FACULDADE DE TECNOLOGIA DE SÃO PAULO

EDUARDO HUBSCH

Uma Abordagem Comparativa do desenvolvimento de aplicações para dispositivos móveis

SÃO PAULO
2012

FACULDADE DE TECNOLOGIA DE SÃO PAULO

EDUARDO HUBSCH

Uma Abordagem Comparativa do desenvolvimento de aplicações para dispositivos móveis

Monografia submetida como exigência
parcial para a obtenção do Grau de
Tecnólogo em Processamento de Dados
Orientador: Prof. Me. Rodrigo Zuolo Carvalho

SÃO PAULO
2012

AGRADECIMENTOS

AGRADEÇO EM PRIMEIRO LUGAR A DEUS, POIS SEM ELE, NADA TERIA ALCANÇADO. PARA NÃO SER INJUSTO OU INGRATO, PREFIRO NÃO CITAR NOMES, CONTUDO TENHO CERTEZA DE QUE AS PESSOAS QUE REALMENTE FIZERAM A DIFERENÇA PARA MIM SABERÃO QUE ESTÃO INCLUSAS NESSE PEQUENO AGRADECIMENTO.

RESUMO

Este trabalho se propõe a fazer um estudo comparativo entre os dois principais sistemas operacionais para dispositivos móveis existentes no mercado atualmente: *Apple iOS* e *Google Android*. Serão detalhadas as plataformas em critérios como arquitetura, componentes fundamentais, interface com o usuário, recursos, ambientes de desenvolvimento, entre outras características. A comparação será baseada na implementação de uma aplicação com foco na capacidade de reprodução de conteúdo multimídia (vídeo).

Palavras-chave: *Android*, *iOS*, *Smartphone*, *iPhone*, Vídeo

ABSTRACT

This paper aims to make a comparative study between the two main operating systems for mobile devices on the market today: Apple iOS and Google Android. Platforms are detailed on criteria such as architecture, key components, user interface, resources, development environments, among other features. The comparison is based on the implementation of an application with a focus on reproductive capacity of multimedia content (video).

Keywords: Android, iOS, Smartphone, iPhone, Video

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

As vendas de *smartphones* no Brasil devem crescer 73% este ano, após o patamar recorde de 8,9 milhões de unidades comercializadas em 2011. Segundo projeção divulgada em 20/03/2012 pela consultoria IDC, especializada no mercado de tecnologia e telecomunicações, serão vendidos no Brasil perto de 15,4 milhões de unidades de *smartphones* em 2012.

As demandas do mercado consumidor vêm aquecendo o mundo da telefonia móvel, Numa pesquisa da consultoria IDC, as vendas de *smartphones* (telefone celular com funcionalidades avançadas) tem previsão de crescimento na ordem de 73% em 2012, no Brasil[1]. Entre 2010 e 2015 as vendas devem crescer três vezes e é esperado que o total do volume de vendas global deva ultrapassar o dos PCs (Computadores Pessoais) já em 2012 segundo o instituto Gartner (MACHILIS, 2011). Muitas das maiores empresas globais de tecnologia e comunicação vem sendo atraídas pelas oportunidades de negócios geradas por essa nova categoria. Englobar uma parcela do valor total gerado pela indústria de *smartphones*, para muitas dessas empresas é vista como uma chave para crescimento e lucro.

Notebooks, *netbooks* e telefones móveis estão sendo crescentemente absorvidos em vendas pelos *smartphones*, uma vez que estes unem características encontradas anteriormente em dispositivos distintos. Este setor ainda conta com outros dispositivos nesta mesma tendência, como *tablets* e leitores de *ebokds* (livros virtuais), se tornando fundamentais no universo móvel (KENNEY, 2011). A figura 1 mostra uma comparação entre as vendas de *smartphones* e PCs.

[1] "G1 Tecnologia. Disponível em http://g1.globo.com/tecnologia/noticia/2012/03/vendas-de-smartphones-no-brasil-devem-crescer-73-em-2012-diz-idc.html.

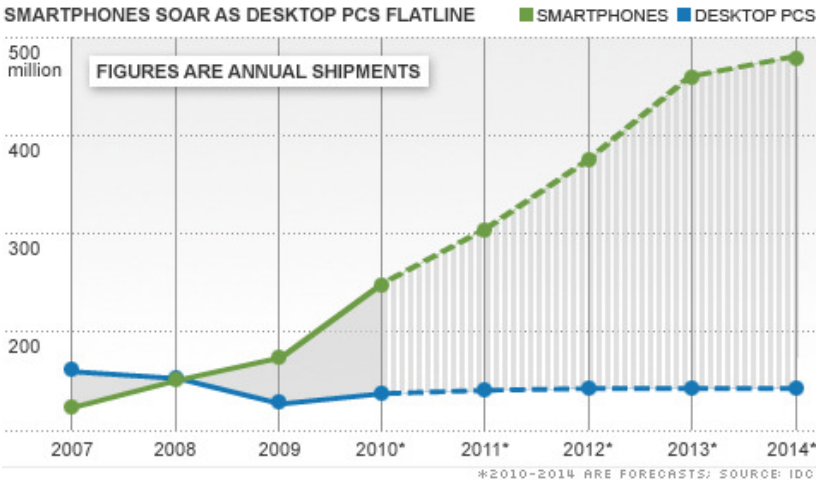

Figura 1 – Comparativo entre vendas de smartphones e PCs

Fonte: http://money.cnn.com/2010/07/20/technology/desktop_PC_death/index.htm

Com previsão do total de receitas dos serviços móveis, incluindo publicidade, inscrições, aparelhos, aplicativos e demais de ultrapassar US $ 1 trilhão em 2014, incluindo todos os dispositivos móveis disponíveis e tendo em vista a taxa que os *smartphones* adentram o mercado e com os preços dos componentes em declínio, até 2015 haverá, pelo menos, dois bilhões de dispositivos móveis inteligentes em uso no mundo.

A indústria e a natureza dos dispositivos móveis em si nos direcionam a uma análise a partir da perspectiva da plataforma tecnológica utilizada por cada dispositivo. Controle da plataforma foi identificado por estudiosos e consultores de gestão como um recurso chave no sucesso do negócio para as indústrias de Tecnologia e Comunicação (KENNEY, 2011). Aproveitando os seus estudos sobre Microsoft, Cisco e Intel, Michael Cusumano concluiu que o vencedor das competições tecnológicas geralmente aquele que tem a melhor estratégia de plataforma e os melhores ecossistemas para apoiá-la (CUSCUMANO, 2010).

Buscando conseguir a atenção dos usuários através de atrativos, existem muitas plataformas para dispositivos móveis no mercado atualmente. Com uma grande fatia de mercado e taxas expressivas de crescimento, algumas destas têm bastante sucesso. Habitualmente os usuários escolhiam aparelhos celulares de acordo com as suas

especificações de hardware e as aplicações suportadas pelos dispositivos ficavam a segundo plano. Atualmente nota-se uma mudança, onde os consumidores já demonstram um comportamento escolhendo os seus aparelhos não devido a sua câmera, mas sim pelo software que ele utiliza. Com uma tendência de popularização desta atitude se o sucesso de cada plataforma dependerá muito da quantidade de aplicações disponíveis para ela. A figura 2 mostra um gráfico com os números de aplicações disponíveis para as principais plataformas móveis entre Junho de 2009 e Março de 2011.

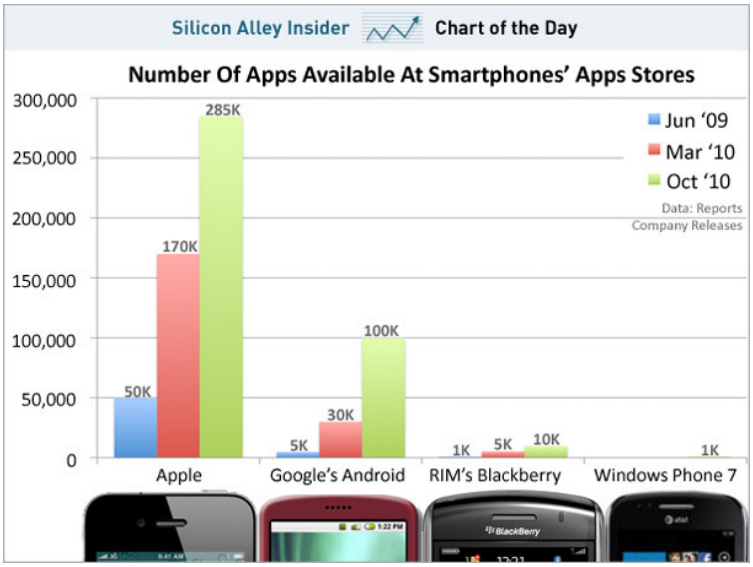

Figura 2 – Número de aplicações disponíveis nas "App Stores" para smartphones
Fonte: http://www.businessinsider.com/chart-of-the-day-smartphone-apps-2011-3

1.1. OBJETIVOS

Este trabalho apresentará um estudo comparativo entre duas das principais plataformas de desenvolvimento para dispositivos móveis: *Google Android* e *Apple iOS*. Essa escolha é baseada no destaque que recebem atualmente, tanto em relação à sua força de crescimento como pela importância perante os desenvolvedores. As plataformas *iOS* e *Android* são o começo de uma nova geração de sistemas operacionais para *smartphones*.

