanto o código quanto a quantidade.

A aplicação retorna a tela para aguardando novos comandos para a contagem.

- **Carregar**

O ator seleciona a opção para carregar a contagem anterior, assim podendo continuar do ponto aonde foi parada a contagem.

A aplicação retorna no campo de texto todos os dados lidos no arquivo texto contendo os dados da contagem anterior.

- **Excluir**

O ator seleciona a opção excluir para limpar os dados coletados armazenados no arquivo texto para começar uma nova contagem.

A aplicação faz uma pergunta se realmente deve excluir o conteúdo do arquivo texto.

O ator confirma a exclusão

A aplicação limpa o arquivo texto e retorna para a tela aguardando nova contagem.

- **Informação**

O ator seleciona a opção informações para visualizar as explicações de como utilizar a aplicação.

- **Compartilhar**

O ator seleciona a opção compartilhar para transferir o arquivo texto para um computador desktop.

- **Manual**

O ator seleciona a opção manual para entrar com o código de barras e a quantidade digitando pelo teclado virtual do dispositivo móvel.

- **Fluxos de Exceção**

O ator não faz a leitura do código.

O sistema exibe mensagem relatando o fato.

O sistema retorna ao Passo P3 do Fluxo Principal.

O ator não entra com a quantidade do produto

A aplicação adiciona automaticamente a quantidade 1 para o produto.

O sistema retorna ao Passo P3 do Fluxo Principal.

O ator não consegue ler o código de barras utilizando a câmera do dispositivo móvel.

A aplicação disponibiliza a opção de entrada manual para o ator entrar com o código de barras e a quantidade manualmente.

- **Pós-condições**

Visualização dos códigos e quantidades dos produtos contados.

- **Requisitos Não-Funcionais**

Não possui.

- **Ponto de Extensão**

Não possui.

- **Critérios de Aceite**

A aplicação disponibilizou as opções do menu principal para o ator escolher.

A aplicação disponibilizou os botões.

A aplicação apresenta o conteúdo dos botões quando este é selecionado.

A aplicação gera o relatório dos códigos e quantidades contadas baseado nas leituras das etiquetas e nos dados de quantidade informados pelo ator.

A aplicação compartilha o arquivo texto quando o ator aciona o botão compartilhar.

A aplicação carrega os dados do arquivo texto quando selecionado o botão carregar.

A aplicação altera os dados do arquivo texto quando selecionado o botão alterar.

A aplicação exclui o conteúdo do arquivo texto quando selecionado a opção excluir

- **Frequência de Utilização**

A frequência de utilização será a critério do encarregado da contagem do estoque.

