UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

AUTOMATIZAÇÃO DE BACKUP DOS CONTATOS DO SISTEMA ANDROID: UMA FERRAMENTA PARA GARANTIR A INTEGRIDADE DOS CONTATOS

FLAVIO DE SOUSA OLIVEIRA

PICOS - PI
2015

FLAVIO DE SOUSA OLIVEIRA

AUTOMATIZAÇÃO DE BACKUP DOS CONTATOS DO SISTEMA ANDROID: UMA FERRAMENTA PARA GARANTIR A INTEGRIDADE DOS CONTATOS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação do Professor Especialista Leonardo Pereira de Sousa.

PICOS - PI

2015

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

O482a Oliveira, Flávio de Sousa
Automatização de backup dos contatos do sistema android: uma ferramenta para garantir a integridade dos contatos / Flávio de Sousa . – 2015.
CD-ROM : il.; 4 ¾ pol. (54 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2015.

Orientador(A): Prof. Esp. Leonardo Pereira de Sousa

1. Dispositivo Móvel. 2. Sistema Operacional Android. 3. Aplicativo Sincronize. I. Título.

CDD 005

AUTOMAÇÃO DE *BACKUP* DOS CONTATOS DO SISTEMA *ANDROID*: UMA FERRAMENTA PARA GARANTIR A INTEGRIDADE DOS CONTATOS

FLÁVIO DE SOUSA OLIVEIRA

Monografia APROVADA como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 29 de JUNHO de 2015

Leonardo Pereira de Sousa
Prof. Esp. Leonardo Pereira de Sousa
Orientador

Allan Jheyson Ramos Gonçalves
Prof. Esp. Allan Jheyson Ramos Gonçalves
Membro

Frank César Lopes Véras
Prof. Me. Frank César Lopes Véras
Membro

Dedico este trabalho a minha família, em especial aos meus pais, por sempre acreditarem nos meus sonhos e me apoiarem durante todo este percurso.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus que sempre esteve a frende deste sonho, pois sem Ele jamais conseguiria realiza-lo.

A minha família pelo incentivo, carinho e confiança. Em especial a minha mãe Francisca pela dedicação, sempre acreditando em meu potencial, e ao meu irmão Valdery que me incentivou, demonstrando que tudo pode ser conquistado, independente de todas as dificuldades, é necessário ter coragem e sempre seguir em frente.

Ao meu orientador, Professor Especialista Leonardo Pereira de Sousa que sempre esteve disposto a ajudar, levantando hipóteses quanto ao destino deste trabalho, sendo paciente e compreensivo diante as dificuldades, construindo minha aprendizagem através de dicas preciosas.

Aos meus amigos que me apoiaram diante todas as dificuldades, estando ao meu lado e utilizando palavras de conforto e incentivo, nos momentos difíceis ajudaram-me e nos momentos alegres dividiram sorrisos.

A minha namorada Hosana Tenório pelo incentivo, por acreditar em meus sonhos e pela paciência, pois sempre esteve ao meu lado.

A todos os professores do curso de Sistemas de Informação. Em especial a professora Patrícia Medyna Lauritzen de Lucena Drumond, que sempre esteve disposta a ajudar, agradeço também pela sua dedicação com que coordena o curso de Sistema de Informação.

A todos vocês muito obrigado!

“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes”

Marthin Luther King

RESUMO

A tecnologia da informação e comunicação está transformando o mundo e maximizando as diversas possibilidades seja nos estudos, no trabalho ou na vida cotidiana. Os dispositivos móveis são uma evolução deste cenário tecnológico em que se está cada vez mais conectado ao mundo, e por consequência, às outras pessoas. Esses dispositivos armazenam uma grande quantidade de informação, entre elas a dos contatos do usuário, que são de grande importância quando se pensa no mundo conectado, a era da informação. A utilização de meios para manter os dados do usuário a salvo é benéfica, tendo em vista, que o usuário não deseja perder seus contatos. Essa situação é vista com bastante constrangimento por alguns, dessa maneira, salvar os contatos em ambientes virtuais ou em nuvem torna-se uma boa opção ao usuário. Este trabalho apresenta o desenvolvimento de um aplicativo para dispositivos móveis que possui o sistema operacional *Android*, visando salvar os contatos do usuário em um ambiente em nuvem. Assim, permitindo a integridade dos contatos do usuário, para quando houver necessidade de requisitá-los. Durante seu desenvolvimento foram utilizados conhecimento sobre a modelagem *UML*, linguagem de programação *Java*, *IDE Android Strudio*, Máquina virtual Android.

Palavras-chave: Dispositivos, *Android*, Sistema Operacional, *Java.*

ABSTRACT

The information technology and communication are transforming the world and extending the various possibilities to optimize human existence, whether at school, at work or in day by day. Mobile devices are an evolution of this technological scenario, where people are every time more connected to the world and consequently to other people. These mobile devices store a lot of information, for example store the user's contacts, which are very relevant in considering the connected world and the era of information. The means used to keep user's data saved are beneficial, because the user doesn't want to lose your contacts, which could mean an embarrassing situation. Then, save the contacts in virtual environments or cloud becomes a good option to the user. This paper presents the development of an application for mobile devices that has the Android operating system, which aims to save the user's contacts in an virtual environment like the cloud. In this way, the integrity of the user's contacts will be possible, for when they are needed to access them. During the development of this paper were used knowledge of UML modeling, Java programming language, IDE Studio Android, Android virtual machine.

Keywords: Technological resources (Devices). Android. OS. Java.

LISTA DE ILUSTRAÇÕES

Figura 1: Distribuição das versões do Android nos dispositivos.....24
Figura 2: Arquitetura da Plataforma Android.....27
Figura 3: Diagramas definidos pela UML.....38
Figura 04: Diagrama de Caso de Uso..... 44
Figura 5: Atividade Definição Diretório.....46
Figura 6: Atividade Sincronizar.....47
Figura 7: Tela Inicial.....48
Figura 8: Tela de Configuração..... 49
Figura 9: tela ligar/desligar sincronização..... 49
Figura 10: Tela tipo de Conexão..... 50

LISTA DE QUADROS

Quadro 01: Requisitos Funcionais.....42
Quadro 02: Requisitos Não-Funcionais.....42

LISTA DE ABREVIATURAS E SIGLAS

ADT	*Android Development Tools*
ANATEL	Agência Nacional de Telecomunicações
API	*Application Programming Interface*
APK	*Android Package File*
AVD	*Android Virtual Device*
DVM	*Dalvik Virtual Machine*
IDC	*InternationalDataCorporation*
IDE	*Integrated Development Environment*
JDK	*Java Development Kit*
JVM	*Java Virtual Machine*
NDK	*Native Development Kit*
OEMs	*Original Equipment anufacturer*
OHA	*Open Handlest Alliance*
SDK	*Software Development Kit*
SO	Sistema Operacional
UML	*Unified Modeling Language*

SUMÁRIO

1. INTRODUÇÃO 14
1.1 Objetivos 15
1.2 Organização do Trabalho 16
2. REFERENCIAL TEÓRICO 17
2.1 Dispositivos Móveis 17
2.2 Sistema Operacional Android 19
2.2.1 Versões do Android 23
2.2.2 Arquitetura 26
2.2.3 Distribuindo um Aplicativo 29
2.3 Linguagens de Programação para *Android* 31
2.3.1 Linguagem Java 32
2.4 Cloud-Computing 35
2.5. Processo De Engenharia De Software 36
2.5.1 UML – Linguagem de Modelagem Unificada 37
3. APLICATIVO SINCRONIZE 41
3.1 Ferramentas utilizadas 41
3.3 Requisitos de software 42
3.3 Diagramas UML 43
4. RESULTADOS E DISCUSSÕES 48
5. CONSIDERAÇÕES FINAIS 51
REFERÊNCIAS 52

1. INTRODUÇÃO

Nas últimas décadas a sociedade perpassa por uma constante revolução tecnológica, de forma que é cada vez mais presente o uso dos computadores, seja em empresas de diversos segmentos, seja no cotidiano das pessoas. Nesse sentido, a automatização da sociedade tem feito com que a tecnologia avance de maneira acelerada e constante.

No século XX os computadores eram privilégio de poucos, atualmente esse mecanismo tecnológico encontra-se cada vez mais presente na vivência das pessoas, apesar de muitos infelizmente ainda não terem acesso a essa ferramenta, isso por alguns fatores como: financeiro e educacional.

Apesar disso, nota-se que a busca por mobilidade acaba sendo essencial, principalmente no que diz respeito ao armazenamento de arquivos pessoais em vários locais, como no caso da telefonia celular, de maneira que com o passar dos anos revolucionou a vida das pessoas, uma vez que possibilita a concentração de diversas funcionalidades e tipos de arquivos.

Os celulares guardam uma grande quantidade de informações como: fotos, contatos, tarefas, *e-mail* e servem até mesmo para realização de pequenos trabalhos. Todavia percebe-se a necessidade de estar conectado, seja para desenvolver atividades relacionadas ao trabalho, ou pessoais:

> Os objetos digitais podem assumir os mais variados tipos, retratando grande parte da nova vida pessoal e profissional, e representam toda informação digital presente em dispositivos eletrônicos como os computadores, telemóveis, smartphones, tablets, entre outros. Esta informação digital é extremamente importante para as pessoas que utilizam estes dispositivos (SOARES; PEREIRA; MARTINS, 2012, p. 76).

A ANATEL (Agência Nacional de Telecomunicações) divulgou que em 2010 o Brasil passou a ter mais de um celular por habitante. A mesma ainda informou que no mês de junho de 2014 já existiam no Brasil mais de 275,71 milhões de linhas telefônicas ativas, assim cerca de 136,06 por cada 100 habitantes.

Para suprir a necessidade da utilização de arquivos em diferentes dispositivos um novo conceito surgiu e continua a crescer, chamado *Cloud Computing*. Sendo uma abordagem em que os arquivos são armazenados em um servidor *web*, disponível ao usuário, onde os mesmos podem acessar, desde que os dispositivos estejam conectados à *internet*.

De acordo com Brian Hayes (2009 apud OLIVEIRA; MOZZAQUATRO, 2011) o uso do termo *Cloud Computing* é caracterizado pela utilização dos servidores físicos e

virtuais “Denominando-se nuvem” para prover espaço computacional. A preservação de arquivos em nuvem é cada vez mais utilizada, pois é levado em consideração a integridade, disponibilização e segurança dos arquivos armazenados em tais ambientes.

A preservação de arquivos importantes nos *smartphones* e *tablets* pode ser considerado motivo de preocupação para seus usuários, doravante a qualquer momento os mesmos perderem suas informações, por danos ao dispositivos, perdas e furtos.

Segundo Silva (2012), o Brasil é o segundo país com o maior número de celulares roubados no mundo perdendo somente para a Índia. Sendo que 25% da população brasileira declaram já terem seus aparelhos telefônicos roubados. Nesse caso, um número muito acima da média mundial que é de 11%.

Atualmente o mercado de *smartphones* é bastante segmentado, existindo diversas empresas que atendem desde usuários menos exigentes, até consumidores que detém o conhecimento sobre esse universo da tecnologia da informação.

De acordo com Monteiro (2012), a empresa *Google* lançou no mercado seu Sistema *Android* que foi adquirido em 2005 e anteriormente desenvolvido pela empresa *Android Inc*. Com seu surgimento, a *Google* passou a dominar este setor de dispositivos, sendo referência para todos os aparelhos com esta proposta.

Os Sistemas Operacionais *Mobiles* existentes no mercado, têm por objetivo proporcionar o acesso aos diferentes conteúdos interativos, passando dessa maneira a incorporar funções de outros aparelhos tecnológicos e até mesmo dos computadores. Dessa forma, diversas vantagens foram garantidas a seus usuários, principalmente no que se refere, à mobilidade de tais dispositivos.

Nessa contextualização, a aplicação que será desenvolvida, visa não somente o *backup* dos contatos nos aparelhos telefônicos, mas a automatização desse processo, pois considera-se o mesmo de suma relevância na utilização dos dispositivos móveis, aos quais são equipados com Sistemas Operacional *Android*.

1.1 Objetivos

O presente trabalho tem como objetivo o desenvolvimento de uma aplicação para dispositivos móveis, exclusivamente para a plataforma *Android*, sendo a mais difundida no mercado de *Smartphones* e *Tablets*. A aplicação que será desenvolvida irá realizar o *backup* automático dos arquivos de contatos quando estiver instalado nos dispositivos do usuário, armazenando em ambientes como *Onedrive* e *Googledrive*. Assim, o usuário não precisará

realizar esta tarefa manualmente, realizando o processo de forma simples e segura. Os contatos estarão disponíveis na nuvem de acordo com a escolha do usuário para que possa restaurar seus contatos quando necessário.

1.2 Organização do Trabalho

Após a introdução que relatou sobre o problema que se pretende resolver e o objetivo do sistema serão apresentados os próximos capítulos, que estão organizados da seguinte forma:

- Capítulo 2 – Referencial teórico dividido nas seguintes subseções: Dispositivos móveis, Sistema Operacional *Android*, linguagem de programação para *Android*, *cloud-computing*, processo de engenharia de software. Este capitulo oferece todo o embasamento teórico deste trabalho.
- Capítulo 3 – Aplicação Sincronize: Será mostrada a análise de requisitos, os diagramas produzidos para o entendimento do problema e o desenvolvimento do sistema.
- Capítulo 4 – Resultados e Discussões: São mostrados os usuários do sistema, como esses se relacionam e as principais funcionalidades que o sistema apresenta.
- Capítulo 5 - Considerações Finais: Apresenta-se a conclusão do trabalho e indicações de trabalhos futuros.

