

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANA - UTFPR  
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE  
SISTEMAS

VINÍCIUS RAMOS KAWAMOTO

**DESENVOLVIMENTO DE APLICAÇÕES PARA *ANDROID* UTILIZANDO O  
*FRAMEWORK ADOBE FLEX***

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA

2011

VINÍCIUS RAMOS KAWAMOTO

**DESENVOLVIMENTO DE APLICAÇÕES PARA *ANDROID* UTILIZANDO O  
*FRAMEWORK ADOBE FLEX***

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas - CSTADS - da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Me. Everton Coimbra de Araújo.

MEDIANEIRA

2011



---

## TERMO DE APROVAÇÃO

### DESENVOLVIMENTO DE APLICAÇÕES PARA *ANDROID* UTILIZANDO O FRAMEWORK *ADOBE FLEX*

Por

**Vinicius Ramos Kawamoto**

Este Trabalho de Diplomação (TD) foi apresentado às 13:50h do dia 28 de setembro de 2011 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. Os acadêmicos foram argüidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

---

Prof. Me. Everton Coimbra de Araújo  
UTFPR - *Campus* Medianeira  
(Orientador)

---

Prof. Juliano Rodrigo Lamb  
UTFPR - *Campus* Medianeira  
(Convidado)

---

Prof. Márcio Matté  
UTFPR - *Campus* Medianeira  
(Convidado)

---

Prof. Me. Juliano Rodrigo Lamb  
UTFPR - *Campus* Medianeira  
(Responsável pelas atividades de TCC)

## **AGRADECIMENTOS**

Primeiramente agradeço a Deus por dar-me forças em todos os momentos.

A minha família que sempre me apoiou em toda minha formação, sempre pensando unicamente em meu bem.

Ao professor, orientador e acima de tudo amigo Everton Coimbra de Araújo, pelos seus ensinamentos, sempre disposto a sanar as minhas dúvidas e pelos conselhos para que fosse possível o desenvolvimento deste trabalho nesta graduação.

A todos meus professores do curso que puderam me auxiliar em todos momentos desta graduação.

E a todos meus amigos e colegas que pude conhecer e trocar conhecimentos.

“O Homem é do tamanho do seu sonho.”

Fernando Pessoa

## RESUMO

Este trabalho apresenta um estudo detalhado sobre o framework *Adobe Flex* aplicado a plataformas móveis, demonstrando a utilização do mesmo especificamente para a plataforma *Android*. Neste estudo são descritos as especificações e o benefício da nova versão do *Adobe Flex* que fornece um modelo fácil e ágil para a criação de aplicações para dispositivos móveis. São apresentados através de uma aplicação prática o processo de desenvolvimento até a publicação do aplicativo no *Android Market*.

**Palavras Chaves:** *Android, Adobe Flex, Dispositivo Móvel.*

## **ABSTRACT**

This paper presents a detailed study on the Adobe Flex framework applied to mobile platforms, demonstrating its use specifically for the Android platform. This study is described in the specifications and benefits of the new version of the Adobe Flex model that provides a quick and easy to create applications for mobile devices by showing a practical application of the development process until the publication of the application on Android Market.

**Keywords:** *Android, Adobe Flex, Mobile Device.*

## LISTAS DE FIGURAS

Figura 1 - Integração RIA com várias plataformas.....	16
Figura 2 - Empresas que fazem parte da <i>OpenHandset Alliance</i> .....	23
Figura 3 - Gráfico relativo as versões mais utilizadas pelos clientes.....	25
Figura 4 - Adobe Flash Builder.....	31
Figura 5 - Tela de criação de um projeto móvel .....	33
Figura 6 - Arquitetura <i>Android</i> .....	35
Figura 7 - Arquitetura do <i>Adobe Flex</i> .....	38
Figura 8 - Caso de uso do Usuário .....	43
Figura 9 - Caso de uso Contatos .....	44
Figura 10 - Caso de uso coordenada ao contato .....	44
Figura 11 - Diagrama de atividade da aplicação .....	45
Figura 12 - Arquitetura da aplicação.....	46
Figura 13 - Definindo permissões no momento de criação da aplicação.....	50
Figura 14 - Tela de Login .....	52
Figura 15 - Mensagem de erro .....	52
Figura 16 - Cadastro usuário .....	53
Figura 17 - Lista de contatos vazia .....	54
Figura 18 - Cadastro de contato.....	55
Figura 19 - Lista de contatos .....	55
Figura 20 - Vincular coordenada ao contato .....	56
Figura 21 - Detalhes do contato .....	57
Figura 22 - Última posição do usuário no ultimo acesso.....	58



## LISTA DE QUADROS

Quadro 1 - Aumento dos Smartphones entre outubro de 2010 e janeiro de 2011 .....	21
Quadro 2 - Aumento dos smartphones no primeiro trimestre de 2011 .....	21
Quadro 3 - Versões do Adobe Flex .....	28
Quadro 4 - Linguagem MXML .....	29
Quadro 5 - Linguagem MXML e ActionScript.....	30
Quadro 6 - Estrutura do projeto Flex e Java .....	48
Quadro 8 - Arquivo de configuração <i>Hibernate.cfg.xml</i> .....	49
Quadro 9 - Linguagem MXML .....	49
Quadro 10 - Arquivo XML principal do BlazeDS.....	51

## LISTA DE ABREVIATURAS E SIGLAS

AIR	-	<i>Adobe Integrated Runtime</i>
API	-	<i>Application Programming Interface</i>
BSD	-	<i>Berkeley Software Distribution</i>
DHTML	-	<i>Dynamic Hipertext Markup Language</i>
DOM	-	<i>Document Object Model</i>
GSM	-	<i>Global System for Mobile</i>
HTML	-	<i>Hypertext Markup Language</i>
HTTP	-	<i>Hypertext Transfer Protocol</i>
IDE	-	<i>Integrated Development Environment</i>
IBM	-	<i>International Business Machines</i>
JEE	-	<i>Java Enterprise Edition</i>
MXML	-	<i>Magic Extensible Markup Language</i>
PDA	-	<i>Personal Digital Assistant</i>
RAD	-	<i>Rapid Application Development</i>
RIA	-	<i>Rich Internet Applications</i>
SDK	-	<i>Software Development Kit</i>
SWF	-	<i>ShockwaveFlash</i>
SQL	-	<i>Structured Query Language</i>
XML	-	<i>Extensible Markeup Language</i>

## SUMÁRIO

<b>1. INTRODUÇÃO .....</b>	<b>12</b>
1.1. <b>OBJETIVOS .....</b>	13
1.1.1. Geral.....	13
1.1.2. Específicos.....	13
1.2. JUSTIFICATIVA .....	13
<b>2. REFERENCIALTEÓRICO.....</b>	<b>15</b>
2.1. RIA - RICH INTERNET APPLICATION.....	15
2.1.1. Benefícios.....	17
2.1.2. Deficiências e Restrições .....	17
2.2. DISPOSITIVOS MÓVEIS .....	17
2.2.1. Vantagens da mobilidade .....	18
2.2.2. História <i>Smartphone</i> .....	18
2.2.3. Sistemas operacionais .....	20
2.2.4. Crescimento dos principais <i>smartphones</i> .....	20
2.3. A PLATAFORMA ANDROID .....	22
2.3.1. Versões.....	24
2.3.2. <i>Android SDK</i> .....	25
2.3.3. <i>Android Market</i> .....	25
2.4. ADOBE FLEX .....	26
<b>3. ARQUITETURA .....</b>	<b>33</b>
3.1. ANDROID.....	33
3.2. ADOBE FLEX .....	37
<b>4. MATERIAIS E MÉTODOS .....</b>	<b>39</b>
4.1. TECNOLOGIAS ENVOLVIDAS.....	39
4.1.1. Java.....	39
4.1.2. MySQL.....	39
4.1.3. MXML e <i>ActionScript</i> .....	39
4.2. ESTRUTURA LÓGICA.....	40
4.2.1. Aplicativos .....	40
4.2.1.1. Adobe Flash Builder.....	40
4.2.1.2. Eclipse.....	40
4.2.2. Servidor .....	41

4.3.	ESTRUTURA FÍSICA.....	41
<b>5.</b>	<b>RESULTADOS E DISCUSSÃO .....</b>	<b>42</b>
5.1.	ANÁLISE GERAL DO SISTEMA.....	42
5.1.1.	Requisitos.....	42
5.1.2.	Caso de uso.....	43
5.1.3.	Diagrama de Atividade.....	45
5.2.	ARQUITETURA.....	46
5.3.	CRIAÇÃO DO PROJETO .....	48
5.4.	ARQUIVOS DE CONFIGURAÇÕES .....	48
5.4.1.	Arquivo de configuração do <i>hibernate</i> .....	48
5.4.2.	Arquivo de configuração de permissão do Android.....	49
5.4.3.	Arquivo de configuração do BlazeDS .....	50
5.5.	TELAS.....	51
5.5.1.	Login.....	51
5.5.2.	Cadastro usuário.....	53
5.5.3.	Tela lista contatos .....	53
5.5.4.	Tela cadastro contato.....	54
5.5.5.	Vincular coordenada usuário. ....	56
5.5.6.	Detalhar contato .....	56
5.5.7.	<i>Google Maps</i> .....	57
<b>6.</b>	<b>CONSIDERAÇÕES FINAIS .....</b>	<b>59</b>
6.1.	CONCLUSÃO .....	59
	<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>60</b>

## 1. INTRODUÇÃO

Com o avanço da internet os usuários comuns estão procurando *smartphones* com vários recursos como multimídia, câmeras, jogos, GPS, acesso internet e TV digital. O mercado corporativo também está em alta, e muitas empresas estão optando por incorporar aplicações móveis para agilizar e trazer qualidade em seus negócios, integrando as aplicações móveis com os seus sistemas *back-end*.

Desta forma aplicações que são executadas em um *smartphone* podem estar literalmente conectadas, sincronizando informações diretamente de um servidor. Para isso, o que todos precisavam era de uma plataforma poderosa e flexível para tornar tudo isso mais viável e cada vez uma realidade para nós. As empresas e os desenvolvedores buscam uma plataforma moderna e ágil para o desenvolvimento de aplicações corporativas para auxiliar em seus negócios e lucros. Já os usuários comuns buscam um celular com um visual elegante e moderno, de fácil navegação e uma infinidade de recursos.