O objetivo é destacar as principais características de cada plataforma e demonstrar como elas podem ser utilizadas através da criação de uma aplicação que explore os recursos de cada uma. Atentando para a nova conjunção do mercado, que disponibiliza inúmeras ferramentas, com características que fornecem possibilidades vastas aos usuários; se faz necessário o enfoque mais detalhado de apenas uma das recentes APIs desenvolvidas, a fim de analisar mais criteriosamente as plataformas.

A ausência de trabalhos acadêmicos com este foco, aliado a multidisciplinaridade que existe neste tema podem ser considerados como a motivação para o desenvolvimento deste trabalho,

O trabalho será baseado em um levantamento bibliográfico a fim de apurar as características dos dois sistemas operacionais, bem como características mercadológicas de ambos. Em um segundo momento, será desenvolvido um aplicativo para visualização de vídeos, uma vez que esta característica é bastante utilizada nas duas plataformas, a fim de poder mensurar as características de cada linguagem, determinando vantagens e desvantagens. Ao final concluir-se-á qual a plataforma mais adequada, ou se há um equilíbrio nos critérios utilizados.

Subdivide-se esta pesquisa nas seguintes partes:

- Introdução: apresentação do tema e das características que nortearão o trabalho;
- *Android*: serão descritas as características deste sistema operacional;
- *Iphone OS:* onde será feito o mesmo, para o sistema operacional concorrente;

- Estudo de caso e implementação: neste capítulo será abordado o desenvolvimento do aplicativo para visualização de vídeos, nas duas plataformas;
- Comparativo: através das percepções obtidas no desenvolvimento, serão tabuladas as características de cada plataforma;
- Conclusão: expostos os resultados, define-se um vencedor para o comparativo (se houver) e ainda sinaliza possíveis abordagens para futuros trabalhos deste tema.

2. ANDROID

O *Android* é uma plataforma para *smartphones*, baseada no sistema operacional *Linux*, que possui diversos componentes, com uma variada disponibilidade de bibliotecas e interface gráfica, além de disponibilizar ferramentas para a criação de aplicativos (LECHETA, 2009).

2.1. HISTÓRICO

Uma pequena empresa chamada *Android Inc*., sediada em Palo Alto, Califórnia, Estados Unidos foi o inicio de tudo. Com a ideia de desenvolver um sistema operacional baseado em *Linux* para telefones celulares e outros dispositivos móveis, atraiu o interesse do *Google*, o qual a comprou em julho de 2005, ainda mais por ser parte da idéia fazer um sistema operacional não para um único fabricante de hardware, mas um sistema que qualquer um pudesse ter a licença e colocar em seu aparelho e que fosse flexível e atualizável (HILL, 2010).

Com a visão de permitir ao usuário acessar a Internet e tirar proveito de informações sobre sua posição e mobilidade através de um sistema operacional para *smartphones* Andy Rubin, *Chief Executive Officer* (CEO) da *Android Inc.*, expressa claramente essa necessidade em uma entrevista para a revista *Bussiness Week* em 2003, onde afirma que havia um grande potencial no desenvolvimento de dispositivos móveis inteligentes que estão mais conscientes da localização de seu proprietário e de suas preferências (MACHILIS, 2011).

A partir daí, o *Android* teve sua criação se deu através de um consórcio de mais de 40 empresas do setor de tecnologia e comunicação, sob o nome *de Open Handset Alliance*, liderado pelo *Google Inc*., com os objetivos principais: (LECHETA, 2009):

1. Possibilitar a personalização de aplicações e componentes do sistema, por serem de código aberto e gratuito,

2. Baseado em uma plataforma moderna e flexível, criar oportunidades de desenvolvimento rápido e moderno de aplicações corporativas,

Nesse momento eles também lançaram a primeira versão do *Software Development Kit* (SDK) do Android para o público[1]. Desde então Google e OHA lançaram algumas novas versões do SDK, que a partir de outubro de 2008 passaram a ser liberados sob a licença *open source Apache*. Sob a licença *Apache*, empresas privadas poderiam acrescentar ao Android seus próprios aplicativos e extensões e vendê-los, sem ter que submetê-los à comunidade open-source (BORT, 2008).

O primeiro aparelho rodando *Android* foi o G1 da T-Mobile também conhecido como *HTC Dream* ou *Google G1 dev phone*. Com tela capacitiva e teclado QWERTY completo foi lançado primeiramente nos Estados Unidos pela operadora T-Mobile[2].

Em 2009 houve uma arrancada no desenvolvimento de aparelhos utilizando *Android*. Disponibilizados por mais de sete fabricantes incluindo HTC, Motorola, LG, Sony Ericsson e Samsung, estes telefones já estavam disponíveis em praticamente todos os mercados[3].

O *Nexus One*, o primeiro "*Google Phone*", se tornou realidade no início de 2010. Marcado como o telefone do *Google*, é, entretanto fabricado pela HTC, que também fez o primeiro celular *Android*, o *HTC Dream*[4]. O *Nexus One* é vendido exclusivamente através da loja do Google e está disponível apenas em alguns países, incluindo Estados Unidos, Reino Unido e Singapura[5].

---

[1] " Open Handset Alliance. Industry leaders announce open platform for mobile devices."
[2] "HTC Inc. T-mobile unveils the t-mobile g1 - the first phone powered by android."
[3] "The Android Project. Developing on a device."
[4] Google Inc. Nexus one phone - feature overview & technical specifications. Disponível em: http://www.google.com/phone/static/en_US-nexusone_tech_specs.html.
[5] Google Inc. Availability in your country and language : Place an order - nexus one help. Disponível em: http://www.google.com/support/android/bin/answer.py?answer=166508

2.2. ARQUITETURA

Com o conhecimento da arquitetura da plataforma, de seus componentes principais, bibliotecas e subsistemas, podemos identificar as limitações e dependências na criação de um aplicativo para dispositivo móvel.

O *Android* é uma plataforma que inclui: Sistema operacional, *middleware*, e aplicativos.

Sua arquitetura é dividida em *Kernel*, *runtime*, Bibliotecas, *framework* e aplicativos, como mostrado na Figura 3 e melhor detalhado abaixo.

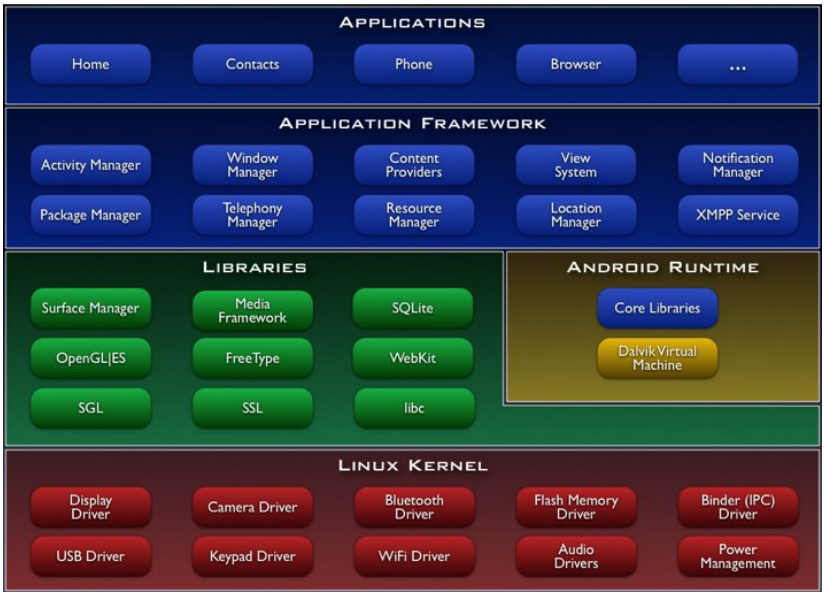

Figura 3 - Arquitetura Android

Fonte: http http://developer.android.com/images/system-architecture.jpg

Kernel: Em face da diversidade de arquiteturas de dispositivos móveis, assim como é em computadores e outros dispositivos eletrônicos, para que um sistema consiga operar com diferentes tipos de hardware, se fazem necessários os sistemas operacionais.

Segundo TANENBAUM (2003), um sistema operacional deve ser capaz de gerenciar o processador, memória e outros dispositivos de entrada e saída, além de fornecer, aos programas de usuário, uma interface mais simplificada com o *hardware*.

Nos *smartphones*, existe a demanda de gerenciar os recursos dos mesmos, tais como processamento e memória, sendo assim, há uma forte tendência de criação de sistemas operacionais com esse intuito.