2. REFERENCIAL TEÓRICO

2.1 Dispositivos móveis

A computação móvel é citada no “uso de dispositivos móveis que ofereçam a capacidade de realizar um conjunto de funções de aplicações, além de serem capazes de conectar-se, obter dados e fornecê-los a outros usuários, aplicações e sistemas” (SANTOS, 2012, p.17). Assim, pode-se aferir diversas peculiaridades dos dispositivos móveis.

Os dispositivos móveis são caracterizados pela sua mobilidade, usabilidade, funcionalidade, conectividade, dessa forma oferecendo métodos de entrada e saída de dados, como um teclado e uma tela. São utilizados para diversos fins, mas nos dias atuais seu principal foco é a interatividade na rede mundial de computadores:

> A demanda de utilização da internet, em tempo integral, os dispositivos usados para esse fim têm como premissa básica possuir um tamanho reduzido, a fim de serem facilmente transportados e utilizados em qualquer local e tempo, realizando todas operações necessárias. Neste contexto, o termo mobilidade está alinhado a descrição de portabilidade, a qual define que computadores devem possuir dimensões reduzidas, ao ponto de serem facilmente transportados e manuseados (ROTONDO *et. al.*, 2014, p.02).

Os dispositivos móveis foram um dos instrumentos responsáveis pela transformação da forma de comunicar-se, possibilitando ao usuário estar sempre conectado ao mundo.

Chainho (2014, p.18) considera que esses dispositivos trouxeram: “Um avanço na tecnologia que colocou a capacidade de processamento de um computador e a capacidade de comunicar de um telefone num dispositivo de dimensões reduzidas”. Essa revolução tecnológica aproxima cada vez mais a capacidade de computadores pessoais da capacidade de dispositivos menores.

Com a crescente evolução da tecnologia, os indivíduos têm a necessidade de se adaptar a mesma, assimilando dessa forma novas maneiras de se comunicar e, consequentemente, de manipular novos aparelhos. No início do século XXI, a mobilidade passou a ter um papel preponderante diante das novas tecnologias, pois os dispositivos que oferecerem mais recursos ao consumidor, fazem com que estes sejam atraentes:

> O desenvolvimento da computação móvel e das novas tecnologias sem fio (*laptops, palms*, celulares) estabelece, no começo do século XXI, a passagem do acesso por ponto de presença (internet fixa por cabos), ao ambiente generalizado de conexão (internet móvel sem fio, telefones celulares, redes *bluetooth*e etiquetas de radiofrequência, RFID), que envolvem o usuário, em plena mobilidade (LEMOS, 2007, p. 08).

Recentemente os avanços ocorridos na área de telecomunicações fazem com que as aquisições de dispositivos móveis sejam cada vez mais crescentes, por parte dos consumidores (CARNEIRO, ROMAN, FAGUNDEZ, 2014), fazendo com que essa tecnologia alcance maior *status* a cada dia.

O IDC (*International Data Corporation*) divulgou que em 2013 foram vendidos cerca de 1 bilhão de *smartphones*, destes 78,6% possuíam Sistema Operacional *Android* (CAMPOS, KULESZA, & COELHO, 2014). Neste cenário, ainda existem diversos outros Sistemas Operacionais, que têm espaço no mercado de dispositivos móveis, sendo que a maioria além de contar com o Android, baseia-se no *iOS* da *Apple*, *Windows Phone* da *Microsoft* e *BlackBerry* (CHAINHO, 2014).

A tecnologia móvel vem auxiliando cada vez mais uma nova força de trabalho, onde o mundo dos negócios está conectado e a informação é sempre repassada rapidamente, neste sentido pode-se dizer que: “Esses aparelhos não só nos auxiliam para a eliminação do papel nos processos comerciais, como também podem nos ajudar no gerenciamento de compromissos e contatos” (SIQUEIRA, 2005 *apud* BOTTENTUIT JUNIOR, 2012).

Os primeiros celulares tinham a capacidade de armazenamento e processamento limitado, ao passar dos anos, esses fatores foram maximizados. Chainho (2014) considera que um dos maiores responsáveis pela ascensão dos smartphones foi o *iPhone* da *Apple*, uma vez que avançou a capacidade dos dispositivos para além da comunicação, criando assim um dos melhores da atualidade, se tornando referência para outros sistemas operacionais.

As funcionalidades dos dispositivos foram estendidas em todos os aspectos, seja em relação ao armazenamento ou a configurações mais robustas, assim se aproximando da capacidade dos computadores de hoje.

Abreu (2005) avalia que este cenário tecnológico pode levar os dispositivos a incorporar cada vez mais as funções de outros dispositivos como pa*gers*, *games*, câmeras fotográficas e outros. Consequentemente, quanto mais robusto o dispositivo, mais existirá a necessidade de sistemas com maior grau de complexidade (GOMES; FERNANDES, FERREIRA, 2012).

A evolução proporcionada pelo aumento do armazenamento de dados, faz com que os dispositivos mais potentes possam utilizar aplicações complexas, armazenando um maior número de informações. Por exemplo, a função de armazenar os contatos telefônicos, nos primeiros celulares realizavam apenas o nome do usuário e seu número. Após cada evolução desses dispositivos, acrescentou-se mais informações ao mesmo, referentes ao contato, como: aniversario, *e-mail*, grupo a que pertence, e tantas outras:

> A quantidade de informações que as pessoas precisam absorver e processar é cada vez maior, logo, a tecnologia deve servir como um agente para liberar o usuário de obrigações tipicamente destinadas às máquinas. Uma vez que o usuário consiga utilizar com facilidade a agenda de contatos de seu aparelho celular, não mais necessita memorizar diversos números, mesmo os mais utilizados. Neste sentido, a tecnologia deveria aliviar a carga informacional que esse tipo de usuário precisa carregar (ABREU, 2005, p.18).

O uso dos diversos dispositivos nos diferentes ambientes ocorre com intuito de proporcionar aos usuários uma boa experiência, onde se possa utilizar a tecnologia ao seu favor, facilitando suas tarefas diárias. Podendo ser desde o salvamento de contatos em um dispositivo, ao registro de atividades em uma agenda, ou até mesmo atividades referentes a trabalho.

Pode-se aferir que o uso da tecnologia "não se evidencia somente no momento em que vemos um dispositivo em uso, mas culturalmente nossas ações, nossas relações e nosso vocabulário denunciam que estamos fortemente influenciados por esta era digital" (SABOIA; VARGAS, VIVA, 2013, p. 04), já que os indivíduos passaram a se comunicar mais através desses dispositivos nos diversos ambientes.

Segunddo Lecheta (2013, p. 18): "O mercado corporativo também está crescendo muito, e diversas empresas estão buscando incorporar aplicações moveis a seu dia-a-dia", dessa maneira os dispositivos móveis estão passando a ocupar um papel importante também no meio corporativo.

2.2 Sistema Operacional Android

Os Dispositivos móveis possuem diferentes características dependendo da empresa fabricante, Chainho (2014, p. 19) argumenta que: "Nos dispositivos móveis existe uma grande variedade de SO, com diferentes versões, por vezes adaptadas pelos fabricantes especificamente para cada dispositivo".

Pode-se analisar que "antes do *Android*, cada marca possuía seu sistema operacional exclusivo. A *Apple* tinha o *IOS*, o *Blackberry* o *RIM*, a *Nokia* o *Symbiam*, a única exceção era a *Microsoft* que exigia uma taxa por cada aparelho vendido" (OTSUKE; ZENELATO, 2012, p. 04), sendo assim o surgimento do *Android* modificou o cenário dos dispositivos móveis no mercado mundial.

A revolução proporcionada pelo *Android* parte do fato de que o SO foi o primeiro a separa o *hardware* do *software* (GARGENTA, 2011, *apud* GOMES, 2011, p. 16), pois permitiu que diversos dispositivos pudessem utilizá-lo, sem a necessidade de adaptá-lo para cada caso. Dessa forma, pode-se dizer que: "Um número muito maior de dispositivos pode executar as

mesmas aplicações e cria um ambiente bem mais rico para desenvolvedores e consumidores" (GOLCALVES; TOLEDO, 2012, p.13).

O *Android* é um sistema Operacional bastante poderoso, ousado e flexível. Lecheta (2013, p. 22) trata-o como "uma nova plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux, com diversas aplicações já instaladas".

Desde seu lançamento o sistema foi difundido por todo o mundo. Dessa forma sendo visto como uma plataforma que pode utilizar máxima performance dos seus dispositivos, uma de suas vantagens é a usabilidade:

> O *Android* é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, *middleware*, aplicativos e *interface* do usuário. *Android* foi construído com a intenção de permitir aos desenvolvedores criar aplicações móveis que possam tirar total proveito do que um aparelho portátil possa oferecer (PEREIRA, 2009, *apud* OTSUKA, ZANELATO, 2012, p.03).

Em sua recente história pode-se estabelecer algumas datas importantes que tiveram influência sobre a forma como as pessoas interagem com seus celulares e dispositivos móveis, uma delas foi a sua criação e sua aquisição junto a *Google* no ano de 2005, seguindo por seu lançamento ocorrido em outubro de 2008.

O primeiro dispositivo lançado com a plataforma *Android* foi o *T-Modile G1* desenvolvido pela empresa *HTC Corporation*, que teve o começo de suas vendas iniciado no dia 22 de outubro de 2008 (LECHETA, 2013), dando assim início a popularidade do *Android* em todo o mundo.

Buscando formar uma plataforma padronizada para celulares e *smartphones*, a *Google* e diversas empresas do ramo, conceberam um consórcio chamado OHA (*Open Handset Alliance*) com o intuíto de construírem uma plataforma única e aberta, satisfazendo o usuário final (LECHETA, 2013).

O consórcio *OHA* é: "Uma associação de empresas de software, hardware e telecomunicações, cuja missão é desenvolver uma plataforma para dispositivos móveis que seja completa, aberta e gratuita" (MONTEIRO, 2012, p.12).

As empresas que formam a *OHA* colaborando na construção do Sistema Operacional *Android*, que por esse motivo passou a ter uma grande quantidade de dispositivos com a adesão a sua proposta.

Deitel *et. al.* (2013) observa que este consórcio contava inicialmente com 34 empresas e atualmente conta com 81, dentre elas encontra-se grandes empresas do mercado de tecnologia como: *Motorola, LG, Samsung*, dentre outras.

O *Android* foi concebido com o intuito de permitir que as aplicações tirem o total proveito dos aparelhos, como por exemplo qualquer aplicação pode chamar qualquer funcionalidade de núcleo do aparelho (PEREIRA, 2009).

É necessário considerar que as empresas podem realizar uma personalização do sistema, de acordo com suas necessidades, assim customizando o Sistema Operacional para cada aparelho, levando em consideração as próprias funcionalidades de cada dispositivo, sendo que não existe necessidade de pagar pelo uso do *Android*:

> Para os fabricantes de celulares, o fato de existir uma plataforma única e consolidada é uma grande vantagem para criar novos aparelhos. A grande vantagem para eles é que a plataforma também é livre e de código aberto. A licença do *Android* é flexível e permite que cada fabricante possa realizar alterações no código-fonte para customizar seus produtos, e o melhor de tudo, sem necessidade de compartilhar essas alterações com ninguém (LECHETA, 2012, p. 21).

Sua relevância consiste estruturalmente em oferecer uma plataforma que possibilite a manipulação de seu código fonte, contando com o auxílio de uma vasta comunidade que busca compartilhar soluções e evoluir no desenvolvimento da plataforma.

Deitel *et. al.* (2013, p. 04), pontua algumas de suas vantagens:

> Uma vantagem de desenvolver aplicativos Android é a franqueza (ou grau de abertura) da plataforma. O sistema operacional é de código-fonte aberto e gratuito. Isso permite ver o código-fonte do *Android* e como seus recursos são implementados. Você também pode contribuir para o *Android* relatando erros ou participando de grupos de discussão do *Open Source Project.*

Por ser uma plataforma *open source,* o *Android* sempre pode ser adaptado, visando assim, incorporar as tecnologias mais recentes, de acordo com a inserção destas no mercado. Dessa maneira, quando um novo recurso estiver disponível a comunidade *Android* irá facilmente incorporá-lo.

Sobre este ponto de vista, Pereira (2009, p. 03) pontua que: “A plataforma vai estar sempre em evolução, já que as comunidades de desenvolvedores estarão trabalhando em conjunto para construir aplicações”.

O Android foi construído levando em consideração o *GNU/Linux*, em sua versão 2.6, tratando-se de uma distribuição *Linux* de código fonte aberto, sendo assim um dos principais motivos pelo seu sucesso no mercado de dispositivos móveis, a este respeito foi:

> O núcleo do *GNU/Linux* que já possui vários recursos necessários para a execução de aplicações, como gerenciamento de memória, gerenciamento de processos, pilha de protocolos de rede, módulo de segurança e vários módulos de núcleo de infraestrutura. Como o sistema operacional é conhecido, também

> facilita o surgimento de melhorias aos *drivers* de *hardware* já existentes (BENDER, 2011, p.14).

No entanto, PEREIRA (2009, p.04) reverbera que a construção do *Android* em uma base do Linux, não quer dizer que o sistema, seja *Linux*, uma vez que "não possui *windowing system* nativo (componentes de GUI), não suporta *glibc* e não possui alguns dos conjuntos de padrões apresentados em algumas distribuições *Linux*".