3.1.3. DIAGRAMA DE CLASSE

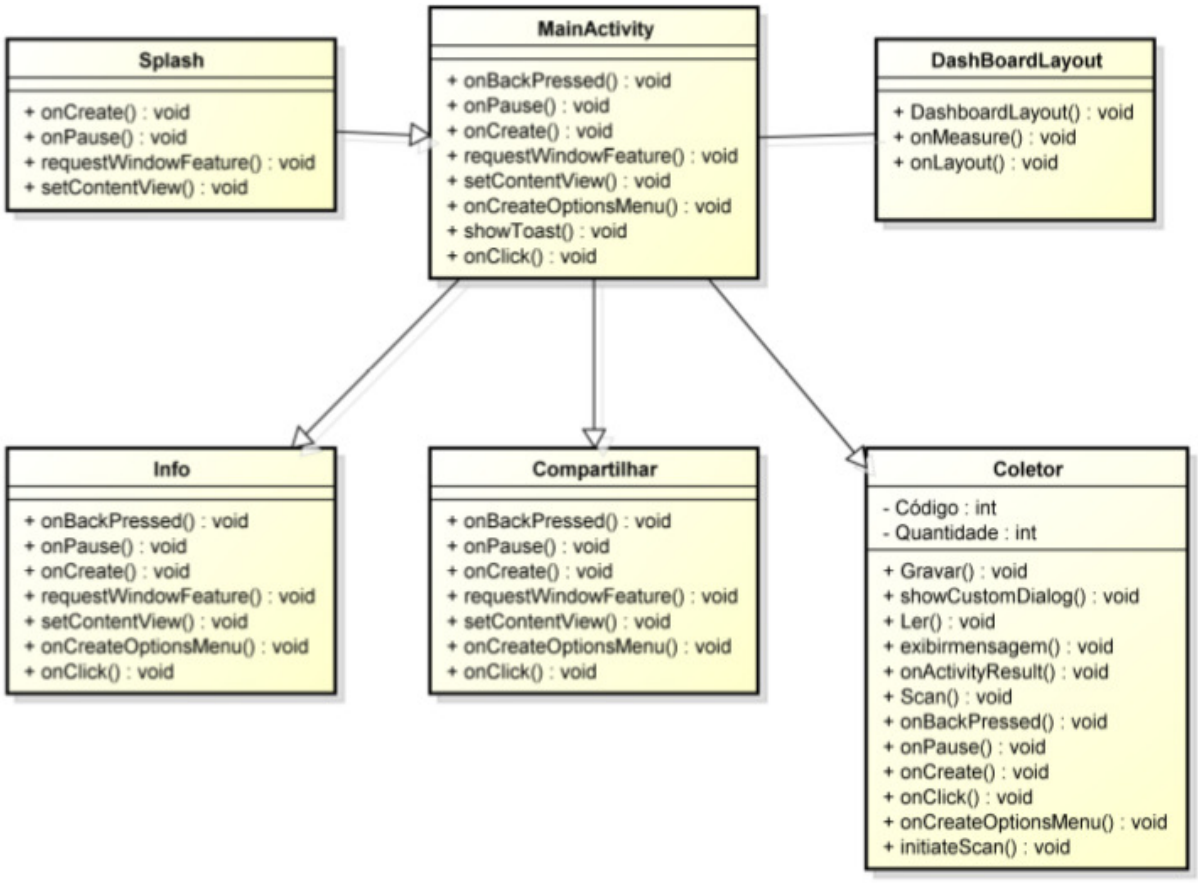

Figura 8: Diagrama de Classe

Fonte: Produção do próprio Autor

3.1.4. DIAGRAMA DE ATIVIDADES

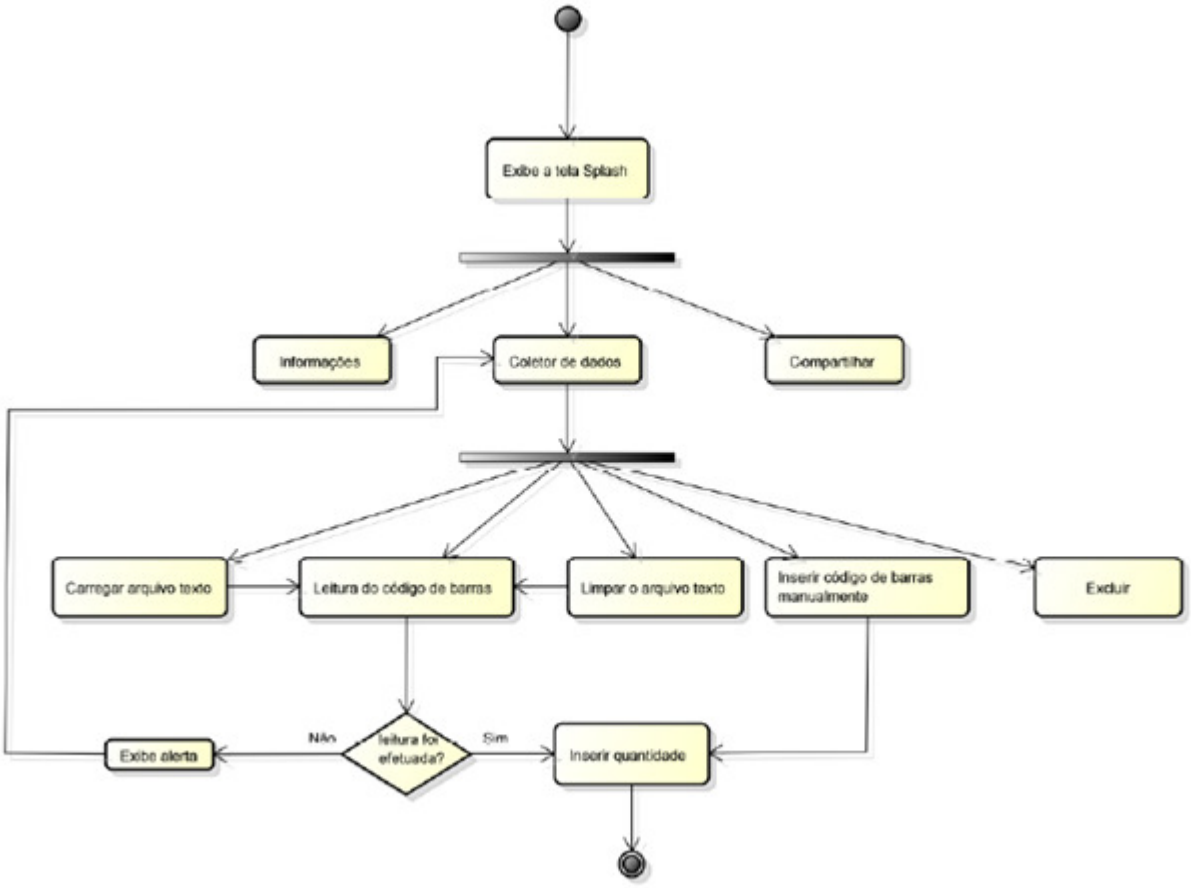

Figura 9: Diagrama de Atividades

Fonte: Produção do próprio Autor

3.2. Biblioteca Zxing

O *Android* é uma ferramenta que aceita várias opções na hora da programação para facilitar o trabalho do programador e enriquecer a experiência do usuário. O Zxing é um projeto livre de uma biblioteca para leitura de códigos de barras, ele possui toda a estrutura pronta para ser incorporada ao projeto principal, tornando fácil e rápido sua chamada e utilização.