Com o ambiente de negócios atual, os clientes estão mais exigentes e esta cada vez mais difícil de obter a aceitação dos produtos, e para o sucesso é essencial interações mais atraentes, aplicações mais leves, fazendo com que as RIAs (*Rich Internet Application*) sejam a melhor opção.

O *Android* é a resposta da Google para ocupar esse espaço, porém, a Google não está sozinha e sim um grupo formado por empresas líderes do mercado de telefonia, fabricantes de *smartphone* e companhias de software totalizando mais de 30 empresas, chamado de *Open Handset Alliance* foi criado com a intenção de padronizar a primeira plataforma de código aberto e livre para celulares, justamente para atender e atrair todas expectativas e tendências do mercado atual.

Em 2011 o framework *Adobe Flex 4.5* trouxe para o desenvolvimento RIA a possibilidade de criar aplicações para dispositivos móveis, na plataforma *Android*, celulares ou *tablets*. E sobre a arquitetura, os dispositivos móveis são mais fracos em processamento e memória, e por isso necessitam de um tratamento diferenciado em relação às aplicações *desktop*, portanto o foco na criação de uma aplicação

móvel é no desempenho. A Adobe trouxe com essa nova versão do framework *Flex* para *mobile* um bom conjunto de idéias e componentes que garantem a mesma qualidade de uma aplicação *desktop* com uma performance mais otimizada.

## 1.1. OBJETIVOS

### 1.1.1. Geral

Verificar a viabilidade em usar o framework *Adobe Flex* aplicado à plataforma *Android* através do desenvolvimento de uma aplicação prática integrada com um servidor JEE (*Java Enterprise Edition*) demonstrando a facilidade e qualidade no desenvolvimento da aplicação.

### 1.1.2. Específicos

- Descrever de forma breve como surgiu o primeiro *smartphone* e como chegaram no nível atual;
- Desenvolver um referencial teórico sobre o framework *Flex*, seus componentes e a estrutura do *Flex* para uma aplicação móvel;
- Desenvolver uma aplicação simples utilizando o *Adobe Flex* para criar aplicações para *Android* acessando um servidor *JEE*;
- Apresentar os resultados obtidos.

## 1.2. JUSTIFICATIVA

A plataforma e o mercado de aplicações móveis estão em crescente avanço interagindo com as tecnologias da informação e comunicação, por este fato a humanidade tende a utilizar os seus aparelhos móveis não mais como simples celulares e sim como *smartphones* capazes de executar tarefas que antes só eram possíveis de serem realizadas em uma plataforma *desktop*. Existem várias tecnologias e *frameworks* que são RIA (*Rich Internet Application*), mas nem todas atraem o desenvolvedor ao ponto de optar em utilizar, e entre as tecnologias que possuem o conceito RIA, se destacando e conquistando os desenvolvedores é o *Adobe Flex*.

O *Adobe Flex* é um framework RIA multi-plataforma que fornece diversos componentes que se destacam através da usabilidade e acessibilidade com o usuário, optou-se por este framework para estudo pela facilidade de realizar o desenvolvimento de aplicativos móveis interagindo com a plataforma móvel livre que mais vêm se destacando no mercado, a plataforma *Android*. Além disso, a junção destes dois componentes permite criar aplicativos que utilizem todos os recursos nativos de um *smartphone* interagindo com tecnologias *server-side*<sup>1</sup> beneficiando a distribuição da arquitetura centralizando as regras de negócio em um servidor JEE (*Java Enterprise Edition*) remoto e robusto. Devido a isso, um estudo sobre a viabilidade em utilizar o *Flex* para o desenvolvimento móvel se torna um boa referencial para desenvolvedores que sempre procuram escolher qual tecnologia utilizar.

---

<sup>1</sup> Termo utilizado para designar operações executadas no servidor.

<sup>2</sup> Modelo cliente-servidor, que depende de um servidor central para o processamento de atividades

## 2. REFERENCIAL TEÓRICO

### 2.1. RIA - RICH INTERNET APPLICATION

As RIAs são aplicações web que se comportam e tem funcionalidades de softwares tradicionais do tipo *Desktop*. Os sistemas em RIA transferem todo o processamento da interface para o navegador da internet, porém mantém a maior parte dos dados no servidor de aplicação (ADOBE, 2008).

As aplicações tradicionais na plataforma WEB centralizam todo o código em uma única arquitetura, trabalha com o modelo de um navegador agindo como um *thinclient*<sup>2</sup>, cuja a função principal era renderizar a página HTML e enviar requisições de volta para um servidor, que executava e trazia uma resposta ao cliente. A grande desvantagem é essa espera da resposta do servidor, para só depois a página do cliente ser recarregada com uma resposta. As RIAs podem diminuir significamente o número de sincronizações com o servidor e aumentar a interatividade do cliente.

Hoje, as demandas por aplicações continuam a crescer e são, por vezes, bem diferentes das demandas de meados dos anos 90. Usuários finais estão esperando mais de seus investimentos em tecnologia de Internet. A capacidade de agregar valor real aos usuários esta forçando muitas empresas a olharem na direção de modelos mais ricos para aplicações de Internet - modelos que combinem o poder rico em mídia do *desktop* tradicional com a distribuição e a natureza rica em conteúdos das aplicações web. E atualmente é fácil construir uma arquitetura sólida para uma aplicação através da separação entre as áreas de negócios, acesso a dados e apresentação. (TAPPER, 2009).

De acordo com Tapper (2009) as RIAs devem ser capazes de fazer o seguinte:

- Fornecer um *runtime*<sup>3</sup> eficiente e de alta performance para a execução de código, conteúdo e comunicações.
- Fornecer modelo de objetivos potentes e extensíveis para facilitar a interatividade. Os Web browsers progrediram nos últimos anos em sua capacidade de suportar interatividade através do DOM (*Document Object*

---

<sup>2</sup> Modelo cliente-servidor, que depende de um servidor central para o processamento de atividades

<sup>3</sup> Ambiente onde a aplicação é executada



Model) via *javascript* e DHTML (Dynamic Hipertext Markup Language), mas ainda carecem de um suporte *cross-browser*<sup>4</sup> e *cross-platform*<sup>5</sup> padronizado. Construir RIAs com estas ferramentas para que elas funcionem com uma variedade de navegadores e sistemas operacionais envolve criar varias versões da mesma aplicação.

- Possibilitar o uso de objetos *Server-side* via *Web Services*, *Remote Object* ou qualquer outra tecnologia do gênero. A proposta de RIAs inclui a capacidade de se separar claramente lógica de apresentação e interfaces de usuário, da lógica da aplicação armazenada no servidor.

Portanto as aplicações ricas para internet oferecem uma experiência rica e envolvente, que melhora a satisfação do usuário e aumenta a produtividade. Usando o amplo alcance da Internet, as RIAs podem ser implantadas em vários navegadores, desktops e dispositivos. A Plataforma *Adobe Flash* é a solução principal para a construção de aplicações RIAs, oferecendo um conjunto completo de tecnologias integradas. (Figura 1).

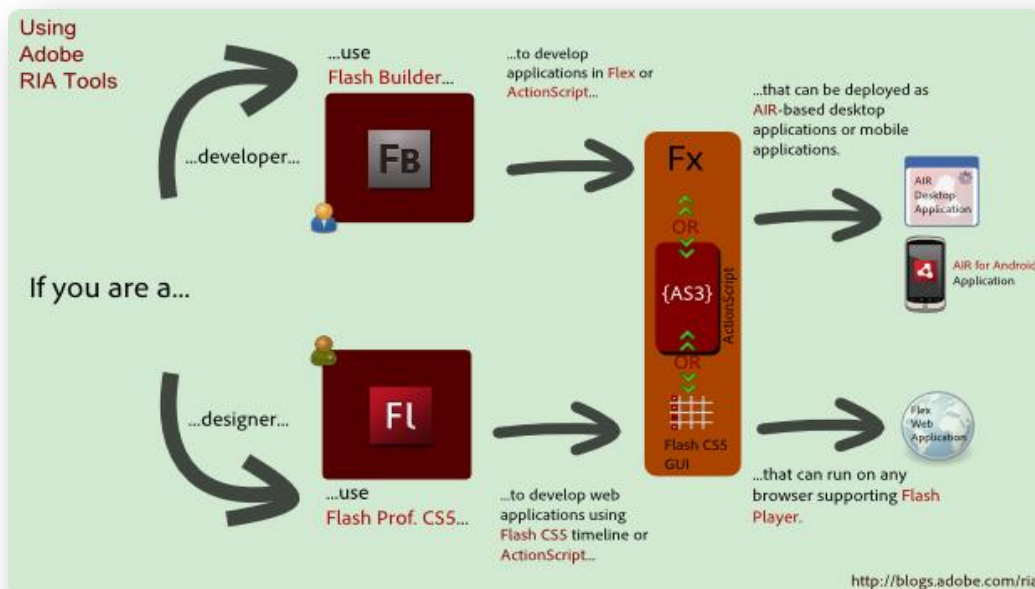


Figura 1 - Integração RIA com várias plataformas.

Fonte: ADOBE BLOG RIA (2010)

<sup>4</sup> Habilidade de uma aplicação Web suportar vários browser

<sup>5</sup> Habilidade de uma aplicação suportar várias plataformas

### 2.1.1. Benefícios

Segundo Velayutham (2010), as principais vantagens das RIAs são :

- O Cliente pode interagir com o Servidor de forma assíncrona, ou seja, determinada ação do usuário não necessita esperar uma resposta do servidor;
- Mais ágil do que aplicações web tradicionais;
- Interatividade com a interface do usuário;
- Reduzir os custos de banda de atualização de página parcial;
- Apresentação clara e atendendo a expectativa do cliente;
- Ajuda o tipo de negócio na construção de aplicações.

### 2.1.2. Deficiências e Restrições

- São executadas em um *Sandbox*<sup>6</sup>, podendo restringir acessos a recursos do sistema;
- Tempo de carregamento de aplicações são sempre carregadas do servidor para o cliente;
- Os sites de busca podem não serem capazes de indexar os textos em RIAs.

## 2.2. DISPOSITIVOS MÓVEIS

Os dispositivos computacionais móveis são aqueles que podem ser facilmente movidos fisicamente ou que se mantenham funcional mesmo em movimento; isso quer dizer que os dispositivos móveis não se englobam apenas para os celulares e *smartphones*. (DEV MEDIA, 2011).