Além disso, é necessário que a criação de *software* não contemple apenas um único modelo de celular, mas um determinado sistema operacional, ficando este responsável por gerenciar as particularidades de cada dispositivo.

O *Android* utiliza o *kernel* do *Linux*, que é responsável pelos serviços de segurança, gerenciamento de memória, processos, rede e *drivers*, este último componente é muito importante, pois garante que o desenvolvedor não precisará se preocupar em como acessar ou gerenciar dispositivos específicos do celular, produzindo assim uma abstração entre o *hardware* e o *software*.

Execução (Runtime): Da mesma forma que com o Java, uma aplicação *Android* é interpretada pelo sistema. A diferença é que o *Android* gera códigos Dalvik Executáveis (.dex), e não os *byte-code* (.class) do Java, tais código são interpretados pela Máquina Virtual Dalvik (MVD) (PROJECT, 2012).

Criada por Dan Bornstein, a MVD é uma alteração da Máquina Virtual Java (JVM), a qual é otimizada para os objetivos que o Android visa suprir (BORNSTEIN, 2008).

Após a compilação, todos os arquivos .dex e outros recursos utilizados pela aplicação (exemplo: imagens, sons, etc.), são compactados em um arquivo do tipo .apk (*Android Package File*), sendo este arquivo, a aplicação finalizada e pronta para ser distribuída e instalada em qualquer dispositivo com Android, (LECHETA, 2009).

A MVD também é incluída no SDK do *Android*, onde transformam os códigos das classes Java (.class) em códigos Dalvik executáveis (.dex), que posteriormente serão executado pelo emulador Android (BORNSTEIN, 2008).

Bibliotecas (Libraries): Atividades como a abertura de arquivos, a realização de cálculos aritméticos, exibição de imagens, entre outras, são atividades básicas, que acabam não sendo o foco principal do desenvolvedor.

Temos as bibliotecas, para se evitar a reescrita dos códigos de funções e rotinas que se repetem ou possuem altas complexidades. As bibliotecas possuem códigos e dados que

auxiliam na execução de serviços e permite a separação de partes do programa (modularização).

O Android possui um conjunto de bibliotecas, disponíveis para a criação de seus aplicativos como a *System C Library, Media Libraries, Surface Manager, Lib Webcore, SGL, 3D libraries, Freetype* e *SQLite.* Tais bibliotecas permitem a manipulação de vídeos, imagens, sons animações, banco de dados, etc.

- *System C Library*

  Considerando que o *Android* se baseia no Linux, parece coerente que exista uma biblioteca que defina as chamadas ao sistema. Suportando padrões *ISO C* e *POSIX*, com apoio às variantes do *Unix* como *BSD* e *System V*, no *Android* a biblioteca é uma BSD (*Berkeley Software Distribution)* do padrão C (Foundation, 2007).

- *Media Libraries*

  Para a reprodução e gravação de mídias de formatos populares, temos a media *libraries*, que é baseada em *Open CORE*. Dentre os formatos suportados, se incluem os formatos MPEG4, H.264, MP3, AAC, AMR, JPG e PNG. Esta biblioteca e desenvolvida pela PacketVideo's, (PACKETVIDEO, 2007).

- *Surface Manager*

Biblioteca que gerencia o aceso ao subsistema de vídeo do dispositivo, sendo capaz de compor gráficos em 2D e 3D a partir de aplicações de múltiplas camadas.

- *Lib WebCore*

Nascido como uma ramificação do *KHTML* e *KJS* do *KDE3*, O *WebCore* é um motor de navegação Web, que possui código aberto, utilizado em diversas aplicações de diversos sistemas operacionais. (NOKIA, 2004)

- Bibliotecas 3D

Baseada em *OpenGL ES* 1.0 as bibliotecas usam, tanto aceleração por hardware na medida do possível, quanto renderização 3D por software, sendo neste caso altamente otimizado,.(PROJECT, 2012).

- *FreeType*

Gerenciador de fontes de código aberto, o qual facilita a renderização de fontes *TrueType*, é e otimizado para diversos tipos de plataformas, principalmente nos casos sistemas incorporado ou portáteis (Turner et al., 2006).

- *SQLite*

SQLite é um sistema gerenciador de banco de dados (SGBD) embutido no Android, de código aberto, que não possui servidor de processos separado.

O *SQLite* é um gerenciador compacto e disponível para várias plataformas, foi desenvolvido para ocupar o mínimo de memória possível, mas com grande quantidade de recursos, permite criação de várias tabelas, trigers, índices e views, tudo em apenas um arquivo. É indicado para dispositivos que possuem limitações de memória e processamento, como os celulares, por exemplo. (LECHETA, 2009)

Framework de aplicações (Application Framework): Os desenvolvedores têm a sua disposição diversos outros componentes que o *Android* disponibiliza além das bibliotecas, como, o provedor de conteúdo, gerenciador de janela, telefone, recursos, atividades e muitos outros.

Sendo um diferencial do Android em relação a outras plataformas para dispositivos móveis, a possibilidade de acesso e modificação dos componentes permite que as aplicações criadas possam interagir com todo o sistema do celular, podendo o usuário/desenvolvedor, alterar qualquer componente que faça a parte do sistema para que esse se adeque as suas necessidades ou as do aplicativo que se está desenvolvendo.

Os componentes permitem a interoperabilidade entre os vários subsistemas do celular. Por exemplo, um aplicativo pode acessar a agenda de contatos e realizar uma chamada, capturar uma imagem pela câmera, descobrir qual a localização do celular, etc. (PROJECT, 2012).

O *framework* de aplicativo fornece componentes que auxiliam na implementação dos programas, permitindo a criação de listas, grades, caixas de texto, botões, etc. Alguns componentes se destacam, tais como:

- Provedor de conteúdo, que gerencia o acesso aos dados realizados pelos aplicativos;

- Gerenciador de notificações, que habilita os aplicativos a exibirem informações e avisos na barra de status do aparelho, assim como a reagirem às notificações recebidas;
- Gerenciador de atividades, que gerencia o ciclo de vida das aplicações e permite sua execução em segundo plano.

2.3. HARDWARE

Devido à flexibilidade do sistema operacional e ao uso deste por diversas empresas, o hardware é bastante diversificado, conforme o nicho de mercado que se pretende atuar.

2.4. GERENCIAMENTO DE THREADS, INTERRUPÇÕES, COMUNICAÇÃO ENTRE PROCESSOS, CHAMADAS DE SISTEMA

O Android possui um sistema bastante sofisticado de gerenciamento de threads. Possui uma thread única nas aplicações padrão e interface com o usuário, ou seja, que todas as tarefas de longa duração dentro de um aplicativo devem rodar em uma thread em segundo plano[1].

Os aplicativos no sistema operacional Android são o resultado da interação de quatro blocos de construção. São eles: atividades (*Activities*), ou todos os elementos da interface do aplicativo; serviços, que são tópicos que realizam todo o trabalho de fundo necessário à aplicação para executar sua tarefa; receptores de transmissão (*Broadcast Receivers*), que são essencialmente os ouvintes que permitem que um aplicativo possa responder à eventos do sistema ou de aplicações; e fornecedores de conteúdos (*Content Providers*), que são conjuntos de dados que a aplicação disponibiliza para outras aplicações. O motivo para essa divisão é o incentivo à pratica de reuso de componentes[1].

[1] Android Developers. Disponível em: developer.android.com

Na maioria dos casos, um aplicativo será baseado em uma atividade (*Activity*) que está sendo exibida e a infra-estrutura em segundo plano para realizar as tarefas de ligadas a essa atividade (BURKE, 2009).

Interrupções, comunicação entre processos e chamadas de sistema necessárias para fazer uma aplicação são lidadas pelo sistema com o uso extensivo de eventos. Funciona assim: uma aplicação dispara os eventos *onCreate*, *onStart* e *onResume* quando é iniciada. Então se em algum momento uma outra janela cobre qualquer parte da aplicação o evento *onPause* é disparado. Uma vez acionado o evento *onPause*, se toda a aplicação não está mais visível, o evento *onStop* pode ser chamado seguido pelo *onDestroy,* fechando totalmente a aplicação. No entanto, *onStop* e *onDestroy* nem são chamados se os recursos do sistema estiverem escassos, pois neste caso o sistema operacional pode simplesmente matar a aplicação. A figura 4 mostra o ciclo de vida de uma atividade em Android.