A biblioteca é incluída inteiramente no projeto, assim não sendo necessário o usuário instalar programas de terceiros para efetuar a leitura e interpretação dos códigos de barra. Ainda possui um aplicativo nativo do projeto que serve exclusivamente para leitura de códigos, chamado Barcode Scanner.

O aplicativo Barcode Scanner, lê as informações do QR Code e também de códigos de barra convencionais, assim fornecendo informações ao usuário. Com o resultado da leitura você pode obter uma descrição do objeto em questão, preço, acesso à URL contida neles, contatos, entre outros.

A leitura é efetuada pela câmera traseira de um smartphone ou tablet com a versão *Android* 2.3.3 ou superior instalado, a informação é mantida em uma variável para ser posteriormente trabalhada pelo programador. São aceitos os formatos UPC-A, UPC-E, EAN-8, EAN-13, Code 39, Code 93, Code 128, ITF, Codabar, MSI, RSS-14 (todas as variantes), QR Code, Data Matrix, Axtec e PDF-417.

A biblioteca é disponibilizada sob os termos da Licença Apache 2.0 distribuído em arquivos com a extensão. apk, todo o projeto está em conformidade com os termos da licença, podendo ser utilizado gratuitamente para fins comerciais ou não.

Formatos suportados pelo Barcode Scaner e seus campos de utilização

1D produtos	**1D industrial**	**2D**
UPC-A	Code 39	QR Code
UPC-E	Code 93	Data Matrix
EAN-8	Code 128	Aztec (beta)
EAN-13	Codabar	PDF 417 (beta)
	ITF	
	RSS-14	
	RSS-Expanded	

Tabela 1: Formatos Barcode

Componentes da biblioteca

Módulos	**Descrição**
Core	A biblioteca de decodificação de imagens.
Javase	JavaSE específico código cliente.

Módulos	Descrição
Android	*Android* Barcode Scanner
*Android*test	Teste de aplicativos *Android*, ZXing Test
Android-integration	Suporta integração com Barcode Scanner via Intent
zxingorg	A fonte do projeto Zxing
zxing.appspot.com	O projeto por trás do gerador de código de barras baseado na web **zxing.appspot.com**

Tabela 2: Componentes da biblioteca

Módulos adicionais para outras linguagens de programação

Módulo	Descrição
cpp	C++ port
iphone	iPhone cliente
objc	Objective C port
jruby	JRuby invólucro
actionscript	Partial ActionScript port

Tabela 3: Módulos adicionais

Código aberto relacionado para terceiros

Módulo	Descrição
QZXing	Funções para o framework Qt.
ZXing.NET	Funções para Windows NET e C #, e relacionados.

Tabela 4: Códigos relacionados

3.3. Recursos Adicionais

Os dispositivos móveis são ricos em recursos tecnológico para serem explorados pelos programadores, assim tornando a interação com o usuário cada vez mais rica e com uma captação de dados enorme.

Na aplicação é utilizado como principal entrada de dados a câmera traseira dos dispositivos, com a manipulação da biblioteca Zxing a câmera é transformada em um leitor para códigos de barras, captando a imagem para ser lida pelo algoritmo, para uma boa captação da imagem é utilizado recursos do hardware para focalizar e deixar nítido o código alvo.

O aplicativo gera um arquivo texto que deve ser absorvido pelo programa ERP para alimentar os dados de estoque, para isso a transição entre a aplicação e o ERP é feita via internet utilizando principalmente o e-mail ou então o serviço *Dropbox*.

Para o armazenamento das informações captadas pela câmera é utilizando um arquivo de texto simples que é criado diretamente na memória interna do dispositivo, caso não exista a memória interna o arquivo é criado em um cartão de memória que esteja disponível.