Segundo Reza B'Far (2005), Sistemas Computacionais Móveis são sistemas computacionais que podem facilmente ser movidos fisicamente ou cujas capacidades podem ser utilizadas enquanto eles estão sendo movidos. Como estes

---

<sup>6</sup>Mecanismo de segurança que cria um espaço virtual no qual todas as alterações em arquivos sejam apagadas após reiniciar o dispositivo.

sistemas prevêm tal mobilidade, eles normalmente oferecem recursos e características que não encontramos em sistemas comuns, como por exemplo:

- Monitoramento do nível de energia e prevenção de perda de dados em caso de pane de energia;
- Armazenamento de dados local e/ou remoto, através de conexão com ou sem fio;
- Sincronização de dados com outros sistemas.

### 2.2.1. Vantagens da mobilidade

De acordo com Mendonça 2010 em um artigo publicado pela DevMedia, a vantagem mais trivial da mobilidade é, logicamente, a possibilidade de acessar dados em qualquer lugar e a qualquer momento. Mas as vantagens não param por ai. Com sistemas móveis bem planejados, é possível:

- Reduzir custos de comunicação, pois você não precisará ligar para outras pessoas para saber informações que seu dispositivo/sistema já possui;
- Reduzir custos de entrada/processamento de dados, já que, em vez de escrever em papel (que teria que ser re-digitado), você escreverá num formato digital, podendo ser transmitido para outros dispositivos ou sistemas;
- Otimizar o tempo, já que você terá um sistema ao seu lado que lê dará informações precisas de forma imediata. Além disso, seu sistema poderá enviar e receber informações remotamente, dispensando seu deslocamento para outros locais para receber tais dados;

Aumentar seu faturamento, porque com uma maior gama de informações disponíveis nos momentos de negociação, você será mais eficiente e terá melhores resultados.

### 2.2.2. História *Smartphone*

Em 1993 o "Simon," A IBM foi a primeira tentativa real da indústria de tecnologia para criar um celular "com algo mais" pela incorporação de voz e dados,

que funcionava como um telefone celular, um assistente (calendário digital pessoal, lista de endereços, relógio mundial, calculadora, bloco de notas, e-mail e jogos), e até mesmo uma máquina de fax. O surpreendente, considerando, que foi lançado em 1993, tinha uma tela de toque para discar números e texto usando um teclado QWERTY. Os aspectos negativos foram o design, tamanho e peso. Já em 1996 o Palm Pilot não era tecnicamente um smartphone, mas foi muito importante porque ajudou a popularizar o uso de dispositivos portáteis, e os usuários se acostumarem com a idéia da capacidade de levar os seus dados de um lugar para outro. Foi muito utilizada por executivos e empresários. O Pilot 1000 tinha um processador de 16MHz e uma memória de 128KB. Em 1998 o Nokia 9110 Communicator era um dispositivo com um design mais semelhante ao que entendemos hoje como smartphone. Sua tela não era de cor, e não poderia navegar na Internet, mas tinha um teclado QWERTY deslizante que serviu de modelo para o telefone atual, como o Motorola Droid.

No final dos anos 90 a empresa canadense *Research In Motion* (RIM) era conhecido por seus *paggers*, que foram utilizados por dezenas de milhões de pessoas em todo o mundo. Mas no início de 2002, a RIM entrou no mercado de telefonia móvel, e desenvolveu em grande estilo: o *BlackBerry* 5810 foi um telefone com a capacidade de verificar e-mails e navegar na Internet. O *Treo* 600 foi o primeiro *smartphone* lançado pela *Palm, Handspring* após a aquisição do fabricante. Este telefone tinha a característica de GSM e suporte CDMA, tinha 32MB de RAM e um processador de 144 MHz que vendeu muito bem, mas foi lançado em 2003, época em que a *Palm* começou a sua queda na popularidade.

O sucesso da Apple com a sua primeira tentativa com o Iphone para entrar no mercado de telefonia móvel em 2007, foi incrível. O produto vendeu milhões de unidades, em parte graças a tela tátil e ofereceu a melhor experiência de Internet até agora. No mesmo ano que a Apple lançou o *iPhone*, o Google lançou seu sistema operacional Android. Esta última versão não foi tão explosiva, ou causou tanto rebuliço na época, mas hoje, quatro anos depois, podemos dizer categoricamente que o Android é bem sucedido porque é um *bestseller* nos os EUA e Europa, tem milhares de aplicações disponíveis no Android Market, e um futuro brilhante. De fato, recentemente tornou-se fato conhecido que nos Estados Unidos há mais telefones com *Android* que *iPhones*.

### 2.2.3. Sistemas operacionais

Sistema Operacional é um software que gerencia recursos e provê aos programadores e usuários uma interface para acessar estes recursos. Ele processa dados e responde pela atribuição e gestão de tarefas e recursos internos do sistema. Ele trabalha sob um processador que dá velocidade à execução destas tarefas. Nos aparelhos móveis existe a limitação de espaço físico para que os processadores e memória sejam mais robustos, no entanto, eles têm recebido melhorias extraordinárias com o tempo (UCEL, 2009).

Atualmente os sistemas operacionais para smartphones existentes mais conhecidos e populares são: *Iphone*, *Windows Mobile*, *Android*, *BlackBerry*, *PalmPre* e *Symbian*.

### 2.2.4. Crescimento dos principais *smartphones*

O crescimento na utilização de smartphones no mundo deve uma grande parte de seu sucesso à plataforma Android, que tem conseguido êxito e gerado boas impressões e críticas nos últimos meses. Apenas nos primeiros três meses de 2011, 427,8 milhões de aparelhos foram vendidos ao redor do mundo, houve um aumento de 19% comparado ao mesmo período do ano passado, segundo um estudo da empresa de pesquisas *Gartner Inc*.

De acordo com a *Gartner Inc* (2011) a plataforma Android passou de quarta colocada a líder de mercado de 2010 até hoje e carrega nas costas 36,3 milhões dos 100,8 milhões de smartphones vendidos no primeiro trimestre de 2010. Ou seja, 36% do total vendido a usuários finais. Logo depois aparecem as empresas renomeadas Symbian, IOS com 27,4% e 16,8% respectivamente.

O crescimento foi visto e comentado pelas principais sites de notícias do país como:

- O GLOBO: “Android estará em quase metade dos smartphones em 2012, diz Gartner.” No site [www.oglobo.globo.com](http://www.oglobo.globo.com);

- Terra: “Android terá 49% do mercado em 2012, diz estudo” no site [www.terra.com.br](http://www.terra.com.br);
- Info: “Android lidera; Nokia e MS perdem mercado” no site [www.info.abril.com.br](http://www.info.abril.com.br).

Segundo a empresa comScore (2011), que é uma empresa líder mundial na medição do mundo digital incluindo a medição de audiência móvel, apurou que o Android se tornou a plataforma móvel mais popular entre os smartphones, ganhando quase oito pontos percentuais no mercado entre outubro de 2010 e janeiro de 2011. (Quadro 1). O sistema operacional da Google evoluiu, de dezembro de 2010 até março de 2011, para 34,7% do mercado (Quadro 2). Analisando os dois quadros é possível ver uma evolução significativa do Android, evoluindo mais de 2% de janeiro a março, enquanto as concorrentes caíram ou obtiveram um aumento bem inferior.

Top Smartphone Platforms 3 Month Avg. Ending Jan. 2011 vs. 3 Month Avg. Ending Oct. 2010 Total U.S. Smartphone Subscribers Ages 13+ Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Oct-10	Jan-11	Point Change
Total Smartphone Subscribers	100.0%	100.0%	N/A
Google	23.5%	31.2%	7.7
RIM	35.8%	30.4%	-5.4
Apple	24.6%	24.7%	0.1
Microsoft	9.7%	8.0%	-1.7
Palm	3.9%	3.2%	-0.7

Quadro 1 - Aumento dos Smartphones entre outubro de 2010 e janeiro de 2011

Fonte : comScore (2011)

Top Smartphone Platforms 3 Month Avg. Ending Mar. 2011 vs. 3 Month Avg. Ending Dec. 2010 Total U.S. Smartphone Subscribers Ages 13+ Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Dec-10	Mar-11	Point Change
Total Smartphone Subscribers	100.0%	100.0%	N/A
Google	28.7%	34.7%	6.0
RIM	31.6%	27.1%	-4.5
Apple	25.0%	25.5%	0.5
Microsoft	8.4%	7.5%	-0.9
Palm	3.7%	2.8%	-0.9

Quadro 2 - Aumento dos smartphones no primeiro trimestre de 2011

Fonte: comScore (2011)

### 2.3. A PLATAFORMA ANDROID

O *Android* é uma plataforma para *smartphone*, baseado em sistema operacional em Linux e uma plataforma Java, possui diversos componentes, com uma variada disponibilidade de bibliotecas e interface gráfica, além de disponibilizar a ferramenta para criação de aplicativos (Lecheta, 2009).

A OHA (*Open Handset Alliance*) é um grupo formado por gigantes do mercado de telefonia de celulares liderados pela Google com mais de 30 empresas que está envolvida em construir a plataforma, inovar e acelerar o desenvolvimento de aplicações e serviços, trazendo aos consumidores uma experiência mais rica em termos de recursos, menos dispendiosa em termos financeiros para o mercado móvel. A aliança OHA é composta por um grupo bastante heterogêneo de empresas, que compreende em operadoras de celular a fabricantes de *handsets*, empresas semi-condutoras e companhias de *software*, entre as empresas mais conhecidas estão na Figura 2.

Segundo Santos (2009), o *Android* trata-se de uma pilha de componentes de *software*, desenvolvida para dispositivos móveis, que inclui sistema operacional, bibliotecas, *frameworks* de *middleware* e aplicações chave.

Os desenvolvedores podem criar aplicações utilizando o *Android* SDK. As aplicações são escritas utilizando a linguagem Java sobre a máquina virtual *Dalvik* que é customizada para dispositivos com restrições de recursos.

O *Android* tem como objetivos principais (Project, 2008; Lecheta, 2009):

- A possibilidade de personalização das aplicações e componentes presentes em seu sistema, por ser de código aberto e gratuito;
- A possibilidade de desenvolvimento rápido e moderno de aplicações corporativas, uma vez que sua plataforma é moderna e flexível.