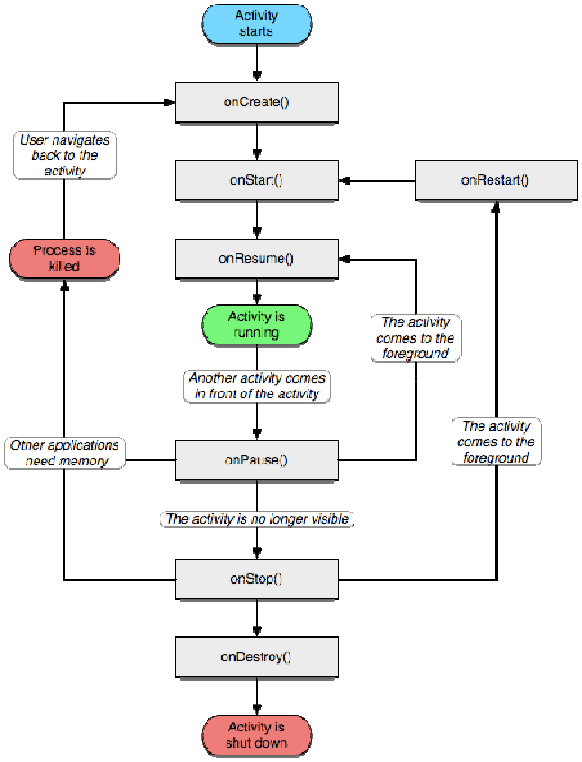

Figura 4 – Ciclo de Vida de uma Atividade em Android

Fonte: http://developer.android.com/images/activity_lifecycle.png

Seguindo este mesmo conceito básico de criação de eventos e disparos destes nos momentos adequados, temos o método utilizado pelo sistema operacional Android para transmitir dados entre threads em um único aplicativo. Um provedor de conteúdo (*Content Provider*) é requerido caso precise ser transmitida informações entre aplicações.

Chamadas do sistema são um pouco mais complexas. Algumas delas são feitas através das mesmas interfaces que são fornecidas aos aplicativos. Outros são feitos por meio de consulta classes que são *built-in*, por exemplo, a classe android.location.Location que dá acesso às coordenadas GPS atual.

2.5. GERENCIAMENTO DE MEMÓRIA

O uso de memória é geralmente limitado a 16 MB em aplicações Android. Essa característica se dá pela capacidade contida em um dispositivo móvel. Quanto mais aplicações o Android puder manter na memória, mais rápido ele será para o usuário quando este precisar trocar entre aplicações. Então, as aplicações devem preferencialmente usar o mínimo de memória possível para garantir várias aplicações rodando ao mesmo tempo sem que sejam finalizadas.

Os aplicativos Android são escritos na linguagem Java, os códigos são compilados com os arquivos de recursos e configuração gerando um arquivo do tipo .apk, este arquivo pode ser instalado através da ferramenta chamada AAPT (*Android Asset Packaging Tool*), que faz o gerenciamento dos pacotes e instala os arquivos .apk no sistema (GOOGLE INC., 2012).

Cada aplicativo Android é executado em seu próprio processo Linux, e cada um tem sua própria máquina virtual. Um aplicativo recebe por padrão uma ID exclusiva, como os aplicativos Linux, sendo assim cada processo é invisível a outro. É possível que mais de uma aplicação compartilhe o mesmo ID, permitindo assim o compartilhamento dos mesmos recursos, arquivos e máquina virtual.

Além disso, não requer nenhum código para alocar ou desalocar memória já que possui o *garbage collector* que gerencia essa questão de alocamento e desalocamento de memória automaticamente.

O *garbage collector* é solicitado sempre que uma aplicação possuir um endereço de momória alocado que não possua nenhuma referência ativa apontando para ele. Por projeto, o *garbage collector* não remove aplicativos que ainda estão sendo usados, e pode haver uma perda de desempenho do sistema enquanto o recurso estiver sendo executado.

Android conta com gerenciamento automático de memória que é gerenciado pelo *garbage collector*, no entanto o *garbage collector* às vezes pode causar problemas de desempenho se a alocação de memória não é tratada com cuidado. O Android SDK também fornece o *allocation tracker*, uma ferramenta para evitar freqüentes ações do *garbage collector*.

2.6. GERENCIAMENTO DE ENERGIA

Desenvolvido a partir da premissa de que a CPU não deve consumir de energia se os aplicativos ou serviços não precisam de energia, o Android exige que os aplicativos e serviços solicitem recursos da CPU com "*wake locks*" através do framework de aplicações Android e bibliotecas nativas Linux. Se não há nenhum *wake lock* ativo, a CPU é desligada para poupar energia. A classe PowerManager é a forma acessível do *Android Framework* disponibilizar o gerenciamento de energia para serviços e aplicações[1].

[1] http://www.cs.uwc.ac.za/~mmotlhabi/apm.pdf

3. IPHONE OS

O iOS é o sistema operacional desenvolvido originalmente para o iPhone, e também usado em outros dispositivos movéis. O iOS é baseado no conceito de manipulação direta, utilizando toque, ou seja, a interação com o sistema operacional com o usuário é imediata, pois esse sistema possibilita ao usuário, com gestos, toques na tela, deslizar dos dedos, por exemplo, ampliar fotos, reduzir imagens, digitar mensagens em teclado virtual, etc.

Por essa praticidade, o sistema, em seu lançamento com o primeiro iPhone, foi e ainda é um sucesso entre os usuários.

Neste capitulo, abordaremos a historia e a técnica sobre o sistema operacional móvel "iOS".

3.1 HISTÓRIA DO SISTEMA OPERACIONAL

Em meados de 2007, Steve Jobs, o CEO da Apple, estava no palco junto com a Cingular, agora departamento de dispositivos móveis da AT&T, apresentando o iPhone da Apple, um dispositivo inovador e um paradigma para a industria. Mas para entendermos sobre o dispositivo, precisamos voltar ao anuncio do primeiro iPod por Steve, cinco anos antes da apresentação do iPhone.

O iPod se tornou um dos maiores sucessos da Apple em 2004 e após alguns anos com os fabricantes do *Mp3 – player* a concorrencia aumentou, bem como celulares , que apresentavam tocadores de musicas embutidos, e para enfrentar esta competição, a Apple se uniu com a Cingular para criar o que viria a ser o Motorola ROKR, sendo sucesso da Motorola. RAZORseries.

A parceria foi criada de modo que a Motorola construiria completamente o telefone enquanto a Apple poderia se concentrar em escrever o software de música. O resultado foi um telefone que era defasado em comparação com a concorrência, não era visto como esteticamente agradável e só podia armazenar 100 músicas (VOGELSTEIN, 2008).

A Apple entendeu que se fosse para criar um celular de música de sucesso, eles teriam que fazê-lo por conta própria e para isso, Steve Jobs começou a procurar uma operadora de

telefonia nos EUA que aceitasse seu negócio. A Verizon recusou e a Apple acabou de volta à AT&T. As negociações foram difíceis devido à nova situação e as demandas de Steve Jobs, mas eles conseguiram terminar um acordo. O acordo previa que a AT&T receberia exclusividade para o dispositivo mais uma percentagem de todas as vendas de músicas para o dispositivo. Em troca, a Apple iria receber US $ 10 por cliente e mês e teria liberdade para criar seu próprio produto (VOGELSTEIN, 2008).

A Apple sabia como fazer o sistema operacional e a interface de trabalho, mas eles não tinham competência para projetar antenas e outros componentes móveis. Ainda assim, eles decidiram fazê-lo por conta própria e, sob extremo sigilo, desenvolveram todo o equipamento necessário. A segurança era tão apertada que para as pessoas que trabalhavam no hardware foram dadas softwares com código dummy (código que não é usado no produto final) para testar sobre eles, e as pessoas que trabalham no software só o viam rodando em uma grande caixa preta com uma tela. O mesmo era verdade para os engenheiros da AT&T que deveriam testar o dispositivo na rede da AT&T, mas só foram entregues simuladores contendo o processador de banda base e antena (VOGELSTEIN, 2008).

O produto resultante era caro, funcionou apenas na rede da AT&T EDGE, não poderia pesquisar através em e-mails e o browser não tinha nem Java nem suporte a Flash. Nada disso importava, os clientes adoraram. Desde aquele dia em 2007, todos os novos telefones liberados foram comparados com a versão atual do iPhone da Apple (VOGELSTEIN, 2008).

A Apple continuou a desenvolver o seu conceito de sucesso lançando novas versões do sistema operacional, bem como novas versões do hardware e com os lançamentos do iPhone 3G e 3GS e mais tarde do iPhone 4, a Apple se tornou o padrão de fato com o qual qualquer outro é julgado contra. Eles também aproveitaram o sucesso do iPhone para o iPod através da criação do iPod Touch.

Na história mais recente eles deram mais um passo importante e lançaram o iPad, um tablet baseado na arquitetura do iPhone. Um tablet é algo que a Apple tinha tentado criar internamente por um longo tempo e rumores sempre circularam.

3.2 ARQUITETURA

No iPhone OS, a arquitetura por baixo do sistema e muitas das tecnologias são semelhantes às encontrados no Mac OS X. O *kernel* no iPhone OS é baseado em uma variante do mesmo kernel base que é encontrado no Mac OS X. No topo deste *kernel* estão as camadas de serviços que são utilizadas para implementar aplicações na plataforma. A figura 5 mostra uma visão geral de alto nível dessas camadas.