Em sua essência, o SO é constituído de um vasto número de funcionalidades, podendo ser manipulado por qualquer pessoa com conhecimentos em linguagem de programação, pois se trata de um software livre, visto sua difusão, cada vez mais desenvolvedores estão interessados nesta plataforma. Lecheta (2013) opina que os programadores podem contribuir para o aperfeiçoamento da plataforma *Android*.

O sucesso do Sistema Operacional é justificado por causa do úmero de dispositivos, pois desta maneira o *Android* pode chegar a muitos usuários em dispositivos de pequeno, médio e grande porte, fazendo com que a sua popularidade aumente cada vez mais.

Para Meira (2014) "essa grande diversificação, apesar de dar ao *Android* um grande alcance, cria um problema sério de fragmentação da plataforma". Dessa forma, alguns dispositivos com menor poder computacional têm grande dificuldade em executar as versões mais recentes da plataforma *Android*.

Nesse sentido, uma grande aposta da *Google* é a disponibilização do *Android SDK* para os desenvolvedores, facilitando assim o desenvolvimento de aplicações para a Plataforma, pois esta ferramenta oferece todos os recursos necessários:

> Esse software disponibiliza os recursos necessários para criação e desenvolvimento de aplicações segundo as classes dispostas na *API*. Essas aplicações são escritas utilizando a linguagem de programação *Java* e executadas por uma máquina virtual projetada para rodar sobre um núcleo *Linux* e aperfeiçoada para o funcionamento em dispositivos móveis, chamada *Dalvik*. **(**KAWAMOTO, 2011, p. 23**).**

Gindri (2011) acrescenta que é fácil desenvolver para a plataforma *Android*, pois *SDK* possui uma boa quantidade de documentação, bastando o desenvolvedor consulta-la, além disso os únicos requisitos básicos para o desenvolvimento de aplicativos para o *Android* é o *Java*, o *SDK* e uma *IDE*.

A *Google* fornece uma infinidade de recursos de *hardware* no *SDK*, assim os desenvolvedores podem utiliza-lo quando estão construindo suas aplicações. São exemplos de componentes de *hardware*: *GPS*, Bússola, Acelerômetro, *Bluetooth*, câmera, entre outros.

Com a evolução da tecnologia móvel novos componentes de *hardware* surgem a cada dia, dessa maneira Crepaldi (2014, p. 32) pondera que:

> "Alguns fabricantes de *hardware* podem adicionar recursos que ainda não são suportados nativamente pelo *Android*. Geralmente quando isso ocorre, estes fabricantes disponibilizam um SDK para que os desenvolvedores usufruíam das características únicas de tais dispositivos"

2.2.1 Versões do Android

O *Android*, como qualquer outro sistema operacional, está em constante mudança, ocasionado pela necessidade de melhorias internas, bem como pela busca por inovações em seu ambiente gráfico e adaptabilidade a novas tecnologias.

A rápida evolução da tecnologia faz com que todos os sistemas operacionais estejam em constate atualização. Uma curiosidade sobre essas versões do *Android*, é que seus desenvolvedores as nomearam com nomes de sobremesas, e em ordem alfabética (Deitell *et. al.*, 2013).

A maioria dos dispositivos no mercado possuem o SO *Android* instalado de fábrica, cada empresa o customiza, tendo em vista garantir características próprias. Apesar da quantidade de dispositivos existente com o Sistema *Android* não significa que cada um possui a mesma versão, sobretudo porque aparelhos mais recentes naturalmente são lançados com versões mais recentes (LECHETA, 2013, p.34).

Cada versão é conhecida como uma plataforma, e cada qual têm um código identificador, que são conhecidos como *API (Application Programming Interface) Level*. Quando se desenvolve aplicativos para o *Android*, o desenvolvedor identifica quais funcionalidades terá o aplicativo e seleciona o nível da *API*, pois a partir deste é determinado quais funcionalidade o aplicativo pode incorporar e também informa os dispositivos que por ventura podem receber a aplicação (GOMES, 2011).

Por exemplo, um aplicativo com a API 19 se possuir alguma peculiaridade surgida no sistema nesta versão, não poderá ser instalada em versões do SO mais antigo, podendo ocasionar lentidão no aplicativo.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

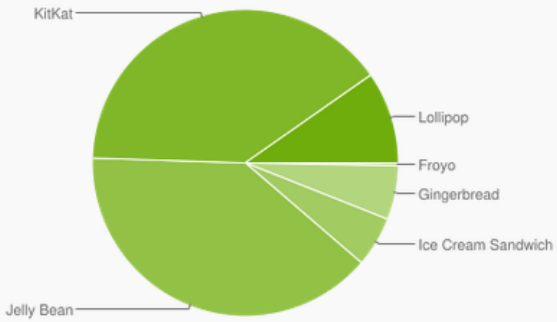

Figura 1: Distribuição das versões do Android nos dispositivos
Fonte: Google (2015)

Pode-se aferir pela figura 01, que o número de dispositivos com a versão mais recente do SO ainda possui poucos adeptos. No entanto, as versões 4.4 (*Android kitkat)* e as versões 4.1, 4.2 e 4.3 (*Android Jelly Bean*) são mais populares por estarem a mais tempo no mercado, assim possuindo um maior nicho de mercado.

É importante ressaltar que a atualização do *Android* não resulta necessariamente em migração das versões mais antigas para as versões recentes do sistema, a este respeito Chainho (2014) afirma que alguns dispositivos não possuem *hardware* adaptados para o sistema, diferentemente dos computadores que recebem atualizações para seus sistemas operacionais com maior facilidade.

Para o Sistema Operacional a fragmentação tem sido algo bom quando levando em consideração a diversificação de aparelhos que podem receber o *Android*. No entanto pode ser considerado ruim quanto a satisfação plena do usuário final, principalmente para os usuários que utilizam aparelhos de baixo custo, uma vez que estes dispositivos muitas vezes não possuem *hardware* que suporte o *Android*, ocasionando uma experiência ruim para o usuário.

> Essa grande diversificação, apesar de dar ao *Android* um grande alcance, cria um problema sério de fragmentação da plataforma. Em sua versão 4.4, apelidada de *KitKat*, o sistema foi adaptado para executar de maneira satisfatória em dispositivos antigos ou com pouco poder computacional, sendo o primeiro passo da plataforma, na tentativa de diminuir a fragmentação. (LOVERIDGE, 2013 *apud* MEIRA, 2014, p.35)

Neste sentido, quanto ao desenvolvimento das aplicações para os dispositivos móveis, é necessário que o desenvolvedor tenha como foco a limitação dos dispositivos, bem como suas funcionalidades, pois o sistema é bastante fragmentado, assim “o programador precisa levar em

conta esses vários fatores na hora do desenvolvimento, ao disponibilizar seu aplicativo para *download* um usuário pode negativa – ló" (MENDONÇA; BITAR; DIAS, 2011, p. 01).

A cada nova versão, o Sistema Operacional *Android* ganha novas funcionalidades, adaptando-se aos diferentes *Hardware*, com isso é necessário que o *Android* SDK também seja atualizado, para que os desenvolvedores possam criar aplicativos tendo como foco as novas tecnologias:

> A cada nova versão do *Android* e, consequentemente, nova versão do kit de desenvolvimento *Android* (SDK), novos recursos são incluídos e outros depreciados, o que requer uma gerencia das funcionalidades que utilizam os antigos e novos recursos para que a aplicação se adapte a versão do Android instalado no aparelho cliente (CAMPOS; KULESZA; COELHO, 2014, p. 10).

Dentre estas versões, o *Android* 2.2 (*Froyo*) acrescentou diversos recursos para os desenvolvedores de aplicativos, a este respeito pode se destacar o serviço *Android Cloud to Device messaging* (C2DM) criado para facilitar a atualizações dos aplicativos:

> O serviço Android *CloudtoDevicemessaging* (C2DM) permite aos desenvolvedores de aplicativos enviar dados de seus servidores para seus aplicativos instalados em dispositivos *Android*, mesmo quando os aplicativos não estão sendo executados. O servidor avisa os aplicativos para que entrem em contato diretamente com o servidor afim de receber dados atualizados de aplicativos ou do usuário (DEITEL et. al., 2013. p. 09).

Em dezembro de 2010 foi lançado o *Android* 2.3 (*Gingerbread*) que acrescentou melhorias quanto ao tratamento ao usuário, dessa forma afetando as seguintes funções e componentes: Gerenciamento de energia, Atalho *Manage Applications*, Comunicação em campo próximo, funcionalidades copiar e colar, câmera (aplicativos acessam câmera traseira e frontal), chamada pela internet e *Aplicativo Downloads* (Deitel et.al., 2013).

Em sua versão 3.0 (*Honeycomb*), o *Android* foi aprimorado e passou a ter compatibilidade com as telas dos tablets, as galerias e a câmera ganharam novas funções para que o usuário aproveitasse ao máximo seu dispositivo.

Além das diversas novidades presentes nos aplicativos, chamou a atenção a possibilidade de customização da tela inicial e o compartilhamento *bluetooth* (GONCALVES; TOLEDO, 2012).

O *Android Ice Cream Sandwich* que foi lançado em outubro de 2011 (GONÇALVES; TOLEDO, 2012) incorporou o *Android* 2.3 e o *Android* 3.0 no mesmo SO, possibilitando o seu uso, em todos os dispositivos *Android*, portanto *tablets* e celulares passaram a ter uma plataforma que poderia ser utilizada em ambos os dispositivos e alcançar a mesma performance:

> Isso permite a incorporação de recursos do *Honeycomb* – como a interface holográfica do usuário, um novo lançador e muito mais (anteriormente disponíveis somente em *tablets*) em seus aplicativos para smartphones – e a fácil adaptação de seus aplicatvos facilmente para funcionar em diferentes dispositivos (DEITEL *et. al.*, 2013).

Diversas mudanças se sucederam no Sistema Operacional, podendo ainda destaca o acesso rápido a notificações, *status* do sistema, botões de navegação suaves que podia ser acessados em todo o sistema e aplicações.

2.2.2 Arquitetura

O *Android* é formado por um conjunto de ferramentas e bibliotecas trabalhando em conformidade com o *Kernel* do *Linux* (WALL, 2008, *apud* UZEJKA, 2011), dessa maneira pode-se dizer que funcionam como uma pilha, portanto hávendo de certa forma uma comunicação entre todas essas camadas, visto que a camada mais abaixo fornece suporte a outra e assim sucessivamente.

A este respeito, Chainho (2014, p.30) avalia que: "A Arquitetura *Android* é formada por camadas que intercomunicam mas dependentes entre si, onde as camadas de baixo nível fornecem serviços para as camadas de nível superior".

Pode ser observado na Figura 02, de forma grafica a pilha de camadas da arquidetura do Sistema operacional *Android*, onde as camadas superiores são dependestes das inferiores.

A Arquiteura do Android está dividida em 4 camadas (OTSUKA; ZANELATO, 2012), possuindo a seguinte ordem da base: a camada mais alta (*figura 02)*: *Linux Kernel*, Bibliotecas e Camada de execução, *Framework* e as aplicações.

A primeira camada na base da arquitetura encontra-se o *Kernel* do *Linux* (Núcleo *Linux*) atuando como uma camada de abstração entre o *hardware* e o *software* (HASLINGER; TONIAZZO, 2009).

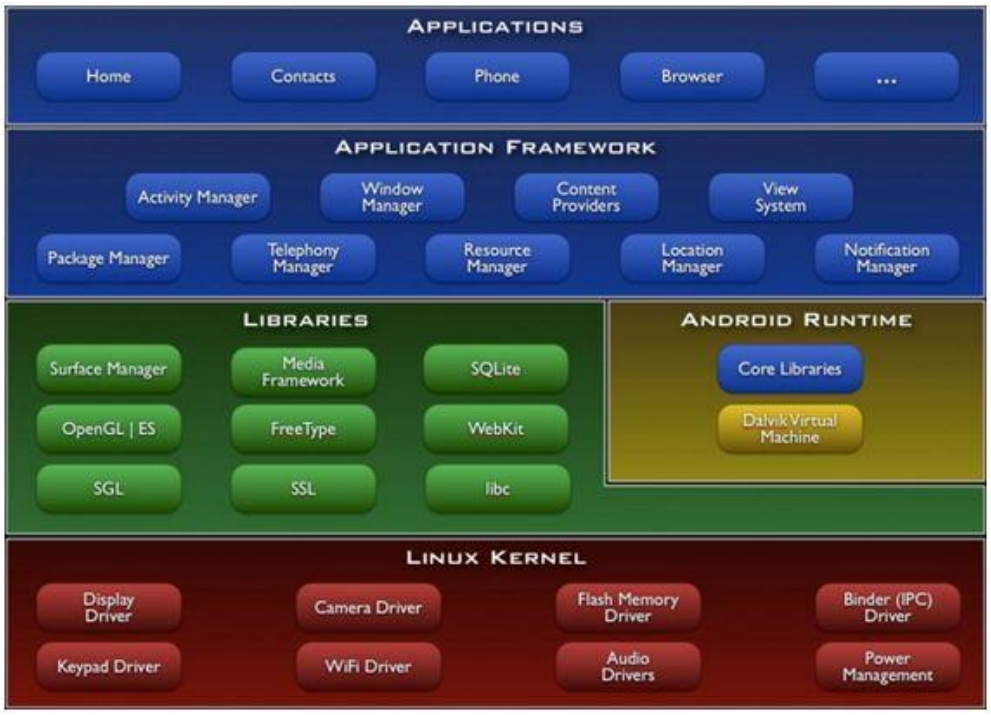

Figura 2: Arquitetura da Plataforma Android
Fonte: Google

Na primeira camada é utilizada a versão 2.6 do *Kernel*, sendo responsável para prover as diversas funções que estão presente no *hardware* (MEIRA, 2014).