Figura 2 - Empresas que fazem parte da *OpenHandset Alliance*

Fonte : (SBROH WordPress, 2011)

Composto por um conjunto de software para dispositivos móveis, um *middleware*<sup>7</sup>, aplicativos e um sistema operacional, uma das grandes apostas dessa plataforma é a disponibilidade do *Android SDK (Software Developed Kit)*. 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*.

A plataforma *Android* possui várias características que a colocaram entre as mais utilizadas, de acordo com Elder Elisandro Schemberger:

- *Framework* de desenvolvimento de aplicações: reutilização de código e facilidade de acesso a recursos exclusivos e manutenção;
- Nova máquina virtual (*Dalvik*): criada e otimizada para dispositivos móveis;
- Navegador web integrado: Baseado no projeto *open sourcewebkit*.

<sup>7</sup>Faz a mediação entre software e as demais aplicações



- Biblioteca de gráficos otimizada para dispositivos móveis, baseada na especificação OpenGL ES 1.0.
- *SQLite*: armazenamento de dados estruturados.
- Suporte multimídia: compatibilidade com os principais formatos existentes.
- Telefonia com tecnologia GSM: As aplicações podem manipular operações telefônicas, caso o fabricante permita esse acesso.
- Bluetooth, EDGE, 3G e WiFi: foco nas principais tecnologias de transmissão de dados sem fio, também depende da permissão do fabricante para o acesso.
- Câmera e GPS: Ter o celular como uma ferramenta para interação com redes sociais, também dependente da permissão do fabricante para acesso.

Ambiente de desenvolvimento com *plugin* para a IDE Eclipse: Inclui emulador, ferramentas para *debug* e supervisão de memória e desempenho.

### 2.3.1. Versões

Desde a primeira versão 1.5 do *Android*, são nomeadas com nomes de sobremesas ou bolos e também seguem uma ordem alfabética:

- 1.5: *Cupcake* (Abril de 2009,);
- 1.6: *Donut* (Setembro de 2009);
- 2.1: *Eclair* (Janeiro de 2010);
- 2.2: *FroYo* (*FrozenYogourt* - Maio de 2010);
- 2.3: *Gingerbread* (Dezembro de 2010);
- 3.0: *Honeycomb* (Janeiro de 2011);
- 3.1 *Ice Cream Sandwich* (Maio de 2011).

A Figura 3 mostra um gráfico com as versões mais utilizadas pelos clientes atualmente.

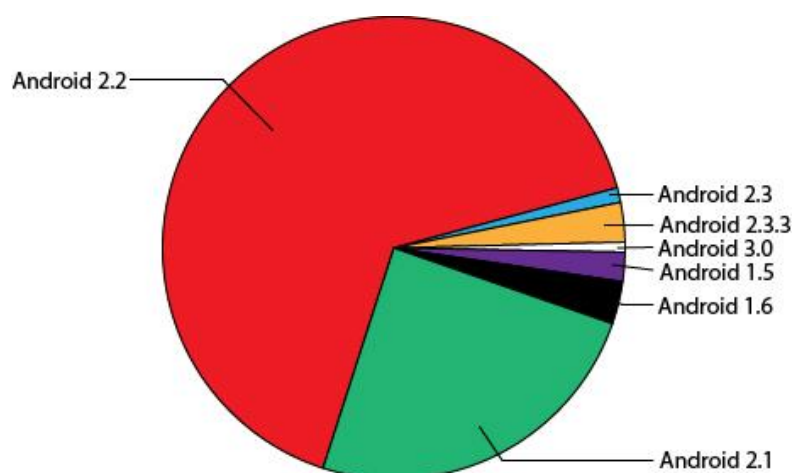


Figura 3 - Gráfico relativo as versões mais utilizadas pelos clientes  
Fonte : *ANDROID VERSIONS* (2011)

### 2.3.2. *Android SDK*

O *Android SDK* é conjunto de emulador, ferramentas e bibliotecas de desenvolvimento para a Plataforma *Android*, onde é possível ter acesso aos recursos do sistema e componentes de hardware de um dispositivo, como por exemplo: Funções para SMS, Web, Mapas, Contatos e Câmera. Essa interação com o SDK ocorre através da linguagem de programação Java.

O Google disponibilizou o Kit de Desenvolvimento do *Android* em Novembro de 2007 ainda como versão de teste, e a versão *Android SDK 1.0 Release 1* em Setembro de 2008, esse kit de desenvolvimento possui ferramentas que são usadas para a criação dos aplicativos. Em 2010 lançaram as versões 2.1, 2.2 e 2.3. O SDK do *Android* está disponível para Desenvolvimento de Aplicativos com Microsoft Windows, Linux e Mac OS (*ANDROID*, 2011).

### 2.3.3. *Android Market*

O *Android Market* permite aos desenvolvedores publicarem os aplicativos, para os usuários que possuem smartphones com *Android*, de uma forma muito

simples. Uma vez registrado, o desenvolvedor tem o controle completo sobre quando disponibilizar as aplicações para os usuários (*ANDROIDMARKET*, 2011).

Os desenvolvedores podem também gerenciar o seu portfólio, ver informações de downloads, as classificações de aplicações mais baixadas, sendo pagas ou não e também publicar atualizações das novas versões de seus aplicativos. Para tudo isso é de acordo com os direitos do *Android Market* é necessário algumas regras para a publicação da aplicação, sendo umas obrigatórias e outras opcionais.

Na fase de *Upload* são obrigatórios:

- Obrigatório que o tamanho do APK, ou arquivo padrão do *Android*, tenha um tamanho máximo de 50MB.
- Obrigatório que tenha 2 *screenshots* da aplicação, podendo ter até 6 como opção.
- Obrigatório que o ícone da aplicação esteja em alta resolução.

Na fase de detalhes para a publicação:

- Obrigatório que o idioma da aplicação siga o padrão que é Inglês dos EUA.
- Obrigatório também que o título da aplicação que aparecerá no *Android Market*, podendo também adicionar um para cada linguagem.
- Obrigatório que possua uma descrição comum limite de 4000 caracteres.

Obrigatório que escolha o tipo de aplicação, jogo ou aplicação.

## 2.4. ADOBE FLEX

O *Adobe Flex* é uma estrutura altamente produtiva, gratuita e de código aberto para a criação de aplicações expressivos para plataformas Móveis, Web e para Desktop. Permite a criação de aplicativos móveis e aplicações Web que compartilham uma base de código comum, reduzindo o tempo e custo da criação de aplicativos e a manutenção de longo prazo.

Segundo TAPPER (2009), sentiu-se uma necessidade de uma ferramenta para a criação de RIAs que fossem mais amigáveis, e com essa necessidade a *Macromedia* atual *Adobe* desenvolveu uma linguagem e um compilador que

propiciaram aos desenvolvedores trabalhar com uma linguagem habitual, que poderia ser rodado no *Flash Player*. A primeira versão do *Flex* surgiu em 2004, e foi bem continuada até hoje. Em relação à arquitetura, aplicações desenvolvidas em *Flex* são parecidas com aplicações em AJAX, porque ambas são capazes de atualizações dinâmicas à interface do usuário e ambas incluem a habilidade de enviar e carregar dados do background.

O *Flex* pode ser usado para criar aplicativos que são executados no navegador através do *Adobe Flash Player*, os aplicativos móveis que são executados em plataformas móveis são executados através do *Adobe AIR* e aplicativos para desktop que são executados fora do navegador e podem ser usados mesmo desconectados da Internet também através do *Adobe AIR*. O *Flash Player* e o *Adobe AIR* são execuções de clientes de classe empresarial com gráficos vetoriais avançados, de alto desempenho capazes de lidar com os aplicativos mais exigentes, que processam grandes volumes de dados.

Oferece um modelo moderno de linguagem e programação baseado em padrões que oferece suporte a modelos comuns de design. MXML, uma linguagem declarativa baseada em XML, é usada para descrever comportamentos e layout de interface de usuário, e *ActionScript 3.0*, uma linguagem de programação orientada à objetos, é usada para criar a lógica de cliente. O *Flex* inclui também uma biblioteca de componentes com mais de 100 componentes de interface de usuário comprovados e extensíveis para a criação de aplicações RIAs, além de um depurador interativo de para aplicação *Flex*.

#### 2.4.1. Versões

A *Macromedia* inicialmente desenvolveu as versões do *Flex* 1.0 e 1.5. Em 2005 a *Adobe* comprou a *Macromedia* e deu continuidade neste Framework. A *Adobe* modificou significativamente a linha do produto com a versão 2. O centro do *Flex 2 SDK* consistia em um compilador de linhas de comando e uma completa biblioteca de classes de componentes e utilitários, disponibilizada gratuitamente para download. Aplicações completas desenvolvidas em *Flex* podiam ser desenvolvidas totalmente utilizando unicamente o SDK, que não possuía limitações ou restrições comparadas com a mesmo SDK incluído com o *Flash Builder IDE*.

O Flex possui várias versões, até 2005, o Flex ainda pertencia a Macromedia, a partir desta data a Adobe comprou e deu continuidade neste Framework :

Versão	Data
<b>Flex 1.0</b>	Março de 2004
<b>Flex 1.5</b>	Outubro de 2004
<b>Flex 2.0 (Alpha)</b>	Outubro de 2005
<b>Flex 2.0 Beta 1</b>	Fevereiro de 2006
<b>Flex 2.0 Beta 2</b>	Março de 2006
<b>Flex 2.0 Beta 3</b>	Mai de 2006
<b>Flex 2.0 Final</b>	28 de Junho de 2006
<b>Flex 2.0.1</b>	5 de Janeiro de 2007
<b>Flex 3.0 Beta 1</b>	11 de Junho de 2007
<b>Flex 3.0 Beta 2</b>	1 de Outubro de 2007
<b>Flex 3.0 Beta 3</b>	12 de Dezembro de 2007
<b>Flex 3.0</b>	25 de Fevereiro de 2008
<b>Flex 3.1</b>	15 de Agosto de 2008
<b>Flex 3.2</b>	17 de Novembro de 2008
<b>Flex 3.3</b>	4 de Março de 2009
<b>Flex 3.4</b>	18 de Agosto de 2009
<b>Flex 3.5</b>	18 de Dezembro de 2009
<b>Flex 4.0</b>	22 de Março de 2010
<b>Flex 4.1</b>	1 de Julho de 2010
<b>Flex 4.5</b>	Setembro de 2010

Quadro 3 - Versões do Adobe Flex

#### 2.4.2. MXML

MXML é uma linguagem de marcação de texto baseada em XML utilizada para criar os componentes de interface do usuário. Esta linguagem os desenvolvedores utilizam para definir o layout da interface do usuário, a aparência e os comportamentos de um aplicativo Flex, também pode ser usada para definir aspectos não-visuais de uma aplicação, tais como o acesso a fontes de dados sobre

as ligações de servidor e de dados entre componentes de interface e fontes de dados no servidor (TAPPER, 2009).