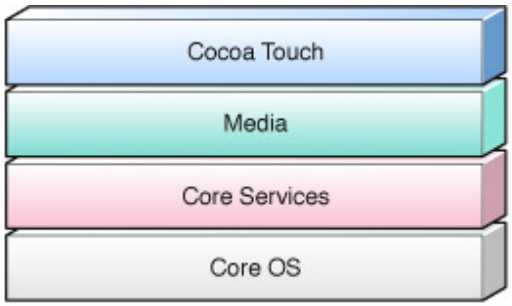

Figura 5 – Camadas do iOS

Fonte: http://developer.apple.com/library/ios/referencelibrary/GettingStarted/URL_iPhone_OS_Overview/Art/overview_systemlayers.jpg

Core OS: É a camada mais baixa do sistema operacional do iPhone. Ele consiste em uma versão reduzida do núcleo do Mac OS X otimizado para dispositivos móveis[1] [18]. O iPhone OS é um sistema *Unix* com o modelos de entrada e saída padrões, *POSIX threads e sockets BSD*. Ela também contém serviços para gerenciamento de energia em dispositivos móveis, segurança e *Bonjour*, que é um sistema sem configuração de descoberta de dispositivos e serviços em redes IP[2] [20].

Core Services: Esta camada fornece os serviços de sistema fundamentais dos quais todas as aplicações fazem uso. Um deles é o *Core Foundation*, que inclui uma interface baseada em C para coleções compostas de listas e dicionários, bem como strings e strings mutáveis. A camada de serviços do *Core* também contém *frameworks* incluindo o

[1] Apple Inc. iphone os technology overview: iphone os technologies.

[2] Apple Inc. Networking - bonjour.

CoreData, que é um *framework* para gerenciar os modelos de dados em uma aplicação, e o *CoreLocation* para gerenciar os dados de localização disponíveis a partir de GPS, triangulação e WiFi . Ele possui também a interface de baixo nível para *SQLite* e *parser* de XML (KENEDDY, 2011).

Media: A camada de mídia contém os gráficos, áudio, vídeo e tecnologias cujo objetivo é criar a melhor experiência multimídia disponível para o dispositivo móvel. A maioria das funcionalidades da camada Media e acima usam o *framework Core Graphics* para desenho. Este *framework* é também conhecido como *Quartz* e é baseado na mesma API de desenho baseada em vetores usada no Mac OS X. No topo desta camada fica a *Core Animation* que fornece uma interface de alto nível para configurar animações e efeitos. A camada de mídia também fornece reprodução de vídeo para os desenvolvedores de aplicações. O *framework* suporta a reprodução de arquivos de filmes .mov, mp4, m4v e gravações 3gp. O áudio pode ser reproduzido e emitido através do *framework Core Audio* e os *frameworks OpenGL ES e OpenAL* são regularmente utilizados no desenvolvimento de jogos (KENEDDY, 2011).

Cocoa Touch: É a camada contém os principais frameworks para a construção de aplicações iOS. Ela pode ser dividida em duas sub-camadas. O nível mais baixo consiste das partes que não são interface do usuário na camada *Cocoa Touch*. Este é o *Foundation Framework*, que é um subconjunto do *Foundation* presente na camada *Cocoa* do Mac OS X, e contém *wrappers* de objetos de *strings* e coleções da camada *Core Services*, bem como alguns outros serviços do sistema incluindo acesso ao sistema de arquivos e APIs de rede (KENEDDY, 2011).

A parte superior da *Cocoa Touch* é o *framework UIKit*, que contém toda a infra-estrutura de aplicativos e todos os componentes gráficos. A parte superior do *Cocoa Touch* também inclui manipulação de eventos, gráficos, janelas, texto e gerenciamento de web. O *framework* UIKit também permite ao desenvolvedor acesso para algumas das interfaces de hardware, incluindo a câmera, acelerômetro e outros sensores no iPhone (KENEDDY, 2011).

3.3 HARDWARE

O iOS é usado nos dispositivos iPhone, iPod Touch e iPad.

O iPhone 4, lançamento mais recente da Apple entre os iPhones, possui um processador *ARM Cortex A8* com *clock* de 1GHz e 512MB de memória RAM. Ele pode ser adquirido em duas versões, uma com 16 GB e outra com 32 GB de espaço para armazenamento interno. Não existem *slots* para expansão de momória neste aparelho. Sua tela de 3,5 polegadas possui *display LED-backlit IPS TFT* capacitivo com suporte à multi-toque e resolução de 640x960 pixels. Estão presentes também acelerômetro, GPS, câmera com 5 megapixels e flash de LED capaz de gravar vídeos em resolução 720p e 30fps[1]

3.4 GERENCIAMENTO DE THREADS, INTERRUPÇÕES, COMUNICAÇÃO ENTRE PROCESSOS, CHAMADAS DO SISTEMA

O iPhone OS fornece todas as tecnologias de gestão de *threads* que são agora consideradas padrão. É possível disparar threads e sincronizá-los usando todas as tecnologias usuais como Exclusão Mútua, fechaduras para leitura/escrita, travas distribuídas, etc. [2]

No entanto a Apple não recomenda o gerenciamento de threads desta maneira. Eles sentem que a programação direta de threads é muito difícil e pode ser feito muito mais eficiente, permitindo que o sistema operacional para lidar com o gerenciamento de *threads*. O recomendo é usar filas de operação. São as filas para as quais você atribui tarefas e o sistema operacional lida com o trabalho de threads necessários para que essas tarefas sejam executadas. Isso permite que o sistema operacional lide de forma mais eficiente com o carregamento de threads e o processamento de tarefas¹. [17].

O iPhone OS permite dois métodos diferentes de lidar com interrupções. Primeiro, há o protocolo *UIApplicationDelegate* que permite um aplicativo seja notificado de uma

[1] GSMArena.
[2] Apple Developer. Disponível em: developer.apple.com.

variedade de atividades e possa tomar as medidas adequadas para cada uma delas. Algumas das ações que podem ser respondidas são término de carregamento, alerta de pouca memória, mudança de orientação, etc. O segundo método é um construtor de nível inferior que pode ser mais complicado de usar. É a classe *NSNotification*. Esta classe permite-lhe ser notificado de quaisquer interrupções ou outras atividades personalizadas que ocorrem dentro do Sistema Operacional. No entanto para aproveitar isso, é preciso estar ciente das atividades que se deseja encontrar e estas devem estar presas especificadamente[1] [17].

A comunicação entre processos dentro do sistema operacional do iPhone é tratada usando manipuladores de URL personalizadas. Essencialmente a razão para isto é que o sistema operacional permite apenas uma aplicação do usuário ativa a cada momento, então aplicações devem se comunicar usando URLs personalizadas, procurando-as especificamente através do protocolo *UIApplicationDelegate* [1].

O iPhone OS proíbe chamadas de sistema. Em alguns casos específicos, chamadas são permitidas se controladas através da biblioteca *libSystem*, porém na maioria do sistema, chamadas simplesmente não são permitidas [2][21].

3.5 GERENCIAMENTO DE MEMÓRIA

IPhone não possui *garbage collector* como o Android. O desenvolvedor é responsável por limpar as variáveis depois de usá-las, caso contrário ocorrerá vazamento de memória no programa. Apesar de classe *NSObject* ter uma ajuda de contagem para manter o controle de quantos outros objetos estão atualmente usando um objeto, isso não ocorre de forma automática e os desenvolvedores têm de ajustar por si só. A regra para gerir a memória é certificar-se de que o número de métodos próprios chamados em um objeto será igual ao

[1] GRIGSBY, D. "Apple Approved iPhone Inter-process Communication." Disponível em: http://www.mobileorchard.com/apple-approved-iphone-inter-process-communication/.
[2] GEBARG, L. "iPhone - is it possible to make system call." Stack Overflow: http://stackoverflow.com/questions/1315939/iphone-is-it-possible-to-make-system-call..

número de perdas de métodos próprios quando o programa tiver terminado a execução. Quando se cria ou copia um objeto, sua contagem é 1. Posteriormente, outros objetos podem expressar um interesse de propriedade em seu objeto, que incrementa sua contagem. Os proprietários de um objeto também podem renunciar à sua participação no mesmo, o que diminui sua contagem. Quando a contagem torna-se zero, o objeto é desalocado (destruído).

3.6 GERENCIAMENTO DE ENERGIA

IPhone não possui o kit de ferramentas de gerenciamento de energia que o Mac OS possui. Em vez disso, esta função é incorporado na camada de núcleo (*Core Layer*). Os sistemas que mais consomem energia no iPhone em ordem descrescente são o sistema 3G, *Wi-Fi*, sistema 2G, *Bluetooth* e GPS. Reduzir o número de aplicações alto consumo de energia ajuda a economizar. Ao colocar um iPhone no modo *sleep*, ele vai desligar da rede, desligar o *Wi-Fi* e a luz tela.