4. RESULTADO E DISCUSSÕES

4.1. Organização Do Projeto

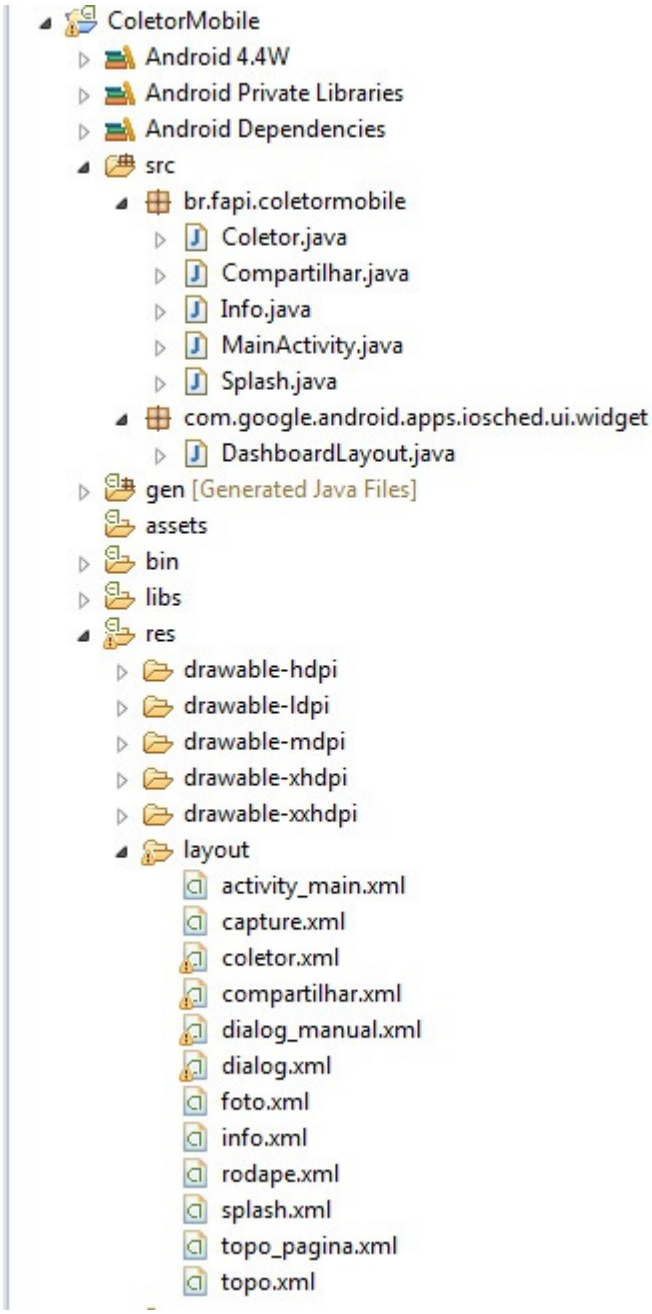

Figura 10: Organização do projeto – Print IDE

4.2. Classes

O aplicativo tem como base de suas funcionalidades seis classes para melhor organizar e disponibilizar as funções, deixando assim o código fonte mais legível e de fácil entendimento.

MainActivity – Classe principal do projeto, onde são disponibilizados os ícones para as aplicações. Uma Activity é basicamente uma classe gerenciadora de UI (Interface com o usuário). Todo aplicativo *Android* começa por uma Activity. Ou seja, quando uma aplicação *Android* é executada, na verdade é a sua Activity principal que é lançada.

OnCreate() é o método que ocorre somente quando seu form é criado, normalmente ao iniciar a aplicação ou quando se cria o mesmo dinamicamente, todas as instruções que estão dentro dele são iniciadas automaticamente quando o form é acionado.

OnPause() é o método que após uma ação do form ele deixa a aplicação aguardando até o retorno da ação.

RequestWindowFeature() método que retira o título superior do form.

setContentView() método que chama o layout do form para ser montado da tela.

OnCreateOptionsMenu() cria o menu padrão das aplicações *Android*.

ShowToast() método que exibe mensagens de aviso ao usuário na tela do dispositivo.

OnClick() método que é acionado quando é clicado no botão que se faz referência.

Info – Classe que chama o layout contendo as instruções de uso do programa.

OnCreate() todas as instruções são carregadas nesse método que é acionado na montagem do form.

Coletor – Classe que trata toda chamada e manipulação da leitura dos códigos e a criação e gravação do arquivo texto.

Gravar() é o método que verifica se já possui o arquivo texto na memória do celular, caso ainda não exista ele cria o arquivo e faz a gravação e atualização das leituras.

ShowCustomDialog() método que faz a pergunta ao usuário sobre a quantidade do produto após a leitura do código e grava a informação no arquivo texto.

Ler() método que faz a leitura do conteúdo do arquivo texto, caso o arquivo não exista ele exibe uma mensagem alertando o usuário.

Exibirmensagem() método que mostra uma mensagem de confirmação de gravação, exclusão ou leitura ao usuário.

OnActivityResult() método que trata o retorno dos dados captados pela câmera do dispositivo, verifica se foi feito uma leitura válida, caso a leitura esteja correta o método retorna os valores em uma variável para ser trabalhada pelo programa posteriormente.

Scan() método que faz a chamada e inicialização dos pacotes responsáveis pelo layout e leitura (BarcodeScanner/ Zxyng).

InitiateScan() método que é acionado junto com o método Scan(), faz o carregamento do layout mostrando um risco vermelho no meio do visor, que torna amigável a leitura para o usuário.

Compartilhar – Classe que faz o compartilhamento do arquivo texto gerado na leitura para ser incorporado pelo software de gestão comercial.