Assim como o HTML, o MXML fornece *tags*<sup>8</sup> que definem interfaces de usuário (Figura 4). No entanto, esta linguagem é mais estruturada do que HTML, e fornece um conjunto de marcas muito mais extensa. Por exemplo, MXML inclui marcas de componentes visuais, e também componentes não visuais que fornecem conexões a outros serviços utilizando *WebService*, *Remote Object*, e efeitos de animação.

Além disso, uma das maiores diferenças entre HTML e MXML é que as aplicações MXML definidas são compiladas em arquivos SWF (*Shockwave Flash*) e executados pelo *Adobe Flash Player*, para aplicações *Web* ou aplicações AIR (*Adobe Integrated Runtime*) para *Desktop*. MXML também suporta componentes personalizados escritos em arquivos MXML e *ActionScript* (ADOBE, 2011).

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
3     xmlns:s="library://ns.adobe.com/flex/spark" title="HomeView">
4     <fx:Declarations>
5         <!-- Place non-visual elements (e.g., services, value objects) here -->
6     </fx:Declarations>
7     <s:TextInput id="inputText"/>
8     <s:Button label="Clique aqui"/>
9 </s:View>
10

```

Quadro 4 - Linguagem MXML

#### 2.4.3. *ActionScript* 3.0

O *ActionScript* 3.0, uma linguagem orientada à objetos com base em ECMAScript, é a linguagem usada para criar a lógica da aplicação no lado do cliente, sendo utilizada principalmente para construir aplicações RIAs. É uma linguagem, originalmente, criada para desenvolver conteúdo interativo desde animações simples a interfaces interativas, com riqueza de dados e aplicações em *browsers*, *desktops* e dispositivos móveis (HERRINGTON, 2008).

De acordo com a Adobe (2009), as versões do *ActionScript* ofereceram a flexibilidade necessária para criar verdadeiramente experiências online. Com o *ActionScript* 3.0 essa experiência avança ainda mais, proporcionando excelente desempenho e facilidade de desenvolvimento para criar aplicações altamente complexas, grandes conjuntos de dados e bases de código reutilizáveis.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
3     xmlns:s="library://ns.adobe.com/flex/spark" title="HomeView">
4     <fx:Declarations>
5         <!-- Place non-visual elements (e.g., services, value objects) here -->
6     </fx:Declarations>
7     <fx:Script>
8         <![CDATA[
9
10             protected function buttonClickHandler(event:MouseEvent):void
11             {
12                 inputText.text = "Olá Professor";
13             }
14
15         ]]>
16     </fx:Script>
17     <s:TextInput id="inputText"/>
18     <s:Button label="Clique aqui" click="buttonClickHandler(event)"/>
19 </s:View>
20
21

```

Quadro 5 - Linguagem MXML e ActionScript

#### 2.4.4. Adobe Flash Builder

O *Adobe Flash Builder* é uma ferramenta de desenvolvimento de nível profissional, baseada no Eclipse, para desenvolver aplicações na plataforma Móvel, Web e Desktop. Inclui suporte para codificação inteligente, depuração em etapas interativa, criação de perfis de aplicativo e design visual para criar layout da interface do usuário. (Figura 4)

---

<sup>8</sup> Estruturas de linguagem de marcação que consistem em breves instruções

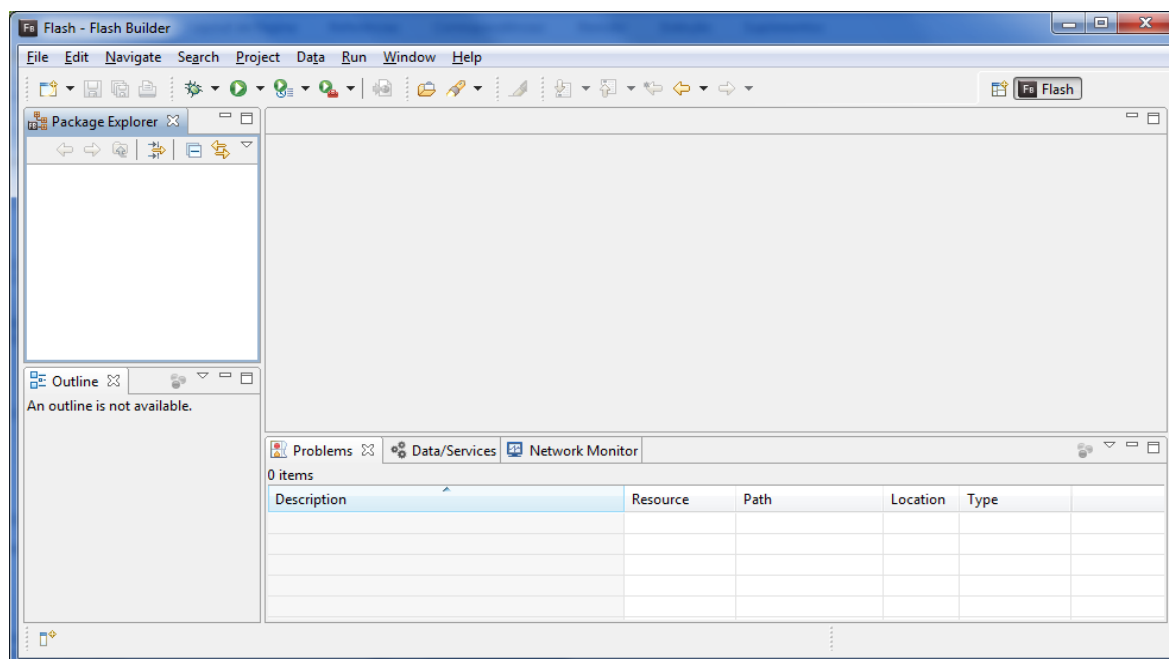


Figura 4 - Adobe Flash Builder

O *Flash Builder* está disponível nas edições Standard e Premium. O *Flash Builder Premium* vem com novas ferramentas poderosas de teste, incluindo visualizador de desempenho de memória, um monitor de rede e suporte integrado para teste, que acelera o desenvolvimento, resultando em aplicações com desempenho maior. Já a versão Standard é de graça para alunos, professores e desempregados, facilitando o acesso a ferramenta e a oportunidade para desenvolver (ADOBE, 2011).

#### 2.4.5. Adobe Flex para dispositivos móveis

A Plataforma Adobe *Flash* já permite aos desenvolvedores criar aplicações em vários navegadores e sistemas operacionais. Com o lançamento do *Flex 4.5 SDK* e *Adobe Flash Builder 4.5*, juntamente com a disponibilidade da execução do Adobe AIR em dispositivos móveis, os desenvolvedores podem criar aplicações móveis para *Smartphones* e *Tablets* com a mesma facilidade e qualidade das plataformas desktop.



O Flex 4.5 fornece um caminho comum na criação de aplicações *Web*, *Desktop* ou *Mobile*, o *Flex* e o *Flash Builder* podem reduzir significativamente o tempo e o custo associados ao desenvolvimento e testes das aplicações, proporcionando aos usuários uma experiência consistente.

O *Flex* 4.5 expande a extensa biblioteca existente de componentes da plataforma *Web* e da *Desktop* adicionando 21 novos componentes móveis otimizados, *touchscreen* e com reconhecimento de densidade, acelerômetro, acelerando o desenvolvimento de aplicativos para dispositivos móveis. Além dos novos componentes móveis, o Flex 4.5 adiciona outros recursos e melhorias importantes para a criação de aplicativos para dispositivos móveis. Esses incluem: cinética e efeitos para rolagem de componentes; otimizações de desempenho para rolagem e transições; dimensionamento automático baseado na densidade de pixels do dispositivo; temas padrão para dispositivos móveis com esquema de cores claras-sobre; suporte nativo do teclado, suporte à tela de abertura; e suporte a bitmaps de várias resoluções.

O *Flash Builder* 4.5 adiciona novos e importantes fluxos de trabalho de desenvolvimento de conteúdo móvel para ajudar a codificar, depurar e aperfeiçoar os aplicativos para dispositivos móveis. Possui um novo tipo de projeto móvel, nova visualização de projeto por dispositivo, novo suporte à criação de várias densidades, edição de permissões de aplicativo por plataforma (Figura 5), e um novo fluxo de trabalho de depuração poderoso que lhe permite depurar em um dispositivo físico ou no desktop usando um emulador integrado.

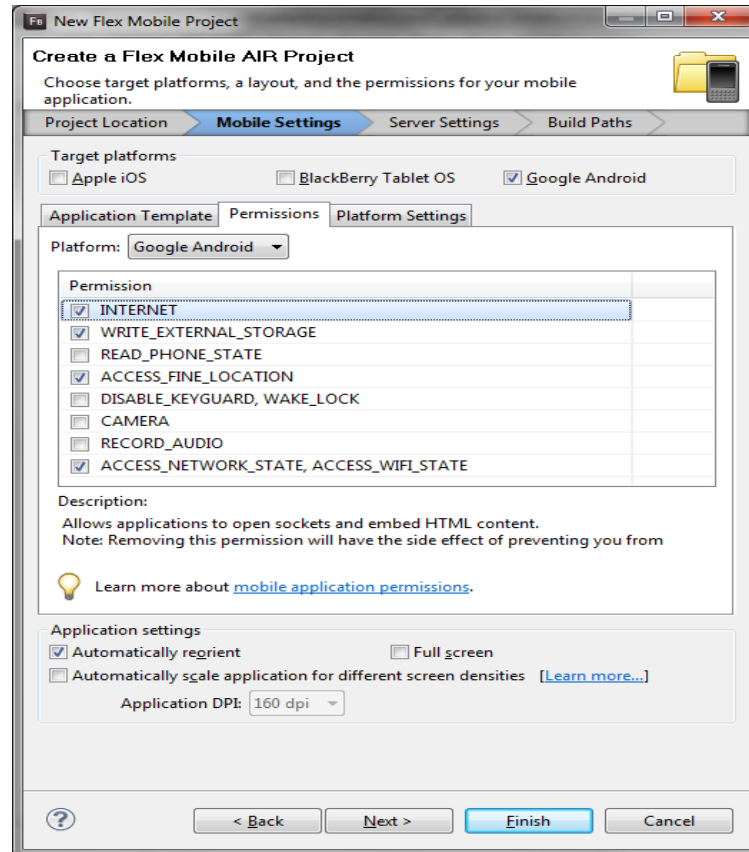


Figura 5 - Tela de criação de um projeto móvel

### 3. ARQUITETURA

#### 3.1. ANDROID

Assim como computadores ou outros dispositivos, o dispositivo móvel possui uma arquitetura distinta. Para que um sistema que um sistema consiga ser operado em diferentes tipos de hardware é necessário um sistema operacional.