4. ESTUDO DE CASO E IMPLEMENTAÇÃO

Este trabalho propõe um estudo comparativo entre as duas plataformas para dispositivos móveis mais evidentes na atualidade. A comparação será baseada no uso de uma característica comum para ambas as plataformas: a reprodução de videos. Para realizar esse estudo foram desenvolvidas duas aplicações parecidas usando Android e iOS.

O aplicativo a ser desenvolvido consiste em uma versão simplificada de um player para videos. Nessa implementação será possível analisar as ferramentas disponíveis para desenvolvimento nas duas plataformas em questão, assim como as APIs disponíveis para acesso aos sensores, complexidade de código e documentação.

4.1.DESENVOLVIMENTO EM ANDROID

Abordado anteriormente, os aplicativos para Android nada mais são do que aplicativos Java que passaram por uma otimização, se transformando em *bytecodes Dalvik*. (LECHETA, 2009)

A aplicação desenvolvida é composta por uma única tela exibindo assim uma única interface com o usuário. Assim, apenas um componente do tipo *Activity* se fez necessário. Nessa *Activity* principal será instanciada um objeto da classe *VideoView,* o qual receberá os atributos com o caminho completo para o vídeo e os controles que utilizará.

Neste caso, pela simplicidade do programa, não há necessidade da interface da *Activity* ser definida a partir de um arquivo XML contendo os componentes da mesma.

O código-fonte desta aplicação se encontra na figura abaixo:

```
1  public class ExemploVideoView extends Activity {
2      @Override
3
4      public void onCreate(Bundle b) {
5          super.onCreate(b);
6          videoView v = new VideoView(this);
7          setContentView(v)
8          v.setVideoPath("/sdcard/exemplo.mp4");
9          v.setMediaController(new MediaController(this));
10         v.requestFocus();
11     }
12 }
13
```

Figura 6 – Código-fonte completo do aplicativo

As capturas de tela da aplicação modelo finalizada rodando na plataforma Android pode ser vista na figura 13.

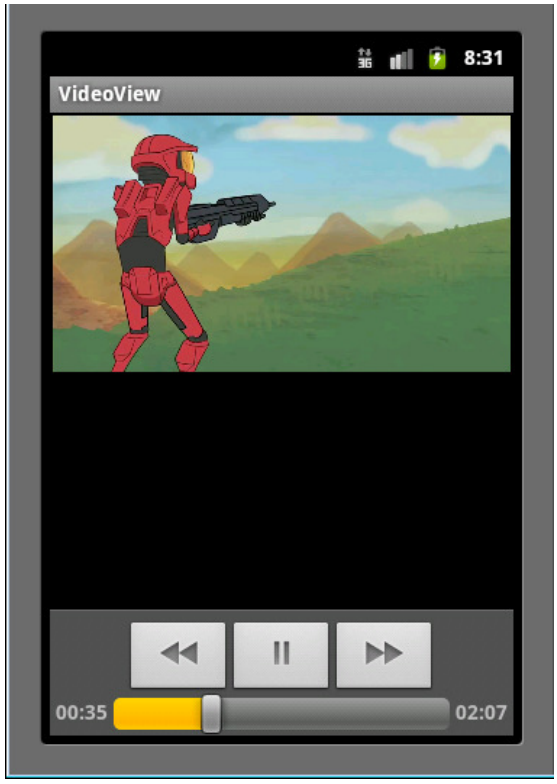

Figura 7 – Capturas de tela da aplicação modelo em Android

4.2.DESENVOLVIMENTO EM IOS

Os aplicativos desenvolvidos para iOS são escritos na linguagem de programação Objective-C. Esta é uma linguagem orientada à objetos que consiste basicamente da união do estilo de mensagens do *Smalltalk* com a linguagem de programação C pura. Ela é a principal linguagem usada pela *Cocoa API* da Apple e conseqüentemente é usada para desenvolver tanto aplicativos para Mac OS X quanto para iOS.

O aplicativo desenvolvido como modelo para o iOS é exatamente igual ao descrito na seção anterior para Android.

Neste caso, temos a classe *MediaPlayerMPMoviePlayerController* que instancia o objeto. O que parece um pouco mais complicado (e na verdade é), é a inserção dentro do método *viewDidLoad* a chamada para acionar (play) no vídeo e uma outra função para remover da tela e liberar o objeto com próprio evento de Callback do MediaPlayer.

```
#import "videoMP4ViewController.h"
#import <MediaPlayer/MediaPlayer.h>

@implementation videoMP4ViewController

- (void)dealloc
{
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    NSString *url = [[NSBundle mainBundle] pathForResource:@"super_mario_beatbox" ofType:@"mp4"];
    MPMoviePlayerController *player = [[MPMoviePlayerController alloc] initWithContentURL:[NSURL fileURLWithPath:url]];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(movieFinishedCallback:)
                                                 name:MPMoviePlayerPlaybackDidFinishNotification object:player];
    player.view.frame = CGRectMake(0, 0, 320, 200);
    [self.view addSubview:player.view];

    [player play];

    [super viewDidLoad];
}
```

Figura 8 – Código-fonte completo do aplicativo

```
27
28      [player play];
29
30      [super viewDidLoad];
31  }
32
33
34  - (void) movieFinishedCallback:(NSNotification*) aNotification {
35
36      MPMoviePlayerController *player = [aNotification object];
37      [[NSNotificationCenter defaultCenter] removeObserver:self name:MPMoviePlayerPlaybackDidFinishNotification object:player];
38      [player stop];
39
40      [self.view removeFromSuperview];
41
42      [player release];
43  }
44
45  - (void)viewDidUnload
46  {
47      [super viewDidUnload];
48      // Release any retained subviews of the main view.
49      // e.g. self.myOutlet = nil;
50  }
51
52  - (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
53  {
54      // Return YES for supported orientations
55      return (interfaceOrientation == UIInterfaceOrientationPortrait);
56  }
57
58  @end
59
```

Figura 9 – Código-fonte completo do aplicativo (continuação)

Codigo fonte com a aplicação para IOS

A figura 10 mostra a tela da aplicação modelo rodando na plataforma iPhone.

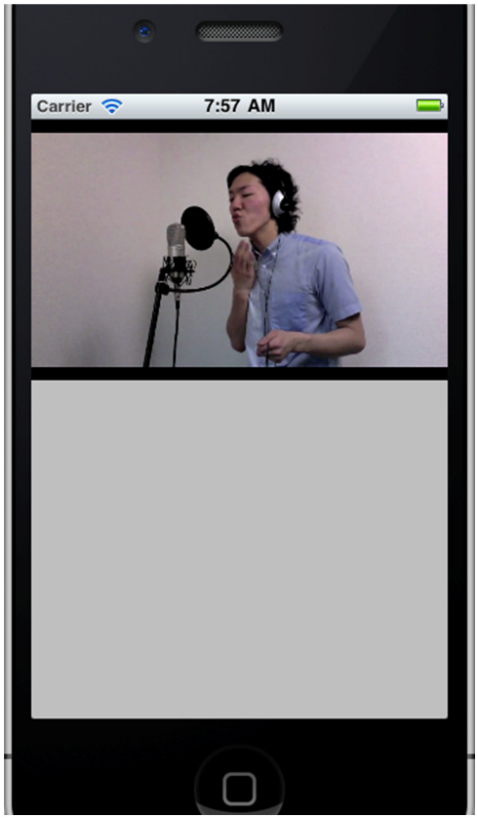

Figura 10– Captura de tela da aplicação modelo em iOS.

5. COMPARATIVO

Como mostrado no capítulo 4, foram implementadas duas aplicações com os mesmos requisitos, uma utilizando Android e a outra iOS. Ambas focavam a reprodução de vídeos em seu desenvolvimento. O objetivo do trabalho, que é o de comparar as duas plataformas, usou as seguintes métricas na avaliação: Ambiente de desenvolvimento, linguagens de programação e documentação. Serão apresentados também os principais pontos fortes e fracos encontrados nas duas plataformas.

5.1. AMBIENTE DE DESENVOLVIMENTO E LINGUAGEM DE PROGRAMAÇÃO

A IDE (*Integrated Development Environment*) XCode, desenvolvida pela Apple é a responsável pelo desenvolvimento para IOS, enquanto o desenvolvimento para Android é feito a partir da popular IDE Eclipse.

O XCode é uma ferramenta de desenvolvimento bem integrada ao ambiente Mac, por outro lado, o Eclipse, já é tido como um padrão da indústria com o qual muitos desenvolvedores já estão familiarizados. Por ser semelhante a outras IDEs já conhecidas, como o Visual Studio, facilita o aprendizado para novos desenvolvedores que aderirem à ferramenta. O fácil gerenciamento de projetos e a possibilidade de usar *plugins* já existentes para Eclipse para conectar-se a uma série de sistemas de controle de versões diferentes, como CVS e SVN podem ser considerados como as maiores vantagens desse ambiente de desenvolvimento para Android. A figura 11 mostra o ambiente de desenvolvimento Eclipse.