O *Kernel* fica a cargo do gerenciamento dos processos, memória, gerenciamento de energia, protocolos de rede e serviços, os *drivers* e entre outros. Esta estrutura fornece toda a base a qual a plataforma necessita para ser construída.

Na segunda camada encontram-se as bibliotecas que são um conjunto de instruções as quais controlam o dispositivo, de acordo com diferentes tipos de dados. Estas bibliotecas são escritas nas linguagens de programação C e C++ e ficam a cargo da. "Parte gráfica (*Surface Manager e OpenGL*), multimídia, banco de dados (*SQLite*), suporte a *browsers* (*WebKit*), além de camadas de rede e gerenciamento de fontes" (UZEJKA**,** 2011**,** p. 18).

Todos os componentes das bibliotecas são utilizados pela camada superior da pilha da arquitetura, pois tratasse de códigos nativos do sistema, mas que fornecem várias funções para os aplicativos que os desenvolvedores desejam construir.

Ainda na segunda camada se encontra a *DVM (Dalvik Virtual Machine*) que foi otimizada para os dispositivos de pequeno porte, substituindo a *JVM* padrão no papel de executar aplicações de código *Java* sobre o sistema.

PEREIRA (2009, p.08) acrescenta que:

> A pequena camada do ambiente de execução (*Android Rutime*) é uma instância da máquina virtual *Dalvik*, criada para cada aplicação executada no Android. A *Dalvik* é uma máquina virtual com melhor desempenho, maior integração com a nova geração de hardware e projetada para executar várias máquinas virtuais paralelamente. Foi projetado para funcionar em sistemas com baixa frequência de *CPU*, pouco memória *RAM*.

A Dalvik sofreu alterações quanto ao uso de memória visando alcançar maior desempenho, o código binário que é gerado na execução de uma aplicação possui um formato próprio, sendo que sua execução ocorre a partir de um *bytecode java* e é convertido para o formato *dex – Dalvik Executable* (LECHETA, 2013).

Na *DVM* os programas são executados um ao lado do outro e de forma independente, mas compartilhando por muitas vezes recursos da plataforma *android*, afim de alcançar um maior desempenho:

> Cada aplicação *Android* é uma instância da máquina virtual *Dalvik*, ou seja, cada aplicação é executada como um processo distinto dos demais. A *Dalvik* é uma máquina virtual desenvolvida especialmente para obter maior desempenho em *hardwares* com pouco poder de processamento **(**HASLINGER; TONIAZZO, 2009, p.36).

Para Gindri (2011, p.13) como a concepção do Android teve o intuito de rodar sobre recurso tanto de memória, processamento e consumo limitado, não pode-se pensar em utilizar a máquina convencional, portanto a *DVM* foi modificada a afim de se alcançar melhor performance.

Na terceira camada fica a *Frameworks de application*, que foram projetados afim de simplificar o reuso dos componentes, assim o desenvolvedor poderia usufruir das mesmas *APIs* de aplicações, abstraindo de certa forma a complexidade existente.

A camada *frameworks applications* além de prover estrutura aos programas, possuem diversos componentes que auxiliam para a construção dos aplicativos, dentre esses componentes pode-se citar *activities, services, content providers* (UZEJKA, 2012), além destes, existem alguns recursos que formam um aplicativo, que são: *layouts, strings*, estilos e imagens e o arquivo *manifest*.

Encontra-se nesta camada todas as *APIs* e os recursos utilizados pelos aplicativos, e outros componentes que ficam disponíveis para o desenvolvedor:

> *View system* (componentes utilizados na construção de aplicativos), provedor de conteúdo (*Content Provider*), que possibilita que uma aplicação possa

> acessar informações de outra aplicação, ou até mesmo compartilharem as suas informações, possibilitando a troca de informações entre aplicativos e gerenciadores de recursos (permite definir e carregar recursos em *Rum Time*), gerenciador de localização (*GPS* e *Cell ID*), gerenciador de notificações (fornece informações sobre eventos que ocorrem no dispositivo), de pacotes e de atividade, que controla todo o ciclo de vida de aplicação e o acesso a navegação entre as aplicações (PEREIRA, 2009, p.06).

Na parte superior da arquitetura da plataforma *Android* encontra-se a camada *applications*, onde é executado as aplicações do sistema *Android*, que são desenvolvidos em Java, independentemente de sua origem, sejam aplicativos nativos, dos fabricantes ou instalados pelo usuário (GOMES, 2011), que são convertidos em *bytecode* em um formato próprio da *Dalvik VM*, depois são executados no sistema.

Os Aplicativos muitas das vezes são fornecidos pela própria plataforma "fabricantes" ou desenvolvedores que os disponibilizam, além disso existem aplicativos já pré-instalados no sistema *Android*, como: " Serviço de voz, serviço de *SMS/MMS*, *email*, calendário, navegador *web* e agenda. Ademais, os *OEMs* (*Original Equipment anufacturer*) podem inserir seus próprios aplicativos no dispositivo por motivos diversos" (BRAGA *et. al.*, 2011, p. 05).

2.2.3 Distribuindo um Aplicativo

Uma aplicação *Android* quando concluída é compactada, formando um arquivo do tipo *.apk (*Android Package File*) e em seguida executada pela máquina virtual *Dalvik* (LECHETA, 2013), que possui os seguintes componentes:

- Arquivos executáveis para a máquina virtual: Estes representam todo o código da aplicação depois de compilado para ser executado pela máquina virtual do *Android, a Dalvik*
- Recursos: São todos os itens que a aplicação necessita que não seja código, como arquivos de imagem, áudio, vídeo e outros.
- Bibliotecas: Eventualmente durante a implementação da aplicação podem ser utilizadas algumas bibliotecas que serão adicionas ao pacote no momento da criação do arquivo final da aplicação.

As aplicações são disponibilizadas na loja de aplicativos do *Google*, que contem atualmente milhares de aplicativos e provavelmente esse número só têm a aumentar, pois existe facilidade para desenvolver *Apps* para o *Android*, levando em consideração fatores como a

grande comunidade na *web* que fornece documentação que ajuda no desenvolvimento e o próprio material disponibilizado pela *Google*.

O número de aplicativos na loja da *Google*, por exemplo, no ano de 2011, movimentou cerca de 8,9 bilhões de dólares (GONÇALVES; TOLEDO, 2012), sendo que este valor é contabilizado através de compras *in-app*, publicidade e patrocínio.

Os aplicativos disponibilizados podem ser gratuitos ou pagos, cabe ao desenvolvedor definir o preço para os pagos, (DEITEL *et. al.*, 2013) analisa que 70% dos rendimentos é repassado ao desenvolvedor, por conseguinte pontua que os desenvolvedores com o intuito de fazer marketing para seus aplicativos disponibilizam versões de forma gratuita, e somente depois de efetuar testes no sistema o usuário possa decidir pelas versões mais completas.

O número de *downloads* de aplicativos gratuito têm sido maiores que o número de *downloads* pagos, visto que é mais provável o usuário baixar um aplicativo se for gratuito (Deitel *et. al.*, 2013). Em muito dos casos se busca alcançar rentabilidade com um determinado aplicativo, o desenvolvedor opta por oferecer uma versão gratuita e a partir de determinado ponto pode solicitar que o usuário atualize, ainda existem possibilidades de anuncio através do aplicativo.

Existem ainda outras formas de instalar aplicativos que não estejam no Loja de Aplicativos do *Android*. No entanto a *Google* recomenda que o usuário baixe os aplicativos diretamente da *Google Play*:

> É comum encontrarmos aplicações para o *download* na *internet* no formato *.apk. Na verdade o ideal é instalar as aplicações pelo *Google Play*, sendo que estas são controladas pelo *Google* e você sabe que não irão danificar seu celular. Mas se você fizer o *download* de aplicações externas e tem certeza que não irá prejudicar seu celular é possível instalá-la facilmente (LECHETA, 2013, p.53).

O usuário pode optar por comprar seus aplicativos disponíveis em outras lojas de aplicativos *Android* e ainda em seu próprio site, usando serviços como *AndroidLicenser*, contudo, é necessário ficar atento ao termo de licença do *Google Play*, pois não se pode utilizar informações do cliente obtido na loja de aplicativos da *Google*.

Quando o desenvolvedor envia seu aplicativo para a *Google Play* deve ter em mente os dispositivos que têm o nível de compatibilidade desejada, além disso é necessário determinar os níveis da *APIs*, ou seja a quais versões do *Android* o sistema poderá rodar e assim alcançar a performance desejada.

2.3 Linguagens de Programação para *Android*

Quando se pensa em produzir aplicações para o Sistema Operacional *Android*, imagina-se uma gama de dispositivos que podem utilizar a aplicação, sobretudo por tratar-se de um mercado em ascensão, cujo nicho comercial é bastante amplo.

Nesse mesmo pressuposto existem diversas possibilidades para o desenvolvimento de aplicações no Sistema Operacional *Android*, cada qual com suas limitações e benefícios. A este respeito ainda, Motyczka (*et. al.*, 2011, p. 03) explica que: "É possível desenvolver aplicativos para o ambiente *Android* utilizando a linguagem *Java* em conjunto com o *SDK* ou mesmo utilizar a linguagem C++ em conjunto com o *NDK* (*Native Development Kit*) ".

Lecheta (2013, p. 32) avalia que o *SDK* pode ser visto como: "O software utilizado para desenvolver aplicações no *Android*, que tem um emulador para simular o celular, ferramentas utilitárias e uma *API* completa para a linguagem *Java*, com todas as classes".

O *SDK* é uma aplicação que engloba um emulador para às diferentes versões do *Android*, possibilitando emular as aplicações desenvolvidas sem a necessidade do uso de um celular físico.

Deitel (et al., 2013, p. 19) explica que o emulador *Android* nos Sistemas Operacionais propiciam: "Executar aplicativos *Android* em um ambiente simulado dentro do *Windows, Mac OS X* ou *Linux.* O Emulador exibe uma janela de *interface* de usuário realista". Dessa maneira é possível construir a aplicação e fazer testes em um computador.

A possibilidade de possuir um ambiente de testes pode vir a trazer diferentes benefícios, principalmente em relação aos gastos no desenvolvimento de aplicações, pois não existiria a necessidade de dispositivos para testar os aplicativos

A esse respeito Motyczka (et. al., 2011, p. 02) pondera:

> "Dentre as diversas vantagens de adotar essa metodologia, destaca-se a eliminação da obrigatoriedade de possuir um dispositivo móvel real para o desenvolvimento e testes iniciais da aplicação a ser desenvolvida. Isso proporciona uma significativa economia, principalmente se a equipe de desenvolvimento for numerosa".

A linguagem de Programação *Java* é bastante utilizada no meio corporativo, sendo uma das mais difundidas do mundo. A utilização dessa linguagem em grande escala pode ser explicada pelo fato de ser de código-fonte aberto e gratuito, assim torna-se uma ferramenta perfeita para desenvolvimento de aplicações, além de possui uma ampla comunidade de desenvolvedores.

Existem outras formas de desenvolver aplicações para o Sistema *Android*, na qual "apesar da linguagem *Java* ser a linguagem oficial no desenvolvimento de aplicações, a plataforma *Android* oferece um modo de invocar códigos nativos como, por exemplo, os escritos em C e C++. Esta ferramenta é chamada de *Android* NDK" (FARINA, 2012, p.35). Dessa maneira, quando o desenvolvedor inicia o desenvolvimento de uma aplicação pode enveredar por um deste segmentos que são amplamente conhecidos.

Para LECHETA (2012, p. 802) o *NDK*: " É um conjunto de ferramentas que permitem o desenvolvimento de código nativo em C ou C++ para ser executado diretamente no Sistema Operacional ".

O uso de C/C++ no *Android* possibilita a utilização de recursos presente no Sistema-operacional, isso acontece porque boa parte do *Android* foi construído nessa linguagem de programação, assim podendo alcançar melhor performance quanto a utilização dos diversos recursos. Assim, em alguns casos mais complexos recomenda-se essa abordagem.

Para o desenvolvimento de aplicações para o Sistema Operacional *Android* é necessária uma análise da necessidade da aplicação, considerando que cada linguagem de Programação engloba diferentes aspectos.

2.3.1 Linguagem Java

Segundo Deitel (*et. al.*, 2013, p.05): "Os aplicativos *Android* são desenvolvidos em *Java* – a linguagem de programação mais usada no mundo".

Para muitos provavelmente uma das melhores opções quando se imagina desenvolver aplicações para o desenvolvimento *Android*, pois como muitos profissionais já têm um bom conhecimento dessa linguagem de programação que é considerada bastante poderosa.

A linguagem de programação *Java* é uma das linguagens de programação mais difundidas no mundo, sendo que hoje está em diversos dispositivos que contém suporte para o desenvolvimento de aplicações dessa linguagem. Sua portabilidade é uma de suas principais características:

> *Java* é uma linguagem orientada a objetos que tem como principal característica a portabilidade. O *bytecode*, que é o código gerado pelo compilador *Java*, pode ser transportado entre plataformas distintas, desde que estas suportem *Java*, e com isso não é necessário recompilar um programa para que ele rode em máquinas e sistemas diferentes". (LOURO, 2014, p.10).