OnClick() método que é acionado pelo botão compartilhar, quando o método é inicializado faz a busca pelo arquivo texto na memória do celular e chama a Intent ACTION_SEND, que é responsável por abrir os programas instalados no dispositivo oferecendo o compartilhamento externo.

Splash – Classe que exibe a tela de abertura com o logotipo e faz as verificações necessárias para a aplicação iniciar corretamente.

OnCreate() método que inicializa todos os comandos dentro dele quando o form é criado. Na no método onCreate() da classe Splash temos a Thread que controla o tempo que a tela de Splash (apresentação) do programa vai ficar na tela e também verifica se o dispositivo está com internet disponível.

DashBoardLayout – Classe padronizada pelo google para organização e exibição de conteúdo.

OnMeasure() é uma peça crítica para o contrato de renderização do seu componente e o seu contêiner. O onMeasure() deve ser sobrescrito para eficientemente e corretamente reportar as medidas de seu contêiner para que a renderização seja perfeita na tela.

OnLayout() método que monta os componentes da dashboard na tela.

4.2.1. MAIN ACTIVITY

Classe principal do projeto, onde são disponibilizados os ícones para as aplicações. Uma Activity é basicamente uma classe gerenciadora de UI (Interface com o usuário). Todo aplicativo *Android* começa por uma Activity. Ou seja, quando uma aplicação *Android* é executada, na verdade é a sua Activity principal que é lançada.

- OnCreate() é o método que ocorre somente quando seu form é criado, normalmente ao iniciar a aplicação ou quando se cria o mesmo dinamicamente, todas as instruções que estão dentro dele são iniciadas automaticamente quando o form é acionado.
- OnPause() é o método que após uma ação do form ele deixa a aplicação aguardando até o retorno da ação.
- RequestWindowFeature() método que retira o título superior do form.
- setContentView() método que chama o layout do form para ser montado da tela.
- OnCreateOptionsMenu() cria o menu padrão das aplicações *Android*.
- ShowToast() método que exibe mensagens de aviso ao usuário na tela do dispositivo.
- OnClick() método que é acionado quando é clicado no botão que se faz referência.

4.2.2. INFO

Classe responsável por chamar o layout contendo as instruções de uso do programa.

- OnCreate() todas as instruções são carregadas nesse método que é acionado na montagem do form.

4.2.3. COLETOR

Classe que trata toda chamada e manipulação da leitura dos códigos e a criação e gravação do arquivo texto.

Gravar() é o método que verifica se já possui o arquivo texto na memória do celular, caso ainda não exista ele cria o arquivo e faz a gravação e atualização das leituras.

ShowCustomDialog() método que faz a pergunta ao usuário sobre a quantidade do produto após a leitura do código e grava a informação no arquivo texto.

Ler() método que faz a leitura do conteúdo do arquivo texto, caso o arquivo não exista ele exibe uma mensagem alertando o usuário.

Exibirmensagem() método que mostra uma mensagem de confirmação de gravação, exclusão ou leitura ao usuário.

OnActivityResult() método que trata o retorno dos dados captados pela câmera do dispositivo, verifica se foi feito uma leitura válida, caso a leitura esteja correta o método retorna os valores em uma variável para ser trabalhada pelo programa posteriormente.

Scan() método que faz a chamada e inicialização dos pacotes responsáveis pelo layout e leitura (BarcodeScanner/ Zxyng).

InitiateScan() método que é acionado junto com o método Scan(), faz o carregamento do layout mostrando um risco vermelho no meio do visor, que torna amigável a leitura para o usuário.

4.2.4. COMPARTILHAR

Classe que faz o compartilhamento do arquivo texto gerado na leitura para ser incorporado pelo software de gestão comercial.

OnClick() método que é acionado pelo botão compartilhar, quando o método é inicializado faz a busca pelo arquivo texto na memória do celular e chama a Intent ACTION_SEND, que é responsável por abrir os programas instalados no dispositivo oferecendo o compartilhamento externo.

4.2.5. SPLASH

Classe que exibe a tela de abertura com o logotipo e faz as verificações necessárias para a aplicação iniciar corretamente.

OnCreate() método que inicializa todos os comandos dentro dele quando o form é criado. Na no método onCreate() da classe Splash temos a Thread que controla o tempo que a tela de Splash (apresentação) do programa vai ficar na tela e também verifica se o dispositivo está com internet disponível.

4.2.6. DASHBOARD LAYOUT

Classe padronizada pelo google para organização e exibição de conteúdo.

OnMeasure() é uma peça crítica para o contrato de renderização do seu componente e o seu contêiner. O onMeasure() deve ser sobrescrito para eficientemente e corretamente reportar as medidas de seu contêiner para que a renderização seja perfeita na tela.

OnLayout() método que monta os componentes da dashboard na tela.

4.3. Interface do Aplicativo

Ao iniciar a aplicação, o usuário terá acesso à tela de boas-vindas (Splash), onde poderá acionar o menu e acessar a tela “Main Activity”.