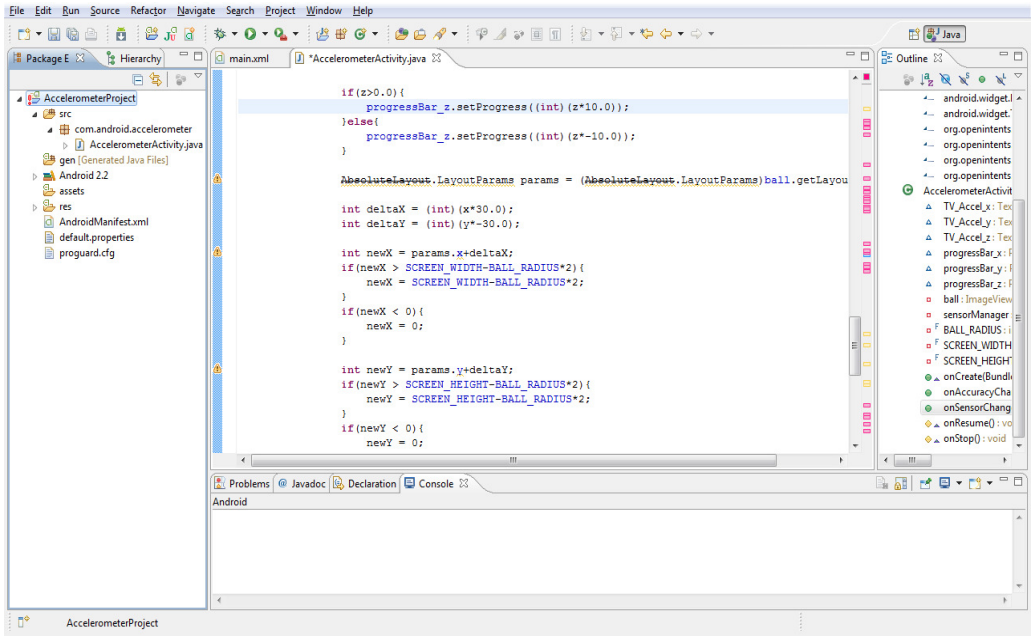

Figura 11 – Eclipse IDE

A comparação entre Objective-C e Java. Objective-C é inevitável. Objective-C assim como o XCode, são específicos para o ambiente Mac. Java, em contrapartida é umas das linguagens de programação mais usadas atualmente. A união entre o Java e o Eclipse configura um ambiente de desenvolvimento bem mais agradável, e Java é, em sua totalidade, uma linguagem de programação mais moderna.

O XCode, no entanto possui vantagens em relação ao Eclipse a exemplo das ferramentas externas feitas para facilitar o desenvolvimento de aplicações para iPhone. O *Interface Builder* e a ferramenta de análise de desempenho são muito melhores e mais bem integradas se comparadas ao *plugin* ADT do Android para Eclipse, que além de não apresentarem uma boa integração não possuem um acabamento no mesmo nível. A ferramenta correspondente ao *Interface Builder* na plataforma Android não possui uma interface tão amigável e falta a funcionalidade para vincular as ações na interface aos métodos no código, uma característica presente no *Interface Builder*. A figura 12 mostra uma imagem do ambiente XCode.

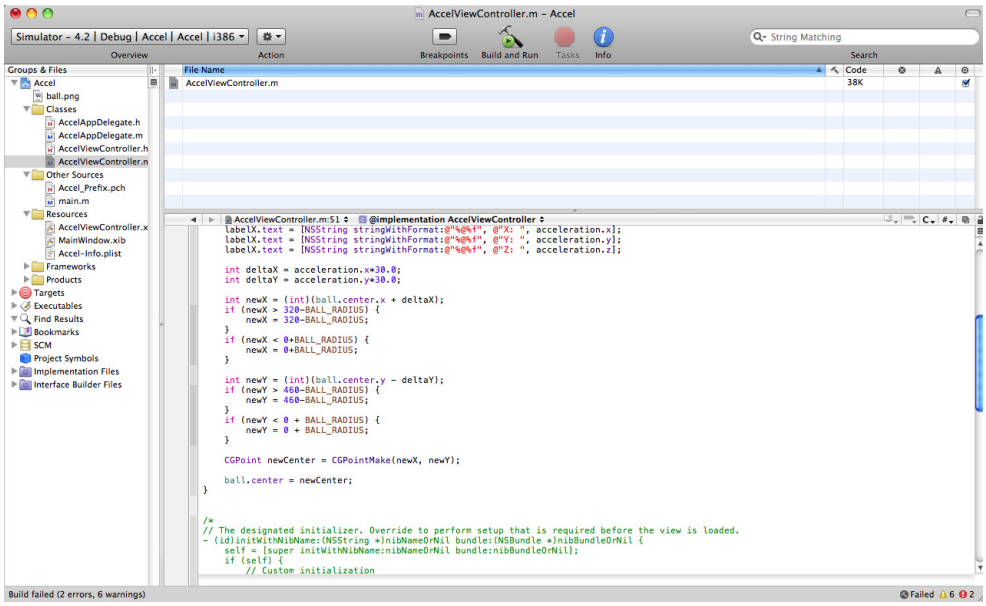

Figura 12 – Xcode IDE

Apesar de requerer um poder de processamento relativamente alto do computador onde está rodando, o emulador da plataforma Android é bem trabalhado. Possui muitas vantagens, entre elas o fato de ser construído sobre o QEMU, que é um software livre que emula um processador, e ainda emular um telefone real. O iPhone Simulator roda como uma aplicação normal e isso limita testar certos tipos de aplicação. O emulador Android possui muitas funcionalidades que faltam no iPhone Simulator, como a capacidade de simular chamadas ou mensagens de texto, alterar a posição de um GPS falso, que é simulado no emulador, assim como limitar a largura da banda e latência da rede para que condições reais possam ser simuladas.

5.2.DOCUMENTAÇÃO

A Apple fornece um bom acervo de fontes aos desenvolvedores que estão iniciando na plataforma iOS. A *iTunes University* oferece uma seção de vídeos para iniciantes onde os conceitos básicos da plataforma são explicados de forma bastante intuitiva, familiarizando assim o desenvolvedor com a plataforma. Os vídeos trazem desde os conceitos básicos até

os mais avançados para iOS. A Apple disponibiliza também um bom número de aplicações-modelo e códigos para demonstrar o uso da API para os usuários.

Para encontrar um material mais específico, a Apple disponibiliza a biblioteca de referência para iPhone (*iPhone Reference Library)*, que cobre todas as classes e protocolos usados na linguagem de programação Objective-C. Embora seja bastante semelhante as demais bibliotecas de referência, não apresenta a mesma clareza e intuitividade de algumas, a exemplo da oferecida aos desenvolvedores Android.

Para Android, a seção *developer* disponível na sua homepage traz um bom material pra quem está iniciando na plataforma. Nela são fornecidos vários guias de desenvolvimento que cobrem os principais conceitos da plataforma, além de um bom manual de referência sobre as APIs disponíveis. Os guias e as referências sobre a API são instalados junto com o Android SDK (*System Development Kit*), estão todos estão disponíveis offline.

O Android não oferece guias em forma de vídeos como a Apple faz para o iOS, porém seus recursos de desenvolvimento são bem mais organizados que os do iOS. Vale salientar que muitos projetos de código aberto Android podem ser usados como idéias sobre arquitetura e uso da API. Nesse ponto o Android sobressai, pois com a política de NDA (Non-disclosure agreement) da Apple não exixtem muitas fontes em termos de código aberto para iPhone.

5.3.PONTOS FORTES

O iOS claramente coloca uma grande ênfase na questão da experiência do usuário, área onde a Apple sempre se destacou. A ênfase no usuário torna o iOS muito fácil de se usar. Além disso, a plataforma e todos os dispositivos móveis lançados pela Apple se popularizaram em demasia na atualidade. Isso significa que grande parte das aplicações móveis estão sendo criadas para iOS. Isto gerou um ambiente que lhe permite fazer quase tudo que seria desejável em um dispositivo móvel através de um aplicativo para iPhone.

O iPhone também possui a vantagem de ter todo o seu espaço de armazenamento em uma única unidade. Logo as aplicações e o sistema operacional usam a mesma partição. Na

maioria dos outros *smartphones*, os sistemas operacionais são instalados em uma pequena partição primária e todo o restante é instalado em uma partição separada que geralmente fica em um cartão de memória. Porém, a maioria das aplicações é escrita sob a suposição de que elas serão executadas em uma única memória interna. Dispensar esse artefato de execução é valioso para o iPhone.

O sistema operacional Android tem uma grande vantagem em ser personalizável. Nesse sentido, há possibilidades ilimitadas para um dispositivo Android, sendo possível escrever quaisquer aplicações desejadas desde que o hardware do aparelho as suporte. A capacidade de executar tarefas em segundo plano é uma enorme vantagem. Há muitas funcionalidades que só podem ser conseguidas por meio de dois ou mais aplicativos rodando ao mesmo tempo e o Android permite que isso ocorra. Além disso, o sistema permite o uso de cartões de memória, que expandem o espaço de armazenamento e ampliam as possibilidades.