O *Java* é utilizado para o desenvolvimento de aplicações seja em ambientes empresariais, servidores *web*, aparelhos de consumidores e ainda podendo ter como foco outros ambientes, trata-se de uma linguagem que se molda a cada ano, e continua a crescer.

Esta linguagem foi criada na década de 90 por uma equipe de programadores chefiado por James Gosling, na empresa *Sun Microsystems* (LOURO, 2014). Segundo Mendes (2009) o principal motivo da criação do projeto que levou ao surgimento da linguagem *Java* foi a intenção de criar uma nova plataforma para computação interativa, portanto o foco principal não seria a linguagem, mas a plataforma com um todo.

A plataforma *Java* é composta pela *Java Virtual Machine* (*JVM*) e a *Aplication Programming Interface* (*API*), sendo que a *JVM* é responsável pelo carregamento e execução dos códigos escritos em *Java*, já as *APIs* consistem em classes prontas que são usadas no desenvolvimento das aplicações.

A linguagem de programação *Java* foi de fato apresentada a todos no dia 23 de maio de 1995, quando o diretor da *Sum Microsystems* John Gage e o executivo do Netscape Marc Andreessem lançaram a plataforma, sendo incorporada ao navegador *Netscape Navigator* (MENDES, 2009), que era o navegador mais popular na época.

Santos (2012) pontua que as principais características da linguagem de programação *Java* são: simples, orientada a objeto, multiprocessamento, interpretada, independente de arquitetura, portabilidade, alto desempenho, robusta.

O motivo pelo qual a linguagem de programação *Java* pode ser considera simples, se deve ao fato de permitir o desenvolvimento em diversos sistemas operacionais e arquitetura de *hardware*, sendo que o desenvolvedor não se preocupe com questões da infraestrutura.

Os projetistas não colocaram algumas características complexas das linguagens C e C++, apesar de possuir sintaxe muito semelhante. “Porém mais simplificada, pois não trabalha com arquivos de cabeçalho, aritmética de ponteiros, estruturas, uniões, herança múltipla e outros” (FARINA,2012, p.28). De certa forma, quando comparada a outras, a linguagem Java possui uma infinidade de classes que podem beneficiar o uso de sua estrutura.

Java é uma linguagem de programação orientada a objeto, e por isso possui em sua essência a possibilidade do uso dos conceitos de polimorfismo, encapsulamento e herança, facilitando o desenvolvimento de aplicações. Diferentemente do paradigma estruturado, o *Java* se assemelha a mecanismo muito próximos do humano quanto a tentativa de retratar a complexidade de um sistema.

Mendes (2009, p.19) pontual quanto a característica *multithread* do *Java*:

> A plataforma *Java* permite a criação de programas que implementam o conceito *multithread*, incluindo sofisticados mecanismos de sincronização entre processos. O *multithreading* é uma técnica de programação concorrente, que permite projetar e implementar aplicações paralelas de forma eficiente.

A JVM basicamente interpreta todo e qualquer *bytecode* gerado após a compilação, bastando para isso ter uma máquina com o sistema operacional onde a máquina virtual *Java* esteja instalado seja em *Windows*, *Linux* ou *Mac OS*. Dessa maneira as aplicações escritas podem ser rodadas, por causa disso pode-se dizer que o *Java* possui a qualidade de ser independente de arquitetura.

Quando se fala na *JVM*, percebe-se que este é o que garante a portabilidade pertencente ao *Java* (MENDES, 2009), pois todo código em si é escrito primeiramente em um documento de texto com extensão *Java*, que após a compilação é convertido em um arquivo "*class"*.

O arquivo "*class"* é um arquivo no formato *bytecode* e poderá ser executado em qualquer plataforma, dessa maneira quando se desenvolve em *Java* pode se executar o mesmo programa em outros sistemas operacionais com o pacote *Java* instalado.

Essa linguagem foi elaborada para criar sistemas confiáveis, sendo considerada robusta por esse fator, mas também, por oferecer outros recursos. Santos (2011, p.40) pontua algumas características: "Verificação de código em tempo de compilação, checagem de variáveis definidas e não inicializadas, inicialização de variáveis e atributos automaticamente com valores *default*, manipulação de exceções".

Quanto a robustez da linguagem Java, Farina (2012, p.28-29) acrescenta que a linguagem de programação Java propõe "a verificação dinâmica posterior (em tempo de execução) e a eliminação de situações sujeitas a erros, como a aritmética de ponteiros das linguagens C e C++".

Deitel (et. al. 2013) afirma que o uso dessa linguagem de programação no desenvolvimento de aplicativos para dispositivos moveis ocorre, porque o programador pode contar com uma poderosa biblioteca de classes, que faz com que o desenvolvimento de aplicações seja mais rápido, alcançando uma melhor performance em um menor tempo de trabalho.

2.4 Cloud-Computing

A computação em Nuvens (*Cloud-Computing*) é uma nova abordagem quanto ao armazenamento dos arquivos e provimento de serviços que não estão presentes em computadores locais, assim as empresas que oferecem esses serviços possuem uma grande infraestrutura e demanda.

Sousa (*et. al.*, 2010, p. 02) faz analogia do uso da computação em nuvens aos serviços de utilidade pública como água, eletricidade, telefone e gás:

> As infraestruturas existentes permitem entregar tais serviços em qualquer lugar e a qualquer hora, de forma que possamos simplesmente acender a luz, abrir a torneira ou usar o fogão. O uso desses serviços é, então, cobrado de acordo com as diferentes políticas de tarifação para o usuário final. Recentemente, a mesma ideia de utilidade tem sido aplicada no contexto da informática e uma mudança consistente neste sentido tem sido feita com a disseminação de *Cloud Computing* ou Computação em Nuvem.

A sociedade contemporânea passou a utilizar vários serviços por demanda, neste contexto pode-se pressupor que os ambiente de tecnologia da informação poderiam seguir nesta mesma direção, de forma que os usuários também podem passar a optar por preferir utilizar serviços computacionais por demanda, acessando-os em qualquer máquina e local.

Em um ambiente computacional em nuvem o usuário passaria a acessar seus arquivos e programas em qualquer dispositivo, assim sua preocupação com os equipamentos físicos seria minimizado, cabendo essa tarefa ao administrador do sistema.

> Com o *Cloud Computing*, muitos aplicativos dos usuários, assim como seus arquivos e dados, não precisam mais estar instalados ou armazenados no computador ficando disponíveis na "nuvem". Ao fornecedor da aplicação cabem todas as tarefas de desenvolvimento, armazenamento, manutenção, atualização, *backup*, escalonamento, etc. Dessa maneira, tem disponível todo o seu material e documentos, em qualquer ambiente independente de onde a aplicação esteja rodando. (OLIVEIRA; MOZZAQUATRO, 2011, p. 03)

A rápida evolução dos ambientes computacionais vem modificando a forma como se interage com as máquinas, com o passar dos anos os *smartphones* passaram a incorporar as funcionalidades dos D*esktops*, nesse sentido podemos considerar a evolução seguinte, o uso da computação em nuvens, pois diversos dispositivos podem se conectar a nuvens e acessar os arquivos dos usuários e os serviços que estes necessitam.

Segundo Sousa (et al., 2010) os recursos computacionais de *hardware* têm tendência a se tornarem obsoletos, assim, o uso de infra-estrutura de terceiros seria uma boa escolha,

considerando que para usar o serviço, não é necessitaria conhecer a infraestrutura de Tecnologia da Informação utilizada.

2.5. Processo de Engenharia de software

A engenharia de *software* tem papel relevante na criação de qualquer aplicação, seja ele de pequeno, médio ou grande porte. Tendo como fundamentação a modularização e a organização do *software*, caminhando em paralelo com os sistemas de informação, dessa forma se tornando cada vez mais importante no desenvolvimento de diversas aplicações. Para Pressman (2011) o processo de engenharia de software deve engloba o software em todos os campos e formas.

Em termos gerais a engenharia de software pode ser denominada como metodologia de desenvolvimento e manutenção de sistemas modulares (RESENDE, 2005). Para melhor identificar este conceito, é necessário ressalvar o significado da palavra engenharia que pode ser definido como a arte de construção, com base no conhecimento cientifico e empírico.

A engenharia estabelece diferentes técnicas e práticas que venham a proporcionar uma ampla cobertura no desenvolvimento de *software*. Dessa maneira, o processo de desenvolvimento de software deve ter como objetivo a obtenção da melhor qualidade:

> Qualquer abordagem de engenharia (inclusive engenharia de software) deve estar fundamentada em um comprometimento organizacional com a qualidade. A gestão da qualidade total Seis Sigma e filosofias similares promovem uma cultura de aperfeiçoamento contínuo de processos, e é esta cultura que, no final das contas, leva ao desenvolvimento de abordagens cada vez mais efetivas na engenharia de software. A pedra fundamental que sustenta a engenharia de software é o foco na qualidade. (PRESSMAN, 2011, p.39).

O processo da engenharia de *software* em vias gerais não define um padrão que deve ser seguido a qualquer custo, mas sim uma abordagem que pode ser moldada de acordo com as necessidades de sua equipe de trabalho, portanto a equipe tem autonomia na definição de suas ações e tarefas.

A engenharia prevê elucidar desde a concepção do projeto do sistema em questão até sua finalização e reutilização, pois os sistemas de informação estão em constante mudança (GUEDES, 2011), assim os mesmos nunca estão completamente finalizados. Para isso é necessário um uma boa elicitação de requisitos e a documentação de todas as fases do desenvolvimento, para a possibilidade de mudanças futuras:

> É importante perceber que os requisitos de um software mudam com o tempo. Essas mudanças ocorrem porque o ambiente em que o software reside também muda. Os países alteram suas leis, as organizações alteram suas práticas, a tecnologia evolui, os usuários exigem novas funcionalidades até então não imaginadas ou desnecessárias. (XEXEO, 2007, p. 42)

Para Guedes (2011) é possível citar outras etapas no processo de desenvolvimento de *software*, que são: Análise de requisitos, Projeto, Codificação, Testes e implementação. Ainda segundo o mesmo, estas etapas podem receber diferentes nomenclaturas dependendo por vezes do método e processo adotado no desenvolvimento, assim as vezes algumas das etapas citadas podem ser divididas.

2.5.1 UML – Linguagem de Modelagem Unificada

A UML (*Unified Modeling Language*) é uma linguagem visual utilizada para modelar software que foi baseada no paradigma orientado a objeto (GUEDES, 2011). Foi criado na década de 90 pela junção de três métodos de modelagem conhecidos, que foram definidos por Booch, Jacobson e Rumbaugh (PÁDUA).

Pressman (2011) observa que a UML foi a combinação de um grupo de modelagem distintas usada na época de seu desenvolvimento, ocasionando assim a popularização desta. A UML é tratada por muitos como fator preponderante na definição de especificações de sistema, pois é através desta modelagem que o desenvolvedor poderá demonstrar interações do sistema, assim como as limitações quanto ao uso do *software.*

Durante a análise de requisitos para o desenvolvimento do *software*, a modelagem da UML é necessária para o bom desenvolvimento do projeto:

> Uma linguagem de modelagem auxilia a levantar questões que não foram concebidas durante as entrevistas iniciais. Tais questões devem ser sanadas o quanto antes, para que o projeto de software não tenha que sofre modificações quando o seu desenvolvimento já estiver em andamento, o que pode causar significativos atrasos (GUEDES, 2011, p.23).

A UML possui 13 diagramas (figura 03) para uso na modelagem de *software* (PRESSMAN, 2011), podendo o desenvolvedor expressar de forma logica todas as características do sistema. Dessa maneira todas as partes irrelevantes são minimizadas para não poluir a própria modelagem do sistema, ou seja, o desenvolvedor pode retirar algum aspecto da modelagem para sanar problemas de interpretações futuras.

Os diagramas que compõe a UML 2.0 são exemplificados na figura abaixo, possuindo a nomenclatura como é demostrado.

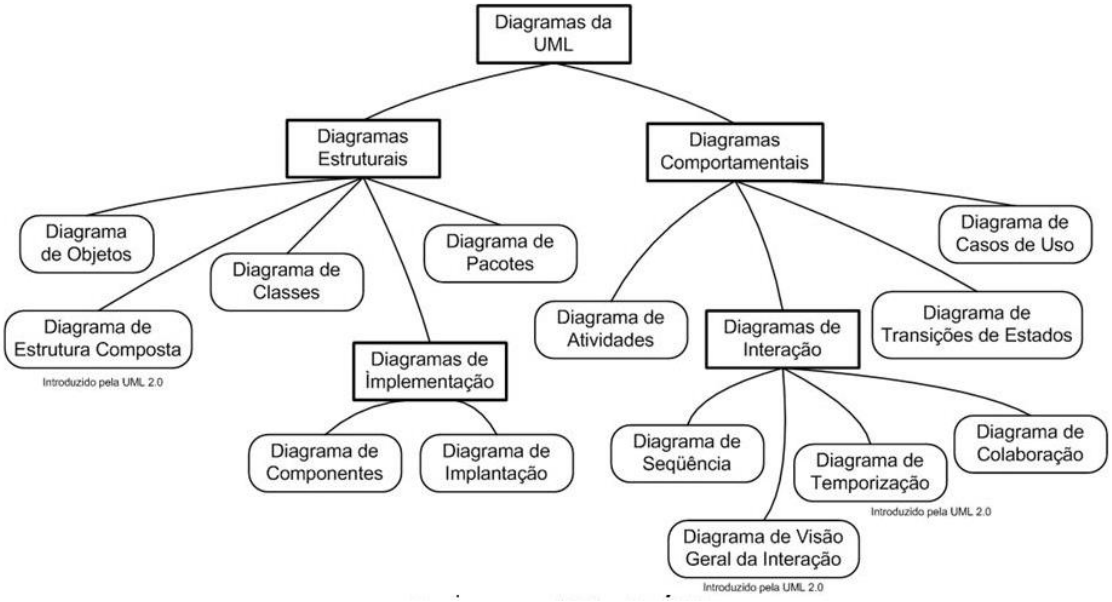

Figura 3: Diagramas definidos pela UML
Fonte: Bezerra(2015)

Os diagramas da UML são divididos em Digramas Estruturais e Diagramas comportamentais, como é mostrado na figura 3. Booch (2005, p.98) define que os diagramas comportamentais são usados para: “visualizar, especificar, construir e documentar os aspectos dinâmicos de um sistema. Considere os aspectos dinâmicos de um sistema como uma representação de suas partes aos quais sofrem alterações”.