Segundo Tanenbaum (2003), um sistema operacional tem que ser capaz de gerenciar o processador, memória e uma interface mais simplificada com o hardware.

De acordo com Silva (2009), a tendência na criação de sistemas operacionais nos dispositivos moveis é crescente, justamente pela necessidade do gerenciamento dos recursos que os *smartphones* disponibilizaram, como processamento e memória. É necessário que a criação de aplicativos não seja

apenas para um único modelo, mas seja para um determinado sistema operacional responsável por gerenciar as particularidades de cada dispositivo.

A arquitetura da plataforma Android é dividida em várias camadas: Applications, Application Framework, Libraries e Android Runtime; e Linux Kernel.

Segundo Rabello, (2009), na camada *Applications*, está localizada uma lista de aplicações padrões que incluem um cliente de e-mail, programa de SMS, calendário, mapas, navegador, gerenciador de contatos, e outros que serão desenvolvidos pela comunidade, sendo todas essas aplicações escritas na linguagem Java.

Já na camada Application Framework estão os componentes que permitirão com que novas estruturas sejam utilizadas para futuras aplicações, enfatizando a reutilização de código. De acordo com Ramon Ribeiro Rabello os seguintes componentes fazem parte desta camada:

- Um rico e extensível conjunto de componentes gráficos que pode ser utilizado para construir uma aplicação, bem como listas, grids, caixas de textos, botões, e até um navegador web embutido.
- Provedores de conteúdo que habilitam às aplicações acessar dados de outras aplicações (como os Contatos, por exemplo) ou compartilhar seus próprios dados
- Gerenciador de recursos que prove acesso a recursos não-codificados como gráficos, e arquivos de layout.
- Um gerenciador de notificação que permite que todas as aplicações exibam mensagens de alerta personalizáveis na barra de status.
- Um gerenciador de atividade que gerencia o ciclo de vida das aplicações e permite controlar os recursos previamente alocados, sendo que caso eles não estejam sendo mais utilizados, os mesmos são deslocados para liberar memória.

A camada abaixo da Application Framework é subdivida no grupo das bibliotecas (*libraries*) e o ambiente de execução (*runtime*) da plataforma *Android*, composto pelas bibliotecas padrão e pela máquina virtual denominada *Dalvik*. No

primeiro grupo estão as bibliotecas escritas em C/C++, que são compostas por uma coleção de bibliotecas que são utilizadas pela plataforma *Android*.

A base da arquitetura utiliza a versão 2.6 do *kernel* do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos. O *kernel* também atua como uma camada de abstração entre o hardware e o resto do software.

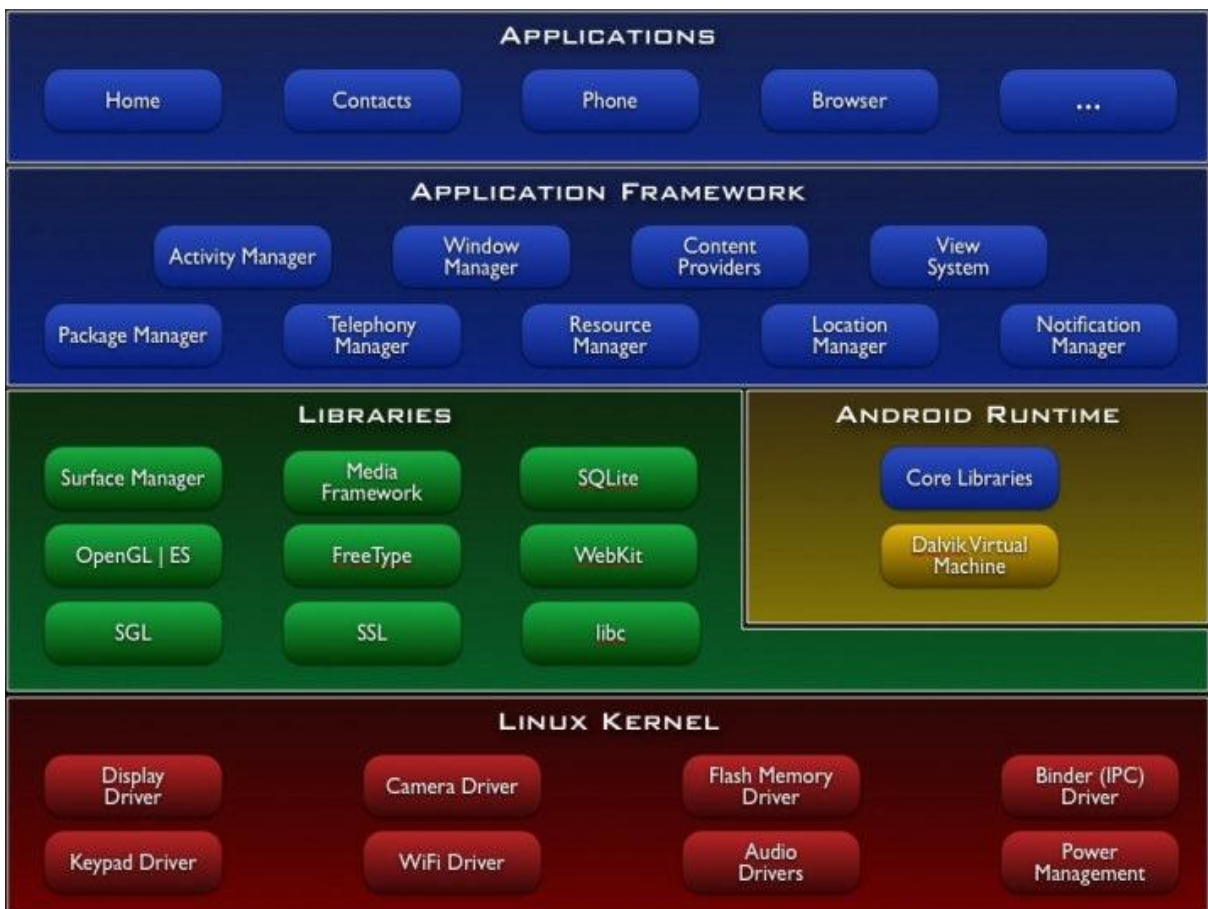


Figura 6 - Arquitetura *Android*  
 Fonte: (*Android Developers*,2011)

O *Android* possui um conjunto de bibliotecas na camada de libraries, disponíveis para o desenvolvimento dos aplicativos como a *System C Library*, *Media Libraries*, *Surface Manager*, *LibWebCore*, *SGL*, *3D Libraries*, *FreeType* e *SQLite*.

Estas bibliotecas permitem o acesso a manipulação de vídeos, áudio, banco de dados e outros recursos.

- **System C Library** : Suporta padrões ISO C e POSIX, com apoio as variantes do Unix como BSD (*Berkeley Software Distribution*) e System V, no *Android* a biblioteca é uma BSD do padrão C (Foundation, 2007).
- **Media Libraries** : Baseada em *OpenCORE*, garantindo a gravação e reprodução dos formatos mais populares de áudio e vídeo e imagens. Os formatos inclusos são MPEG4, H.264, MP3, AAC, AMR, JPG e PNG. Esta biblioteca é desenvolvida pela *PacketVideo's* (PacketVideo, 2007).
- **Surface Manager** : Gerencia o acesso ao subsistema de *display* do dispositivo, é capaz de compor gráficos em 2D e 3D a partir de aplicações de camadas múltiplas.
- **LibWebCore** : Utilizada para navegação Web que possui o código aberto, também utilizado nos navegadores *Safari* utilizado nos sistemas operacionais MAC OS (Apple Computer, 2006).
- **Bibliotecas 3D** : Baseada em *OpenGL ES 1.0* as bibliotecas usam podendo optar por utilizar a aceleração pelo hardware (Project, 2009).
- **FreeType** : Gerenciador de fontes, de código aberto e otimizado para diversos tipos de plataformas (Turner et al., 2006).
- **SQLite** : Sistema gerenciador de banco de dados embutido no *Android*, de código aberto.

O desenvolvedor além de utilizar as bibliotecas tem o acesso nativo para modificar os componentes dos *Android* que é um diferencial em relação a outras plataformas para dispositivos móveis. Tais componentes permitem que as aplicações criadas possam interagir com todas funções do celular, alterando qualquer componente para que se adequem as suas necessidades ou do aplicativo em desenvolvimento.

Os componentes permitem a interoperabilidade entres os vários subsistemas do celular, por fornecer uma plataforma de desenvolvimento aberta, o *Android* oferece aos desenvolvedores a habilidade de construir aplicações extremamente ricas e inovadoras. Os desenvolvedores são livres para aproveitarem

o hardware do dispositivo, informações do local de acesso, rodar serviços em background, definir alarmes, adicionar notificações na barra de status e muitos outros recursos.

Toda essa documentação se encontra no endereço do *Android Open Project* (<http://www.source.android.com>), podendo ter acesso aos códigos da plataforma *Android*.

### 3.2. ADOBE FLEX

A arquitetura do Adobe Flex contém uma plataforma *Adobe Flash* que possui duas runtimes primárias. O *Flash Player* é baseado no browser e o Adobe AIR tem como base o Desktop. Pelo fato do *Adobe AIR* ser contruído sobre o *Flash Player*, as APIs do *Flash Player* estão disponíveis no *Adobe Air*. O *Flex* foi contruído sobre ambos, assim ele pode ser executado em ambas runtimes, *Flash Player* e AIR. Pode ser executado nas três principais plataformas, Windows, Linux e Mac, acessando serviços remotos de várias tecnologias utilizando diferentes meios de comunicação (Figura 7).