5.4.PONTOS FRACOS

O iOS possui o recurso de multitarefa bastante limitado. Antes da chegada o iOS 4, ele aplenas permitia que aplicativos nativos como Mail, iPod e telefone pudessem rodar em segundo plano. Com o iOS 4 a Apple disponibiliza aos desenvolvedores uma pequena e limitada lista de APIs que podem executar determinados serviços em segundo plano. Isso remove uma série de funcionalidades muito úteis das possibilidades da plataforma. O desenvolvimento para o iPhone também é limitado apenas à plataforma Mac. Fato que encarece o custo de entrada para aqueles que não possuem o hardware necessário. Além disso, o desenvolvimento para o iPhone requer aprendizagem de uma língua que não é valiosa fora da comunidade Apple.

O sistema operacional Android, por sua vez, já está a ter problemas com o fato de que os aplicativos não são sempre portáveis através dos dispositivos, com um número de desenvolvedores reclamando sobre como compatibilidade entre dispositivos é complexa (GRUMAN, 2010). É importante notar também que a interface para o usuário do

sistema operacional Android não é tão agradável e polida como a do iOS. Ela passa um aspecto mais áspero e não é tão intuitiva. Esta é uma área onde a Apple faz um trabalho muito melhor.

6. CONCLUSÃO E TRABALHOS FUTUROS

O propósito desse trabalho foi realizar um estudo comparativo entre as principais plataformas móveis disponíveis no mercado atual. Google Android e Apple iOS. A intenção não foi apontar a melhor plataforma de desenvolvimento de aplicações, mas tentar mostrar as forças de cada uma de forma a possibilitar uma escolha mais embasada para futuros trabalhos.

Para tanto, vimos detalhes técnicos e ambientes de desenvolvimento além de demonstrar superficialmente a criação de uma aplicação simples com foco na exibição de vídeos. No final pudemos formar algumas opiniões sobre as características das plataformas estudadas que estão condensadas na tabela 1.

CRITÉRIOS COMPARATIVOS	IOS	ANDROID	OPINIÃO
Linguagem de programação	Objective-C	Java	Java por ser mais difundida é uma linguagem melhor para se desenvolver. Objective-C encontra-se restrita aos dispositivos Apple.
Ambiente de desenvolvimento	XCode	Eclipse	XCode possui melhor integração e ferramentas para interfaces bastante superiores. Eclipse possui a vantagem de ser mais difundido.
Emulador	iPhone Emulator	Android Emulator	O emulador Android dispõe de mais funcionalidades para simular um dispositivo real.
Hardware	iPhone, iPad, iPod Touch	Dispositivos de vários fabricantes	A Apple possui a vantagem de ter configurações já definidas para rodar as aplicações. Como

			Android possui muitos aparelhos é difícil desenvolver aplicações compatíveis com todos.
Licença	Proprietária EULA	Apache 2.0 (Open Source)	O sistema Android pode ser acessado totalmente, possibilitando a criação de aplicações mais robustas.

Tabela 1 - Comparativo entre as plataformas

Os resultados obtidos nesse estudo mostraram a necessidade de trabalhos futuros que complementem essa comparação permitindo ainda mais avanços nesse setor. Devido ao curto espaço de tempo disponível e às restrições dos objetivos estabelecidos, não foi possível o aprofundamento em alguns aspectos. A análise da segurança das plataformas e o desempenho das aplicações em cada uma delas são temas que podem ser abordados em trabalhos futuros.

7. REFERÊNCIAS BLIOGRÁFICAS

Android Developers. Disponível em: <http://developer.android.com> Acessado em: 30/08/2012

Apple Inc. **iphone OS technology overview: iphone os technologies**. Disponível em: <http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html>. Acessado em 29/08/2012.

Apple Inc. **Networking bonjour**.Disponível em:<http://developer.apple.com/networking/bonjour/index.html>.Acessado em: 29/08/2012.

BORT, Dave. 2008. **Android is now available as open source**. <http://source.android.com/posts/opensource> Acessado em: 20/04/2010.

BORNSTEIN, D. (2008). **Dalvik virtual machine**. Disponível em: <http://www.dalvikvm.com/>. Acessado em 01/07/2012.

BOUCHAUD, J. - **Apple Becomes Second Largest Buyer of Consumer Cell Phone MEMS** Sensors in 2010. Disponível em: <http://www.isuppli.com/mems-and-sensors/news/pages/apple-becomes-second-largest-buyer-of-consumer-cell-phone-mems-sensors-in-2010.aspx>. Acessado em: 30/06/2012

BURKE, E.M. - **A Simple Android App and a Threading Bug**. Disponível em: <http://jnb.ociweb.com/jnb/jnbJan2009.html>. Acessado em: 03/07/2012.

CUSCUMANO, Michael. 2010. **Technology Strategy and Management: The Evolution of Platform Thinking.** Communications of the ACM. Disponível em: <http://dl.acm.org/citation.cfm?doid=1629175.1629189 > Acessado em 03/10/2012

FOUNDATION, F. S. (2007**). The gnu c library**. Disponível em: <http://www.gnu.org/software/libc>. Acessado em 18/10/2012.

G1 Tecnologia. **Vendas de smartphones no Brasil devem crescer 73% em 2012 diz IDC**. Disponível em <http://g1.globo.com/tecnologia/noticia/2012/03/vendas-de-smartphones-no-brasil-devem-crescer-73-em-2012-diz-idc.html>. Acessado em 20/11/2012

GEBARG, L. **iPhone - is it possible to make system call. Stack Overflow**: Disponível em: <http://stackoverflow.com/questions/1315939/iphone-is-it-possible-to-make-system-call>. Acessado em: 30/08/2012

Google Inc. **Nexus one phone - feature overview & technical specifications**. Disponível em: <http://www.google.com/phone/static/en_US-nexusone_tech_specs.html>. Acessado em: 27/08/2012.

Google Inc. **Availability in your country and language : Place an order - nexus one help**.Disponível em: <http://www.google.com/support/android/bin/answer.py?answer=166508>. Acessado em: 27/08/2012

GRIGSBY, D. "**Apple Approved iPhone Inter-process Communication."** Disponível em: <http://www.mobileorchard.com/apple-approved-iphone-inter-process-communication>. Accessado em: 30/06/2012.

HILL, Simon. 2010. **" History of Android: First Applications Prototypes & Other Events"** Disponível em < http://www.brighthub.com/mobile/google-android/articles/18260.aspx> Acessado em: 30/10/2012

"HTC Inc. **T-mobile unveils the t-mobile g1 - the first phone powered by android."** Disponível em: <http://www.htc.com/www/press.aspx?id=66338&lang=1033>. Acessado em: 27/08/2012

KENNEY, Martin. **Structuring the Smartphone Industry: Is the Mobile Internet OS Platform the Key?** ,Davis: University of California, 2011. Disponível em <http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1851686> Acessado em 20/11/2012

LECHETA, R. R. (2009). Google Android - **Aprenda a criar aplicações para dispositivos móveis** com o Android SDK. 1ª Edição, São Paulo: Editora Novatec, 2009

MACHILIS, Sharon. **Mercado móvel em números: iOS lidera tablets e android smartphones.** IDG Now, 09 Jun. 2011. Disponível em: <http://idgnow.uol.com.br/mercado/2011/06/09/mercado-movel-em-numeros-ios-lidera-tablets-e-android-smartphones> Acessado em: 30/06/2012

MOTHABI, MICHAEL B. (2010) **Advanced Android Power Management and Implementation of Wakelocks**. Disponivel em: <http://www.cs.uwc.ac.za/~mmotlhabi/apm2.pdf> Acessado em 10/11/2012

NOKIA, R. C. (2004). **Gtk+ webcore project**. Disponível em: <http://gtk-webcore.sourceforge.net/index.html>. Acessado em 11/10/2012.

Open Handset Alliance. Industry leaders announce open platform for mobile devices. Disponível em: <http://www.openhandsetalliance.com/press_110507.html>. Acessado em: 27/06/2012.

PacketVideo, C. (2007). **Opencore**. Disponível em: <http://www.packetvideo.com/resources/OpenCORE-brochure.pdf>. Acessado em 11/10/2012.

PROJECT, A. O. S. (2012). **Android open source.** Disponível em: <http://source.android.com>. Acessado em 18/06/2012.

SYMBIAN FOUNDATION. **The history of symbian.** Disponível em: <http://www.symbian.org/about-us/history-symbian>. Acessado em: 27/08/2012.

The Android Project. **Developing on a device**. Disponível em: <http://developer.android.com/guide/developing/device.html>. Acessado em: 27/06/2012

VOGELSTEIN, Fred. The untold story: How the iphone blew up the wireless industry. **Wired Magazine**, Setembro 2008.