Os comportamentais são formados pelos seguintes diagramas:

- Diagrama de Caso de Uso (*Use Case*) – são usados na etapa de análise e levantamento de Requisitos do Sistema.
- Diagrama de Máquina de Estados – servem para acompanhar certas mudanças ocorrido por um objeto dentro de certo processo.
- Diagrama de Atividades - descreve os passos para a conclusão de uma atividade.
- Os Diagramas de Interação também possuem uma subdivisão, trata-se dos seguintes diagramas:

    1. Sequência – serve para a descrição da ordem temporal em que as mensagens acontecem entre os diferentes objetos.
    2. Geral interação - variação dos diagramas de atividades que fornece visão geral dentro do sistema ou processo do negócio.

3. Temporização - descreve mudanças de estado, ou na condição de uma instância de uma classe ou seu papel durante o tempo.

Segundo Booch (2005, p.96) os diagramas estruturais são usados para: “visualizar, especificar e documentar os aspectos estáticos de um sistema. Considere os aspectos estáticos do sistema como uma representação de seu esqueleto e estrutura relativamente estáveis”.

Os diagramas estruturais são:

- Diagrama de Classe – é fundamental e o mais utilizado na UML, pois serve de apoio aos outros diagramas. Mostrando o conjunto de classes com atributos, métodos e os relacionamentos entre classes.
- Diagrama de Objeto – está relacionado com o diagrama de classes, sendo considerado um complemento deste. Fornecendo visão dos valores armazenados nos objetos de um Diagrama de Classe em um determinado momento da execução do processo do software.
- Diagrama de Componentes - tem objetivo de indicar os componentes do software e os relacionamentos.
- Diagrama de implantação – especifica as necessidades de hardware e características físicas para o Sistema.
- Diagrama de Pacotes - representa os subsistemas englobados, desta maneira determinando as partes que os compõem.
- Diagrama de Estrutura - escreve a estrutura interna de um classificador.

Bezerra (2015, p.28) observa que: “O desenvolvimento de um sistema de *software* complexo demanda que seus desenvolvedores tenham a possibilidade de examinar e estudar esse sistema a partir de perspectivas diversas”.

Na construção de um *software* deve ser avaliado diversos ângulos acerca do problema proposto, assim o desenvolvedor deve analisar para posteriormente criar os diagramas, pois cada um dos diagramas proporciona visões diferentes do projeto.

Guedes (2011, p.30) cria uma analogia quanto ao uso de tantos diagramação para mostrar um único sistema:

> Percebemos que ao se projetar uma construção, esta não tem apenas uma planta, mas diversas, enfocando o projeto de construção do prédio sob diferentes formas, algumas referentes ao *layout* dos andares, outras apresentando a planta hidráulica e outras ainda abordando a planta elétrica, por exemplo. Isso torna o projeto do edifício completo, abrangendo todas as

> características. Da mesma maneira, os diversos diagramas *UML* permitem analisar o sistema".

A *UML* pode ser usado em projetos de pequeno médio e grande porte, pois independentemente do tamanho do software é necessário que o desenvolvedor tenha conhecimento daquilo com que está trabalhando.

Booch (2005, p.07) a este respeito explica:

> "Qualquer projeto será beneficiado pelo uso de algum tipo de modelagem. Inclusive no setor de softwares comerciais, em que ás vezes é mais comum distribuir softwares inadequados devido à produtividade oferecida pelas linguagens de programação visual, a modelagem poderá auxiliar a equipe de desenvolvimento a visualizar melhor o planejamento do sistema e permitir que o desenvolvimento seja mais rápido, ajudando a construir o item correto. Quanto mais complexo for o sistema, maior será a probabilidade de ocorrência de erros ou de construção de itens errados, casa não haja qualquer modelagem".

3. APLICATIVO SINCRONIZE

Esse capítulo mostra as especificações do Sistema proposto nesse trabalho, que de agora em diante será tratado. Será mostrado o levantamento de requisitos, diagramas de casos de uso, diagrama de classes e principais diagramas de atividades do sistema para que seja possível identificar todas as funcionalidades do sistema.

A ferramenta desenvolvida tem por objetivo principal a automatização de *backup* dos contatos do sistema *Android* de forma a proporcionar que o usuário tenha facilidade ante a integridade das informações dos seus contatos, visto que por algum motivo teve o dispositivo móvel danifica, roubado, entre outros motivos, necessitando a restauração de seus contatos.

3.1 Ferramentas Utilizadas

Para a modelagem do sistema foi utilização a aplicação *Astah Ruting*, onde foram modelados os diagramas UML que têm importância significativa na elaboração das próprias funcionalidades deste e de qualquer outra aplicação, pois este fator vem a contribuir para o seu bom desenvolvimento.

A linguagem de programação utilizada neste projeto foi a linguagem Java, apesar de existir outras linguagens de programação que podem ser utilizadas, a escolha desta se deve principalmente ao fato de possuir uma boa quantidade de documentação e materiais didáticos para estudo, facilitando a consulta e correção de problemas oriundos do desenvolvimento do *software*. Além do mais é necessário ressaltar também que a própria *Google* indica a linguagem de programação *Java.*

O protótipo do sistema foi desenvolvimento na *IDE Android Studio* 1.1, sendo está uma ferramenta desenvolvida pela própria *Google*, podendo ser citadas as seguintes vantagens quanto à sua utilização:

- Interface atraente, possibilidade de customizar atalhos.
- Instalação do *SDK* sem complicações.
- Funcionalidade *Injection Language*, onde *strings* de outras linguagens possam ser validadas pela *IDE*, permitindo a realização de testes com estas entradas.
- Integração com sistemas de controles de versão.
- *Sample Importing* & *templates* oferecendo ao desenvolvedor possibilidades de importar códigos externos.

- Ferramenta de performance (*Memory* monitor) para a aplicação desenvolvida.
- Ferramenta para a interação com o serviço da *Google*

3.3 Requisitos de software

Com o sistema de *backup* de contatos do *Android* deve realizar o envio do arquivo de contatos para um ambiente em nuvem, bastando para isso o usuário realizar configurações iniciais no aplicativo. Após uma análise foi observado os seguintes requisitos funcionais que são apresentados na Quadro 01.

Identificador	**Descrição**	**Depende de**
RF01	O sistema deverá permitir a escolha de um ambiente online	
RF02	O sistema deverá permitir que o usuário escolha um diretório para salvar o arquivo dos contatos	
RF03	O sistema deverá permitir a sincronização com ambiente online	RF01
RF04	O sistema permitirá a reestruturação de seus contatos	RF03
RF05	O sistema permitirá que o usuário permita ligar a sincronização	RF01, RF02
RF06	O sistema deve permitir a escolha de um diretório para sincronização	

Quadro 01: Requisitos Funcionais
Fonte: O Autor(2015)

O Quadro 02 atenta para os requisitos não funcionais que a aplicação deve possuir, o quadro descreve de forma sucinta estes requisitos.

Identificador	**Descrição**	**Depende de**
RN01	O Sistema deverá estar sincronizando em tempos pré-determinado, dessa forma o sistema se usara de meio para enviar o arquivo a um ambiente virtual	
RN02	O Sistema deve poder se comunicar com diferentes ambientes em nuvem	RN01
RN03	O Sistema deve permitir sanar arquivos duplicados	
RN04	O Sistema deve permitir que os arquivo de contatos fiquem disponível para o usuário se conectar com qualquer computador ou dispositivo móvel e recuperar o arquivo.	RN02
RN05	O Sistema deve permitir a manipulação de suas configurações básicas	

Quadro 02: Requisitos Não-Funcionais
Fonte: O Autor (2015)

Com a definição dos requisitos que a aplicação deve possuir é necessário construir um modelo de dados, com telas que irão se comportar de acordo com o clicar do usuário, assim o aplicativo o levará a diferentes telas. É necessário perceber que quando se desenvolve qualquer

aplicação após se ter o levantamento de requisitos, o usuário deve construir e documentar diagramas a fim de facilitar o desenvolvimento.

3.3 Diagramas UML

Os diagramas da UML são de grande importância na execução de qualquer projeto de *software*, pois oferecem diferentes visões para o desenvolvedor sobre o que está sendo produzido Booch (2005, p.92) acrescenta:

> Um diagrama é uma apresentação gráfica conectado de vértices (itens) e arcos (relacionamentos). Use os diagramas para visualizar o seu sistema sob diferentes perspectivas. Uma vez que nenhum sistema complexo pode ser compreendido em sua totalidade a partir de única perspectiva, a UML define um número de diagramas que permite dirigir o foco para aspectos diferentes de seu sistema de maneira mais independente.

Neste trabalho, foram utilizados três diferentes diagramas para a visão do sistema como um todo, pois apesar de todos possuírem utilidades faz-se necessariamente somente o uso do diagrama de caso de uso (*Use Case*) e diagrama de atividade.

Guedes (2011) esclarece que o diagrama de caso de uso é mais geral e informal, este é bastante utilizado no levantamento de requisitos de qualquer software, além de ser a base para a construção de outros diagramas. O *Use Case* oferece a possibilidade de mostrar como o sistema irá se comportar em si, identificando os usuários, os outros sistemas que venham até algum relacionamento com o sistema desenvolvido e até mesmo características especiais que estejam sendo desenvolvidas.

O diagrama de caso de uso é mostrado na figura 04, pode-se notar que existem dois atores no diagrama, o *USER* exemplificando o usuário que irá interagir com o sistema e o ator Tempo que para este diagrama simboliza o próprio sistema, ou algo que este venha a executar internamente sem o usuário saber.

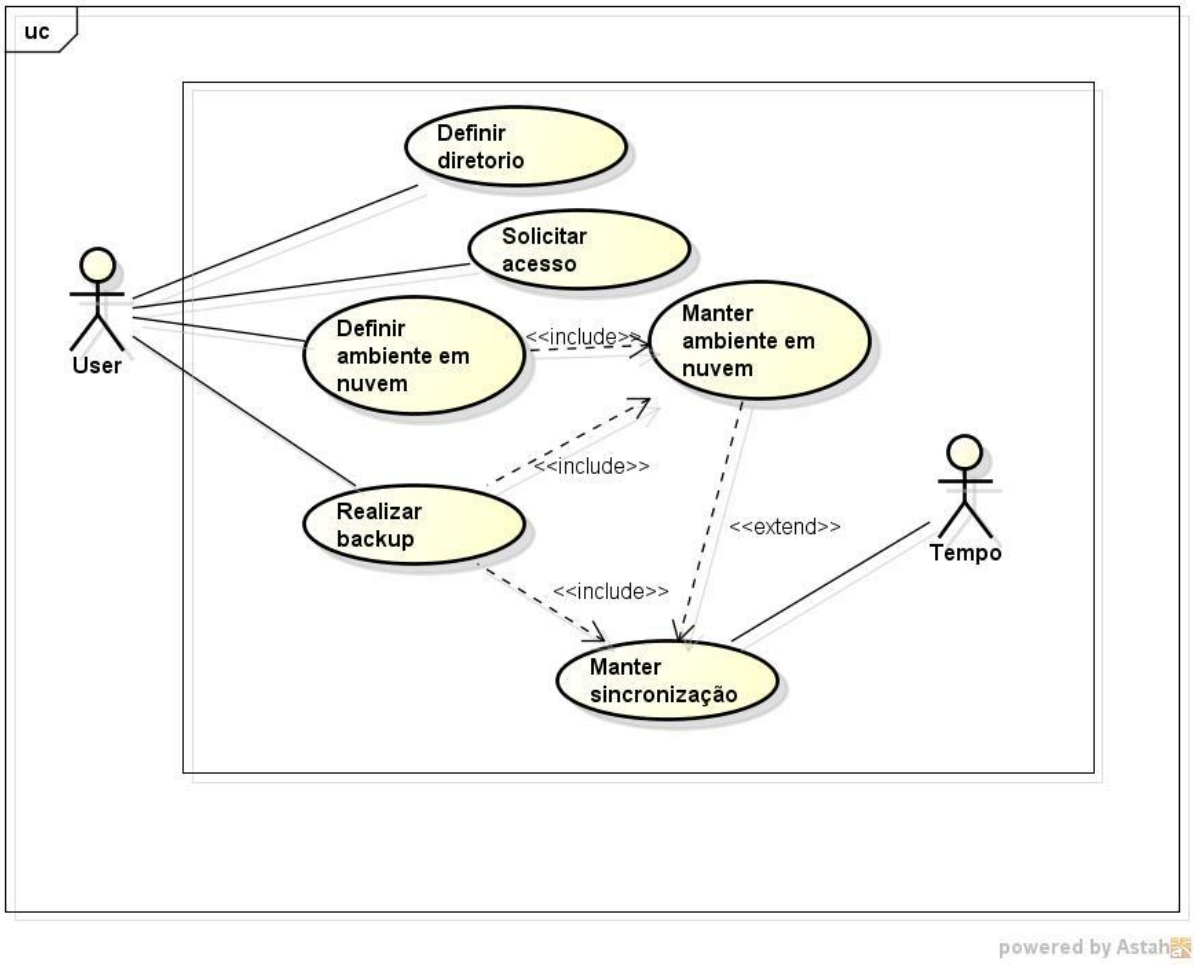

Figura 04: Diagrama de Caso de Uso
Fonte: O Autor (2015)

A seguir são mostrados os detalhes dos use case da Figura 04, mostrando os atores que estão associados, sua descrição e os requisitos que esses estão relacionados:

- **UC01 – Definir Diretório:**
  - **Ator:** *User*
  - **Descrição:** Pode-se decide-se onde deverá salvar o arquivo gerado pelo sistema, o qual contém os dados referentes aos contatos.
  - **Requisitos:** RF02.
- **UC02 – Solicitar Acesso:**
  - **Ator:** *User*
  - **Descrição:** O sistema permite o acesso ao ambiente virtual, comunicando-se com o mesmo para envio de arquivo.