Figura 11: Tela: Splash

Tela MainActivity, menu principal da aplicação, onde o usuário pode interagir com as opções: informações, coletor de dados, Enviar p/ PC e sair.

Figura 12: Tela: MainActivy

Tela informações, são descritas instruções de como proceder para utilizar o coletor mobile para contagem do estoque, explica todas as funções existentes na tela coletor de dados.

Figura 13: Tela: Informações

Tela coletor de dados, são disponibilizadas as opções scan, alterar, leitura e excluir. A opção scan prepara o dispositivo mobile para a leitura do código de barras. Opção alterar guarda no arquivo texto as alterações que são feitas pelo usuário no campo texto. Opção leitura carrega todo o conteudo do arquivo texto no campo retorno da leitura. Opção excluir exclui todo o conteudo do arquivo texto.

Figura 14: Tela: Coleta de Dados

Tela de leitura do aplicativo, ao enquadrar o código de barras no retângulo com a faixa vermelha ao centro, a leitura é feita automaticamente.

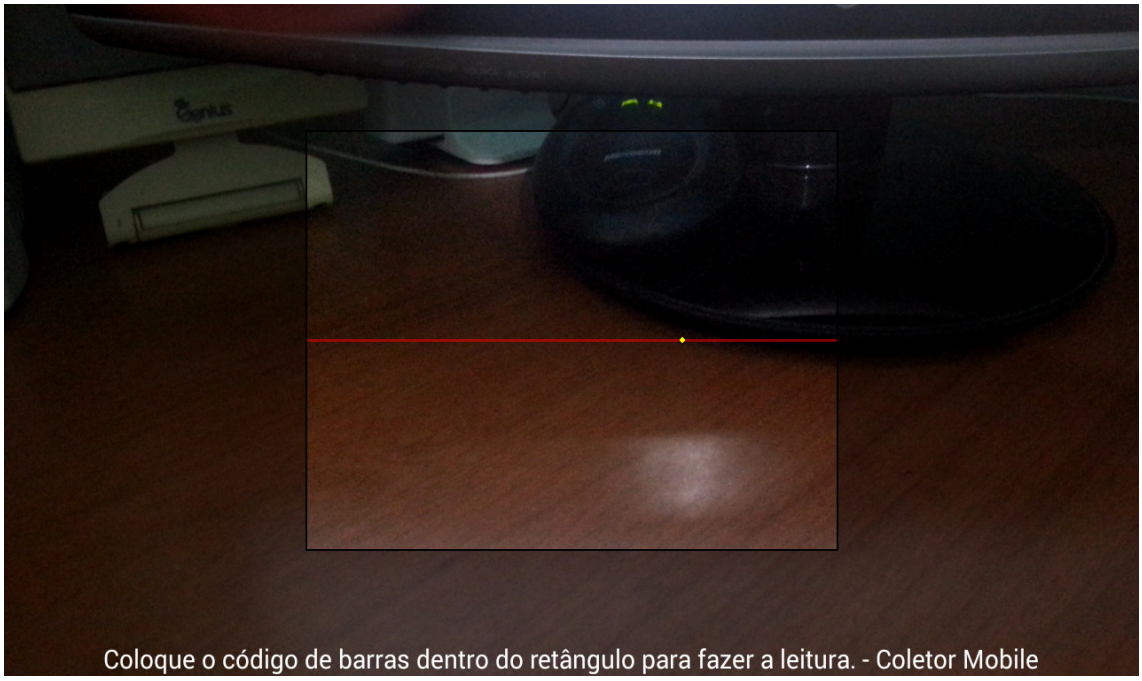

Figura 15: Tela: Scanner

Tela abaixo mostra o momento que o código é captado pela aplicação. Logo após será solicitado a quantidade relacionado ao codigo capturado.