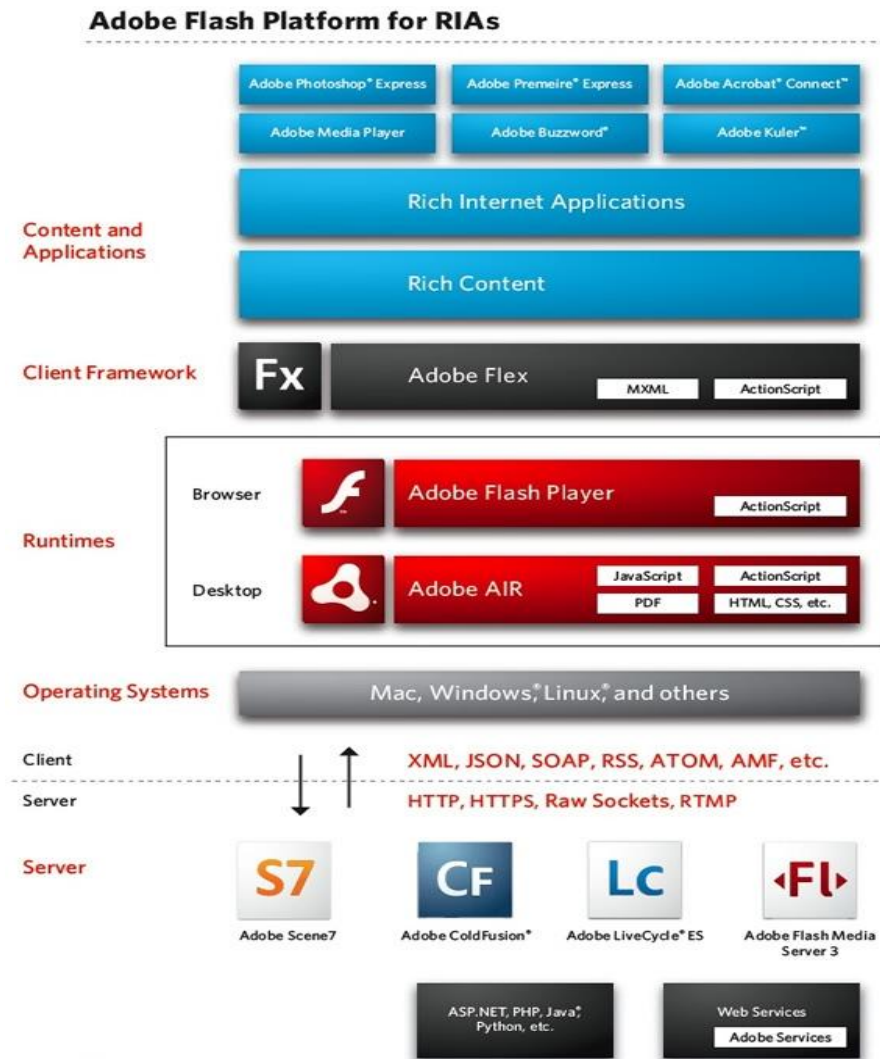


Figura 7 - Arquitetura do *Adobe Flex*

Fonte : RIA DEVELOPMENT (2008)

## 4. MATERIAIS E MÉTODOS

O ambiente experimental utilizado se baseia primordialmente em ferramentas liberadas *open-source* e que sanam as necessidades do trabalho.

### 4.1. TECNOLOGIAS ENVOLVIDAS

#### 4.1.1. Java

A linguagem de programação Java será utilizada por suportar o desenvolvimento de um servidor pra ser utilizado como *back-end* da aplicação por prover uma forma simples e flexível e robusta para o desenvolvimento de sistemas.

Uma das principais características do Java é o suporte ao paradigma de programação da orientação a objetos, sendo este imprescindível para o desenvolvimento deste trabalho.

#### 4.1.2. MySQL

MySQL é um Sistema Gerenciador de Banco de Dados (SGBD), desenvolvido primeiramente pela empresa MySQL AB, sendo que esta foi adquirida pela *Sun Microsystems* e a Sun adquirida pela *Oracle Corporation*. O MySQL suporta a *Structured Query Language* (SQL) e se comunica com a linguagem Java através de *driver* desenvolvido pela própria *Sun Microsystems*.

A principal características do MySQL é a simplicidade na configuração e utilização, o fato de se comunicar com a linguagem Java e de ser suportado pelo *framework Hibernate* foram pontos determinantes da escolha deste gerenciador para ser utilizado nesse trabalho.

#### 4.1.3. MXML e ActionScript

*ActionScript* é a principal linguagem para o Flash Player, sendo assim possível criar toda aplicação utilizando apenas as classes em ActionScript, mas há



também a possibilidade de utilizar MXML, deixando a construção da aplicação mais fácil.

## 4.2. ESTRUTURA LÓGICA

A estrutura lógica consiste em mostrar o sistema operacional utilizado, bem como os aplicativos necessários para a realização da aplicação.

### 4.2.1. Aplicativos

Para a criação de RIAs, foi utilizado o *framework* Flex, e como IDE foi utilizada os o *Flash Builder* e para a criação do projeto em Java, o Eclipse, aplicativos apresentados nos tópicos que se seguem.

#### 4.2.1.1. *Adobe Flash Builder*

É uma ferramenta baseada no Eclipse para construir expressivas aplicações móveis, *Web* e *Desktop* utilizando *ActionScript* e o *framework opensource* Flex. Para realizar a implementação deste trabalho foi utilizada a versão 4.5.

#### 4.2.1.2. *Eclipse*

Eclipse é uma IDE (*Integrated Development Environment*) liberada como software livre através da EPL (*Eclipse Public License*) criada por uma subsidiária da IBM, que pretendia diminuir a incompatibilidade entre os ambientes de desenvolvimento utilizados na empresa. Em 2001 foi criado um consórcio de empresas chamado Eclipse Consortium com a intenção de realizar o desenvolvimento da ferramenta em código aberto. Em 2004 é lançada a primeira versão livre e fundada a *Eclipse Foundation*, quem gerencia o projeto até os dias de hoje.

A versão utilizada foi a Helios 3.6.2.

#### 4.2.2. Servidor

O *Tomcat* é um servidor de aplicações *Web* escrito em Java, distribuído como *software* livre e desenvolvido como código aberto, sendo um subprojeto da *Jakarta Apache Foundation*. Foi utilizado o *Tomcat 7.0*.

#### 4.3. ESTRUTURA FÍSICA

Foi utilizado uma máquina para a implementação e teste do sistema sendo esta um notebook Sony Vaio contendo um processador Intel Core i3 e 4Gb de memória RAM.

Também foi utilizado um Samsung *Galaxy Tab* para testes e execução da aplicação desenvolvida.

## 5. RESULTADOS E DISCUSSÃO

Nesse capítulo será apresentado um estudo de caso, onde foi desenvolvido um aplicativo visando explorar os detalhes da especificação e codificação da simulação criada de acordo com o proposto nos capítulos anteriores.

Este trabalho utilizou um sistema para controle de uma agenda pessoal e os resultados obtidos indicam a viabilidade da utilização das tecnologias e mostram como deixaria o sistema com qualidade e segurança.

### 5.1. ANÁLISE GERAL DO SISTEMA

A aplicação móvel desenvolvida é uma agenda de contatos, onde a aplicação possui através de um servidor JEE cadastrar e armazenar todos contatos do usuário. Todos os serviços consumidos pela aplicação móvel é fornecido pelo servidor JEE. O sistema também detecta se um usuário é um contato e pergunta se ele quer disponibilizar a posição no *Google Maps* para os outros usuários que possuem ele poderem verificar onde esteve pela ultima vez que entrou na aplicação. Todas transferências dos serviços foram utilizados *Remote Object*, que utiliza um protocolo AMF disponível através do *BlazeDS*, dessa forma tendo um desempenho ótimo no trafego dos dados.

Como as empresas esta optando em incorporar os smartphones para auxiliar e aumentar a produtividade utilizando um servidor para o *back-end*, foi desenvolvido um aplicativo que tem por objetivo manter dados em um servidor caso não esteja com o dispositivo em mãos, mas há a necessidade de um contato no momento. Sendo assim os dados estarão por usuário e não por dispositivo móvel.

#### 5.1.1. Requisitos

Foi feito um levantamento de requisitos para analisar o que o sistema necessitaria:

- Manter usuários;
- Manter contatos;
- Vincular coordenada ao contato.

### 5.1.2. Caso de uso

O usuário é o próprio administrador, ele quem cria uma conta com usuário e senha para ser utilizada no sistema (Figura 8).

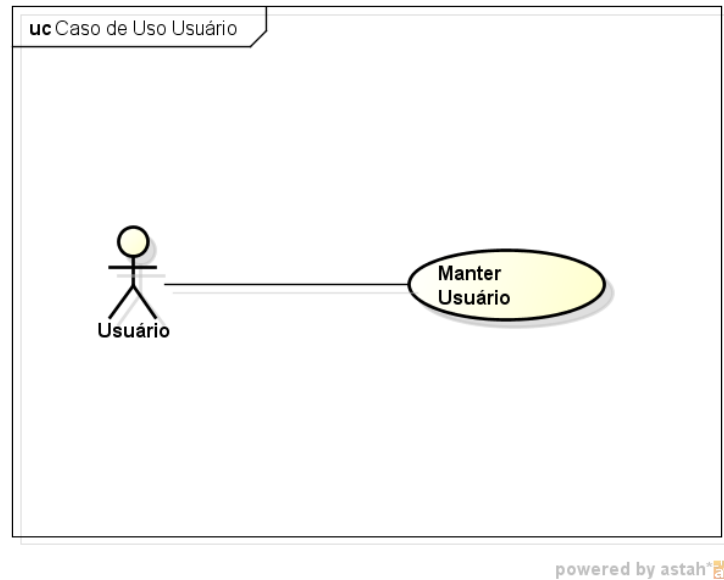


Figura 8 - Caso de uso do Usuário

O caso de uso de Contatos é feito pelo usuário, o usuário apenas cadastra o contato, podendo obter uma lista em ordem alfabética, detalhar um contato ou remover o contato (Figura 9).