- **Requisitos:** RF03.

- **UC03 – Definir ambiente em nuvem:**
  - **Ator:** *User*
  - **Descrição:** Deve permitir a escolha de qual ambiente em nuvem deve sincronizar com o aplicativo;
  - **Requisitos:** RF01.
- **UC04 – Manter ambiente em nuvem:**
  - **Ator:** *User*
  - **Descrição:** Manter sincronização com um ambiente escolhido pelo usuário.
  - **Requisitos:** RF01.
- **UC05 – Realizar backup:**
  - **Ator:** *User*
  - **Descrição:** Realizar o envio do arquivo de contatos do *Android* para o ambiente selecionado pelo usuário.
  - **Requisitos:** RF01.
- **UC06 – Manter sincronização:**
  - **Ator:** *User*, Tempo
  - **Descrição:** A sincronização deverá ser estendida até que o usuário queira que ela se desfaça com o ambiente em nuvem escolhido.
  - **Requisitos:** RF01.

Após a construção do diagrama de caso de uso e a identificação dos requisitos, têm a construção dos demais diagramas que são criados tendo como base.

A figura 05 mostra o diagrama de atividade que enfatiza a sequência e condições para coordenar o comportamento, assim este descreve a situação de definição de um diretório, em que o usuário solicita permissões para navegar entre as diferentes pastas e definir o diretório que ficará conectando com o ambiente em nuvem.

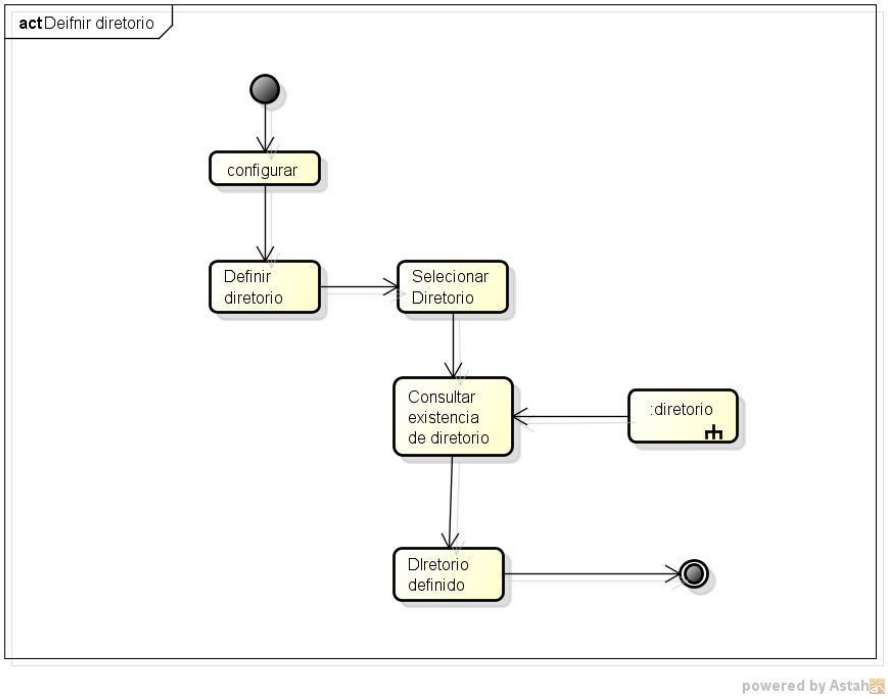

Figura 5: Atividade Definição Diretório
Fonte: O Autor (2015)

O usuário deve selecionar um diretório que deverá possuir a sua versão mais recente do arquivo de contatos, isso ocorre porque o sistema deve manter de certa forma uma conexão com o ambiente em nuvem. Quando há este tipo de comunicação entre a aplicação sincronize e os diretórios do *Andoroid,* a aplicação sincronize chama o próprio gerenciador do Android.

O aplicativo para dispositivos existe com a necessidade de uma conexão de dados, quando sincroniza, o aplicativo busca saber se está conectado, assim possibilitando que a conexão seja estabelecida e por consequência a requisição seja enviada e posteriormente o dispositivo poderá conectar-se e enviar os dados.

O diagrama demostrado na figura 06 exemplifica o caso da sincronização do sistema, o sistema funciona como um serviço, possibilitando que o usuário não tenha domínio sobre esta sincronização. No entanto, o usuário fica a cargo de habilitar ou desabilitar o próprio sistema, outra questão no diagrama é o fato que o usuário defini o ambiente em nuvem.

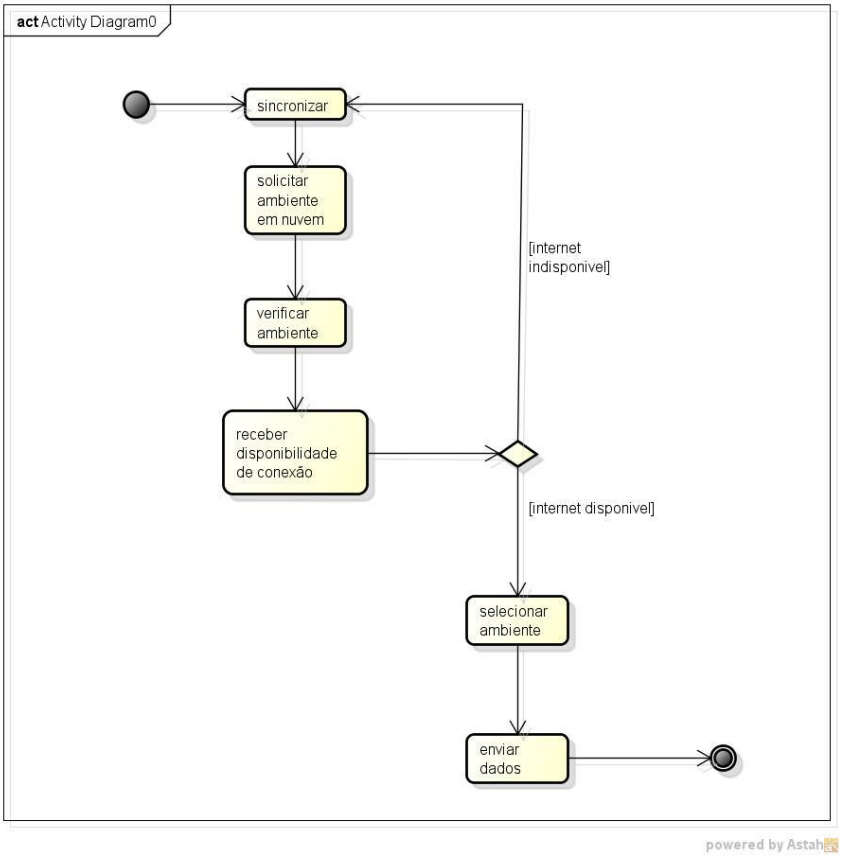

Figura 6: Atividade Sincronizar
Fonte: O Autor (2015)

4. RESULTADOS E DISCUSSÕES

Durante todo o projeto foram identificados diferentes problemas quanto ao acesso de rotinas internas do sistema operacional Android, como seria feito o acesso ao ambiente em nuvem, assim como a forma de captura dos contatos. Os ambientes em nuvem que podem ser citados são: *Dropbox*, *GoogleDrive*, *SkyDrive*, entre outros.

A Figura 07 retrata a tela inicial, que contém os botões de configuração, botão de informações e o botão de sincronizar

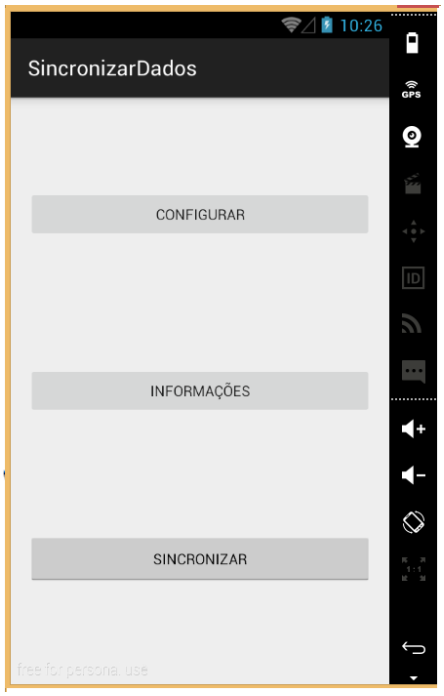

Figura 07: tela Inicial
Fonte: O autor

O Botão configurar redireciona o sistema para uma nova tela, onde o usuário irá determinar qual ambiente em nuvem deseja realizar seu backup, seleciona o local onde a pasta deve sincronizar os dados.

O Botão Informações oferece um pequeno tutorial de como é efetuado o processo de sincronização, assim como os dados da aplicação. O terceiro botão é responsável por ativar e desativar o processo de sincronização, assim como a obtenção de dados que estão em um ambiente já selecionado.

A tela de configuração é mostrada na Figura 08, é nesta tela que o usuário irá definir a pasta para que o arquivo dos contatos seja importado e exportado, assim como é neste mesmo ambiente que o usuário poderá escolher um ambiente em nuvem que esteja a sua disposição. O Botão determinar liga o sincronismo dos dados e habilita-o de forma imediato.

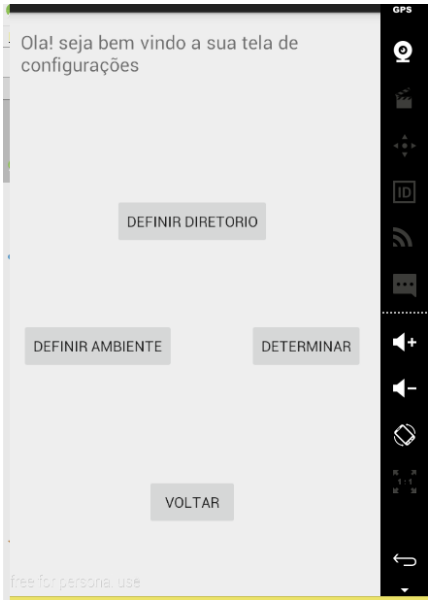

Figura 08: Tela de configurações
Fonte: O Autor

A tela que é representada na Figura 09, o usuário poderá ativar e desativar a sincronização dos dados, sendo que isto ocorre somente quando a conexão de dados, caso o usuário esteja usando conexão 3G, o sistema ira avisar que este pode economizar em seu pacote de dados mudando para uma rede *wi-fe*.

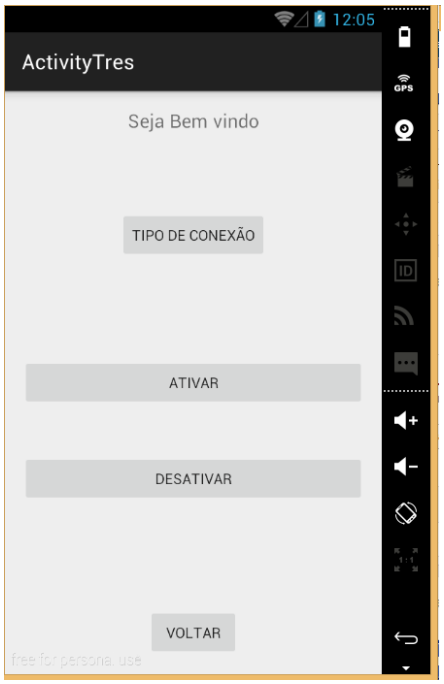

Figura 09: Tela para efetuar ligar/desligar sincronização
Fonte: O Autor (2015)

As telas mostradas aqui fazem parte do protótipo do sistema para sincronização dos contatos do Android, visto que são as telas principais para o funcionamento de todo a estrutura do aplicativo.

A Figura 10 apresenta a tela Tipo de Conexão, em que se o usuário desejar economizar em sua taxa de transferência de dados, ele pode habilitar o sistema a quando a conexão 3G for habilitada, os dados de sincronização são interrompidos.

Figura 10: Tela tipo de Conexão
Fonte: O Autor (2015)

5. CONSIDERAÇÕES FINAIS

Este trabalho foi desenvolvido com o objetivo de apresentar um aplicativo que se conectasse a um ambiente em nuvem, buscando manter os contatos do usuário. O sistema é reverberado sobre o sistema operacional *Android*, já que este é um dos sistemas mais utilizados no mercado. O mesmo tem como finalidade também facilitar o processo de sincronização com um ambiente em nuvem, visto que buscou-se simplificar da melhor maneira possível, assim beneficiando o usuário final.