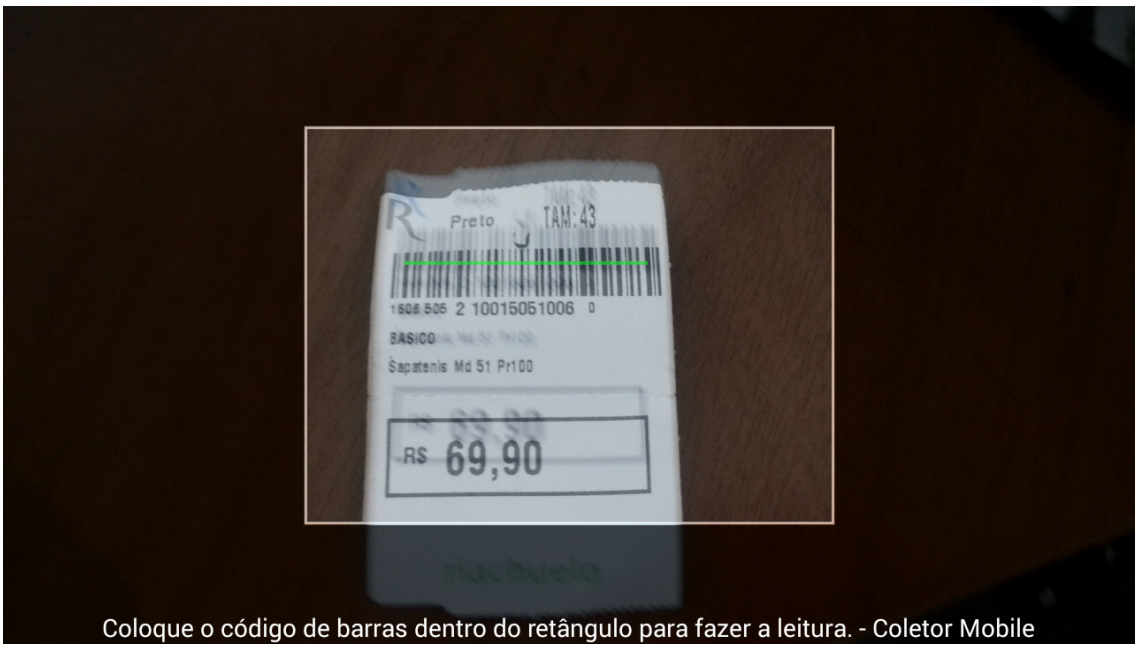

Figura 16: Tela: Exemplo de leitura

Tela transferência do arquivo, ao clicar no botão compartilhar são carregadas as opções de compartilhamento pré-configurados no celular para enviar o arquivo texto ao computador, alimentando o software ERP.

Figura 17: Tela: Compartilhar

Tela seguinte mostra os programas que estão instalados no dispositivo que podem ser utilizados para escolher o local de armazenamento do arquivo texto gerado pela aplicação.

Figura 18: Escolha do programa para compartilhar

4.4. Trabalhos Futuros

Os resultados obtidos no presente estudo sugerem diversos trabalhos a serem realizados em um futuro próximo. A implementação utilizando um banco de dados online para substituir o arquivo texto será de grande valia, será possível captar mais dados sobre o produto, tais como a foto, modelo, cor, etc. Complementando ainda mais a abstração de dados para alimentar o programa ERP.

Uma outra implementação válida será a inserção de dados diretamente no banco de dados do ERP, tendo assim uma atualização em tempo real, sem a necessidade de envio e leitura dos dados para atualizar o estoque. Com essa melhoria o gerenciamento do trabalho de contagem de estoque será muito facilitado, pois o encarregado poderá consultar gráficos e planilhas para acompanhar todo o processo.

Outro estudo interessante é abrir o programa para outras plataformas, tais como IOS e Windows Phone, aumentando cada vez mais as possibilidades de utilização para o cliente, não limitando somente a dispositivos *Android*, tornando a aplicação cada vez mais acessível em todos os ambientes de trabalho.

5. CONCLUSÃO

Com o grande crescimento do ERP no Brasil e a busca das empresas por excelência na qualidade com baixo custo, abre-se um grande mercado para a exploração das tecnologias dos dispositivos móveis para área comercial e industrial. Com o conteúdo estudado é sabido afirmar que os sistemas de informações estão cada vez mais dentro de nosso dia a dia, tornando-se ferramentas poderosas para auxilio em quase tudo que fazemos, como trabalhar, estudar e para o laser. E comparando com indicadores podemos afirmar que estamos apenas na "ponta do iceberg", a tecnologia mobile chegou para ficar, e cada vez mais se faz presente.

O interesse pelo tema escolhido se deve pela grande necessidade que as empresas encontram hoje em dia para controlar seus grandes estoques para serem competitivas no mercado, mas que para isso precisam se dispor de um grande investimento em maquinas coletoras de dados. Com a aplicação desenvolvida cria-se a possibilidade de qualquer empresa de pequeno ou médio porte ter acesso a essa solução que até então é restrita a grandes empresas devido ao seu alto custo de aquisição.

Os dispositivos móveis têm como grandes vantagens a disponibilidade no mercado, facilidade de operação e seu baixo custo, assim dando a possibilidade de empresas de pequeno e médio portes também utilizarem a tecnologia dos coletores de dados profissionais sem grande investimento.

Em vista do crescimento da utilização em grande demanda de dispositivos mobiles verifica-se uma grande porta para futuros investimentos aliando automação comercial e dispositivos como tablets, smatphones, smartwatches, etc, como acompanhamento total da loja pelo gerente mesmo que fora do estabelecimento, relatórios diversos direcionado aos controladores do negócio, entre outros. Concluímos assim que a automação comercial utilizando dispositivos móveis ainda há muito a ser explorada, com a certeza de retorno de bons frutos para o idealizador, o investidor e para o cliente final.