Foram demostrados todo o processo de desenvolvimento de software, visto que este processo abrange a própria engenharia de *software* que é a essência para aplicações que venham a ser desenvolvidos, seja em ambiente educacional ou comercial.

As tecnologias utilizadas para o desenvolvimento do aplicativo que foram supracitadas durante todo o exercício de desenvolvimento deste trabalho foram essenciais para o funcionamento do mesmo. Sendo que a aplicação foi desenvolvida visando rodar em dispositivos móveis que possuam o sistema operacional *Android*, este contém uma ampla documentação na *internet* que é referenciada em diversos livros.

O desenvolvimento de aplicativos para este sistema operacional ocorre com o auxílio do próprio *SDK*, que é fornecido pelo próprio *Android*, já a linguagem de programação mais utilizada no desenvolvimento de aplicações é a linguagem *Java*.

Em relação a trabalho futuros, percebe-se que existe a necessidade de maximizar a possibilidade de ambiente em nuvem e também em servidores, destes determinados pelo usuário. Doravante têm-se como objetivo, adaptar o aplicativo para outros sistemas operacionais, tornando-o multiplataforma.

REFERÊNCIAS

ABREU, Leonardo Marques de. **Usabilidade de telefones celulares com base em critérios ergonômicos**. Disponível em: http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=6705@1. Acessado em 10 de abril. 2015.

ANATEL (Agencia Brasileira de Telecomunicações). **Brasil fecha junho de 2014 com 275,71 milhões de acessos móveis.** Disponível em: http://www.anatel.gov.br/Portal/exibirPortalInternet.do. Acessado em: 24 de julho de 2014

BENDER, Carlos Roberto. **G-SMS: protótipo de aplicação de envio de SMS georeferenciadas**. Disponível em: <http://campeche.inf.furb.br/tccs/2011-I/TCC2011-1-10-VF-CarlosRBender.pdf>. Acessado em 09 de abril de 2015.

BEZERRA, Eduardo**. Princípios de análise e projeto de sistemas com UML**. 3. Ed. Rio de Janeiro: Elsevier, 2015.

BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivar. **UML- Guia do usuário**. Rio de Janeiro : Elsevier, 2005

BOTTENTUIT JUNIOR, João Batista. **Do Computador ao Tablet: Vantagens Pedagógicas na Utilização de Dispositivos Móveis na Educação**. Disponível em: <http://www.latec.ufrj.br/revistas/index.php?journal=educaonline&page=article&op=view&path%5B%5D=291>. Acessado em 05 de abril de 2015.

BRAGA, Alexandre Melo; Erick Nogueira do Nascimento; Lucas Rodrigues da Palma e Rafael Pereira Rosa. **Introdução à Segurança de Dispositivos Móveis Modernos – Um Estudo de Caso em Android.** Disponível em: <http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/ceseg:2012-sbseg-mc2.pdf>. Acessado em 02 de abril de 2015.

CAMPOS, Edmilson; KULESZA, Uirá; COELHO, Roberta. 2014. **Um Estudo Exploratório da Arquitetura e Projeto de Aplicativos Android**. Disponível em: <http://hillside.net/sugarloafplop/papers/12.2.pdf>. Acessado em 17 de abril de 2015.

CARNEIRO, Mariana; ROMAN, Clara; FAGUNDEZ, Ingrid. **Vendas de smartphones e tablets crescem mais de 100% em 2013**. Disponível em: <http://www1.folha.uol.com.br/mercado/2014/01/1391973-vendas-de-smartphones-e-tablets-cresceram-mais-que-100-em-2013.shtml>. Acesso em 15 de janeiro de 2015.

CHAINHO, Francisco Nicolau Gomes. **Plataforma parametrizável para análise forense de dispositivos móveis: análise forense para SO Android**. Disponível em: <https://repositorio.ipbeja.pt/handle/123456789/756>. Acessado em 12 de abril de 2015.

CREPALDI, Danielle Domeneghetti**. Desenvolvimento de objeto de aprendizagem para dispositivos móveis explorando os sensores internos do sistema operacional Android**. 2014. Universidade Estadual Paulista, Faculdade de Ciências, 2014. Disponível em: <http://hdl.handle.net/11449/118796>.

DEITEL, Paul; DEITEL, Harvey; DEITEL, Abbey; MORGANO, Michael. **Android para programadores: uma abordagem baseada em aplicativos.** Revisão técnica: Daniel Antonio Callegari. - Porto Alegre: Bookman, 2013.

FARINA, André Marcelo. **Biomobile: sistema de identificação de usuários em dispositivos móveis na plataforma Android utilizando reconhecimento de faces a partir de vídeo**. 2012. 72 f. Dissertação (mestrado) - Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas, 2012. Disponível em: <http://hdl.handle.net/11449/89344>.

GINDRI, Alexandre Felin. **Estudo de injeção de falhas para a Máquina Virtual do sistema Android**. Disponível em: <http://hdl.handle.net/10183/31048>. Acessando em 20 de abril de 2015.

GOMES, Evandro Westphalen Carlos. **Global Interface Process usando android através de webservices.** 2011. 47 f. Universidade Tecnológica Federal do Paraná, Curitiba. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/619>. Acessado em 1 de maio de 2015.

GOMES, Rafael Caveari; FERNANDES, Jean alves R.; FERREIRA, Vinicius Corrêa**. SISTEMA OPERACIONAL ANDROID.** Disponível em: <http://www.andrix.com.br/wp-content/uploads/2014/08/Android-SO.pdf>. Acessado em 12 de abril de 2015.

GONÇALVES, Luiz Gustavo dos Santos; TOLEDO, Leandro da Costa. **Protótipo de Aplicativo Para Propaganda Mobile Em Android**. Disponível em: <http://177.107.89.34:8080/jspui/bitstream/123456789/19/1/Gon%C3%A7alvesToledo.pdf>. Acessado em 2 de abril de 2015.

GOOGLE. **Dashboards**. Disponível em: <https://developer.android.com/about/dashboards/index.html>. Acessado em 12 de maio de 2015.

GUEDES, Gilleanes T. A. **UML 2: Uma abordagem prática.** 2. Ed. São Paulo: Novatec Editora, 2011.

HASLINGER, Mayara Cristina; TONIAZZO, José Carlos. **Protótipo para localização de pontos de referência na cidade de Chapecó utilizando google Android e google Maps**. 2009. 1 CD-ROM : Monografias (Conclusão do curso de Sistemas de Informação) -- Universidade Comunitária Regional de Chapecó, 2009. Disponível em : <http://www5.unochapeco.edu.br/pergamum/biblioteca/php/imagens/000082/00008281.pdf>. Acesso em 09 de abril de 2015.

KAWAMOTO, VINÍCIUS RAMOS. **Desenvolvimento de aplicações para Android utilizando o framework Adobe Flex**. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/533>. Acessado em 20 de abril de 2015.

LECHETA, Ricardo R. **Google Android : Aprenda a Criar aplicações para dispositivos moveis com Android SDK**. 3ª ed. – São Paulo : Novatec Editora, 2013.

LEMOS, André. **Cidade e mobilidade. Telefones celulares, funções pós-massivas e territórios informacionais**. Disponível em:

<http://200.144.189.42/ojs/index.php/MATRIZes/article/viewArticle/3993>. Acessado em 12 de maio de 2015.

LOURO, Rodrigo Duarte. **Jogo Para a Plataforma Android utilizando Web Service.** Disponível em: <https://linux.ime.usp.br/~murch/mac0499/monografia.pdf>. Acessado em 27 de abril de 2015.

MARTINS, Rafael J. Werneck de A. **Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android.** Disponível em: <http://www.icad.puc-rio.br/~projetos/android/files/monografia.pdf>. Acessado em 23 de abril de 2015.

MEIRA, Guilherme Tebaldi. **Projeto e desenvolvimento de um aplicativo móvel para acesso ao Portal do Aluno da UFES na plataforma Android.** Disponível em: <http://www.lprm.inf.ufes.br/sites/default/files/monografia-versaofinal.pdf>. Acessado em 13 de abril de 2015.

MENDES, Douglas Rocha. **Programação Java com ênfase em Orientação a Objetos**. Novatec Editora, 2009.

MENDONÇA, Vinicius Rafael Lobo; BITTAR, Thiago Jabur; DIAS, Márcia de Souza. **Um estudo dos Sistemas Operacional Android e iOS para desenvolvimento de aplicativos. Disponível em:** <http://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011_submissi on_54.pdf>. Acessado em: 30 de junho de 2014.

MONTEIRO, João Bosco. **Google Android Crie Aplicações para celulares e tablets**. 1ª Ed. São Paulo: Casa do Código, 2012.

MOTYCZKA, Leonardo Bressan; SAUSEN, Paulo Sérgio; CAMPOS, Mauricio de; SAUSEN, Airam. **Framework de referência para desenvolvimento de aplicações Android aplicado a automação de subestações de energia elétrica**. Disponível em: <http://www.metaeventos.net/inscricoes/formularios_off/resumo_preenchido/DINCON/Motyczka-L-1.pdf>. Acessado em 30 de junho de 2014.

OLIVEIRA, Leander Cordeiro de; MOZZAQUATRO, Patrícia Mariotto. **Estudo sobre Cloud Computing; Um novo paradigma para E-LEARNING e M-LEARNING**. Disponível em: <http://www.unicruz.edu.br/16_seminario/artigos/agrarias/ESTUDO%20SOBRE%20CLOUD% 20COMPUTING%20-%20UM%20NOVO%20PARADIGMA%20PARA%20E-LEARNING%20E%20M-LEARNING.pdf>. Acessado em: 15 de junho de 2014.

OTSUKA, Gilberto Sadao; ZANELATO, Ana Paula Ambrósio. **O SISTEMA ANDROID NO UNIVERSO DOS DISPOSITIVOS MÓVEIS.** Disponível em: <http://intertemas.toledoprudente.edu.br/revista/index.php/ETIC/article/viewArticle/3759 >. Acessado em 9 de abril de 2015

PÁDUA, Wilson de. **Engenharia de software**: **fundamentos, métodos e padrões**. 1ª edição. Rio de Janeiro: Editora LTD, 2000.

PEREIRA, Lúcio Camilo Oliva. **Android para desenvolvedores.** Rio de Janeiro : Brasport, 2009.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. 7ª Edição. Porto Alegre: AMGH, 2011.

ROTONDO, Gustavo; Wilson Jacobsen; Daniela Almeida; Mariana Pompeo Freitas; Gerson Nunes; Érico M. H. Amaral. **Segurança e integridade de dados em sistemas operacionais móveis – Um estudo de caso sobre a plataforma Android**. Disponível em: http://periodicos.unesc.net/index.php/sulcomp/article/viewArticle/1774. Acessado em 12 de maio de 2015.

SABOIA, Juliana; VIVA, Marco Aurélio de Andrade; VARGAS, Patrícia Leal de. O. **USO DOS DISPOSÍTIVOS MÓVEIS NO PROCESSO DE ENSINO E APRENDIZAGEM NO MEIO VIRTUAL**. Disponível em: <http://ojs.cesuca.edu.br/index.php/cesucavirtual/article/view/424>. Acessado em 12 de abril de 2015.

SANTOS, Jefferson Luiz Ribeiro dos. **Protótipo de aplicativo para acompanhamento da carteira de ações para a plataforma Android**. 2012. 100 f. Universidade Tecnológica Federal do Paraná, Curitiba, 2012. Disponível em: <http://www.infomoney.com.br/minhas-financas/gadgets/noticia/2580608/brasil-segundo-pais-com-mais-roubo-celulares-mundo>. Acessado em 30 de junho de 2014.

SILVA, Juliano Américo Lourenço. **Brasil é o segundo país com mais roubo de celulares no mundo – InfoMoney.** Disponível em: <http://www.infomoney.com.br/minhas-financas/gadgets/noticia/2580608/brasil-segundo-pais-com-mais-roubo-celulares-mundo>. Acessado em 30 de junho de 2014.

SOARES, Raquel; PEREIRA, Marco; MARTINS, Joaquim Arnaldo. **Recolha, preservação e contextualização de objectos digitais para dispositivos móveis com Android.** Disponível em: <http://www.scielo.gpeari.mctes.pt/scielo.php?script=sci_arttext&pid=S1646-98952012000100 00 7&lang=pt>. Acessado em: 20 de junho de 2014.

SOUSA, Flávia R.C.; MOREIRA, Leonardo O.; MACÊDO, José Antônio F.; MACHADO, Javam C. **Gerenciamento de Dados em Nuvém: Conceito, Sistemas e Desafios.** Disponível em <http://www.lia.ufc.br/~flavio/papers/sbbd2010.pdf>. Acessado em 1 de julho de 2014.

UZEJKA, Guilherme de Moraes**. Modelando um cliente voip inteligente para a plataforma Android**. Disponível em: <http://hdl.handle.net/10183/31033>. Acessando em 12 de maio de 2015.

XEXEO, Geraldo. **Modelagem de Sistemas de Informação: Da Analise de Requisitos ao modelo de Interface**. Versão 2007/Agosto.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Flavio de Sousa Oliveira, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação AUTOMATIZAÇÃO DE BACKUP DOS CONTATOS DO SISTEMA ANDROID: UMA FERRAMENTA PARA GARANTIR A INTEGRIDADE DOS CONTATOS de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 17 de novembro de 2015.

Flavio de Sousa Oliveira
Assinatura

Flavio de Sousa Oliveira
Assinatura