6. REFERENCIAS

ANGELO, C.F.S. **Marketing no varejo: gestão operacional**. São Paulo: Atlas, 1996

ANGELO, C.F.S. **Varejo: modernização e perspectivas**. São Paulo: FEA/USP, 1995

ANGELO, C.F.S & SILVEIRA, J A. **Varejo competitivo**. Vol 4 São Paulo: Atlas, 2000

AQUINO, J. **Plataformas de desenvolvimento para dispositivos móveis**. 2007.

BARROS, T. **Cinco anos de Android: relembre a história e todas as versões do sistema,** 2013**.** Disponível em: http://www.techtudo.com.br/noticias/noticia/2013/09/cinco-anos-de-android-relembre-historia-e-todas-versoes-do-sistema.html. Acesso em: 23 nov 2014

BRANDAO, M. **Mercado de ERP segue em ascensão no Brasil.** 2014**.** Disponível em: <http://b2bmagazine.consumidormoderno.uol.com.br/ index.php/negocios/item/3387-mercado-de-erp-segue-em-ascensao-no-brasil>. Acesso em: 12 nov 2014.

CECCATTO, J. T. **Aplicativo de Comparação e Divulgação de Produtos em Tempo Real para Dispositivos Móveis.** Santa Catarina, 2012. p. 46.

CIPOLI, P. **O que é Java, JRE, JVM e JDK?** Disponível em: <http://corporate.canaltech.com.br/o-que-e/programacao/O-que-e-Java-JRE-JVM-e-JDK>. Acesso em 23 nov 2014.

DIMARZIO, J. F.; ***Android*: A programmer's Guide, New York**. McGrawHill E-Book, 2008.

ECLIPSE. **About *Eclipse***. Disponível em <http://www.ec*lipse*.org>. Acesso em 01 out 2014

FURGERI, S. **Java 2:** ensino didático: desenvolvendo e implementando aplicações. São Paulo: Érica, 2002. p.112

GOMES, R. et al. **Sistema Operacional Android.** Rio de Janeiro, 2012. p. 7.

GUIMARAES, G. **A história do sistema operacional Android,** 2013. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/jornal/agosto2013/materias/historia_da_computacao.html>. Acesso em: 23 nov 2014.

JAQUES, A. P. **Programação Básica em Java.** Rio Grande do Sul, 2007. p.14

JUNNARKAR, S. **IBM makes $40 million open-source offer.2001.** Disponível em <http://news.cnet.com/IBM-makes-40-million-open-source-offer/2100-1001_3-275388.html>. Acesso em 01 out 2014.

LAUREANO, A. M.; MAZIERO, A. C. **Virtualização: Conceitos e Aplicações em Segurança.** Paraná, 2008. p. 42.

LECHETA, R. R. **Google *Android*: Aprenda a criar aplicações para dispositivos móveis com *Android* SDK**. 2° Edição. São Paulo. Novatec Editora, 2010.

LUCIEL, T. **Dossiê Android: 3 anos de vida.** 2010. Disponível em: <http://mobilidade.fm/geral/2010/11/historia-do-android/>. Acesso em: 23 nov 2014.

MONTEIRO, J. B. **Google *Android*** – Crie aplicações para celulares e tablets. 1. ed. São Paulo, 2013.

MOREIRA, D. **Oracle compra Sun por US$ 7,4 bilhões**. 2009. Disponível em: <http://info.abril.com.br/noticias/negocios/oracle-compra-sun-por-us-7-4-bilhoes-20042009-5.shl>. Acesso em 01 out 2014.

MURARI, M.L. **Desenvolvimento de uma ferramenta computacional de apoio ao ensino de sistemas eletrônicos digitais**. São Paulo. 2008. p. 24

NETBEANS. **About *NetBeans***. Disponível em <https://*NetBeans*.org/about/index.html>. Acesso em 01 out 2014.

OHA et al. **Building a Better phone for consumers**. 2011. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 15 out 2014.

QUADDRO. **Resultados da primeira pesquisa Quaddro do mercado mobile no Brasil**. Disponível em: <http://www.quaddro.com.br/1-pesquisa-mobile-quaddro>. Acesso em: 01 out 2014.

RIBEIRO, J. A. **Diferenciais da linguagem Java.** Campinas: IPEP, 2003. p. 93

STREY, M**. Mobile World – Utilizando ZXING Para Ler Códigos de Barras**. 2012. Disponível em:<http://maicon.strey.nom.br/blog/2012/04/27/utilizando-zxing-para-ler-codigo-de-barras/>. Acesso em: 01 out. 2014.

STREY, M**. Documentação do projeto Zxing**. 2012. Disponível em: <http://zxing.github.io/zxing/apidocs/>. Acesso em: 01 out. 2014.

TIERGARTEN, M. **A automação comercial como fator de eficiência e produtividade para o varejo brasileiro**. Rev Aut Com, 29 ed, n 106. São Paulo, 2007, p 13-17

VELLOSO, C. F. **Informática Conceitos Básicos.** Rio de Janeiro, 2011. p. 67