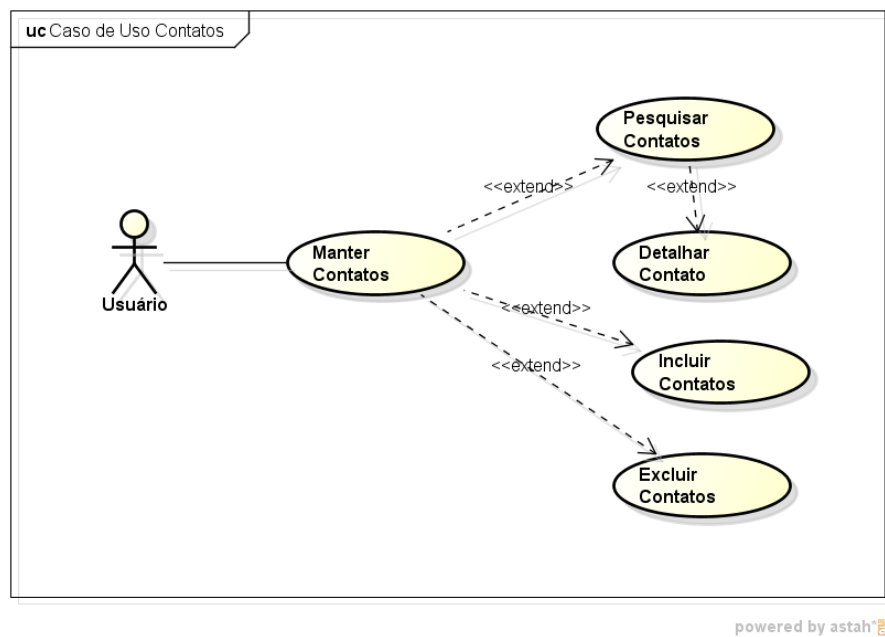


Figura 9 - Caso de uso Contatos

Já o vincular coordenada ao contato quem faz é o sistema, ele verifica se um contato é também um usuário e automaticamente grava a coordenada do contato (Figura 10).

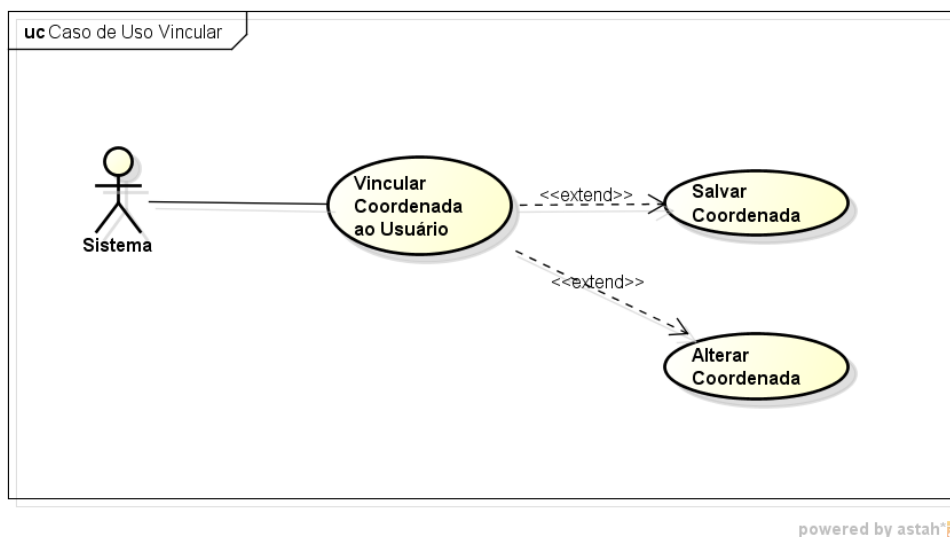


Figura 10 - Caso de uso coordenada ao contato

### 5.1.3. Diagrama de Atividade

Foi feito um diagrama de atividade para analisar e mostrar todo fluxo do controle de uma atividade para outra (Figura 11).

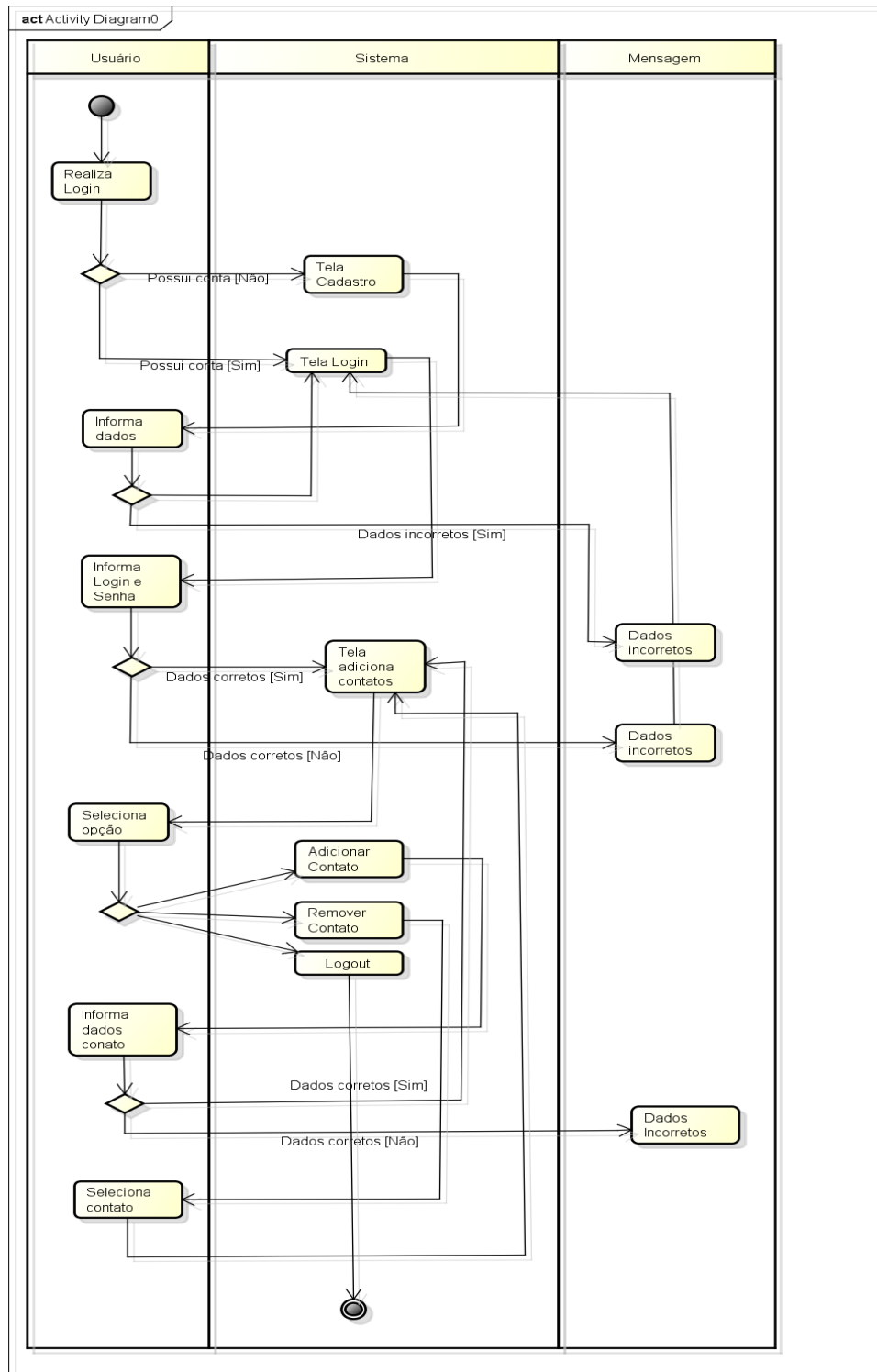


Figura 11 - Diagrama de atividade da aplicação

## 5.2. ARQUITETURA

Para a aplicação foi utilizada a seguinte arquitetura, o padrão *Model View Controller*. A arquitetura MVC foi originalmente desenvolvida para mapear as tarefas tradicionais de entrada, processamento e saída para o modelo de interação com o usuário. Usando o padrão MVC fica fácil mapear esses conceitos no domínio de aplicações Web multicamadas.

A aplicação ficou bem estruturada porque foi fornecida uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados (Figura 12).

Sendo assim todas chamadas de serviços serão delegados para o *controller*, ele trata a chamada e o retorno do método. A *view* apenas chama o *controller* (Quadro 6). Portanto a *view* apenas é encarregada de mostrar o conteúdo visual, e o *controller* encarregado de manipular a parte de negócios da aplicação (Quadro 7).

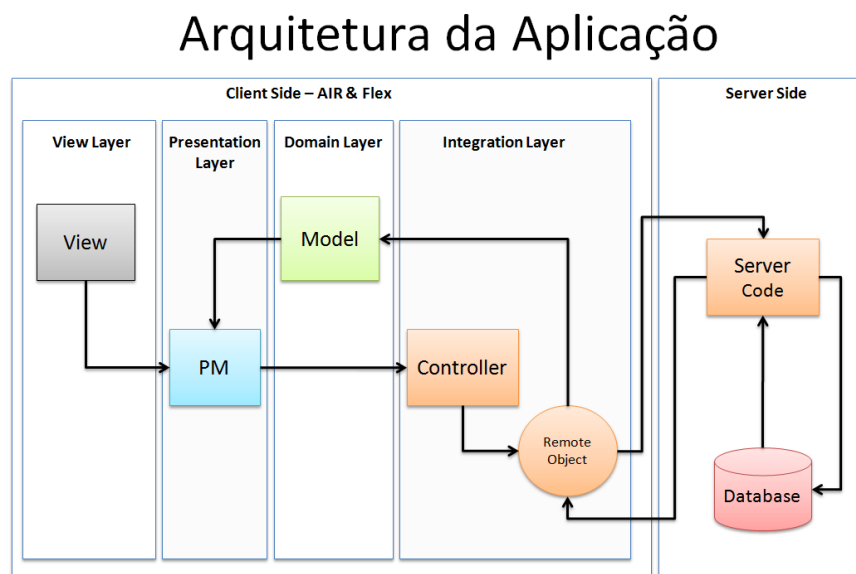


Figura 12 - Arquitetura da aplicação

```

17
18     import mx.collections.ArrayCollection;
19
20     private var usuarioCadastroController:UsuarioCadastroController;
21
22     private function createUsuario():void
23     {
24         usuarioCadastroController = new UsuarioCadastroController(this);
25
26         if( inputCadastrarSenha.text != inputConfirmarSenha.text )
27         {
28             this.errorLabelCadastro.text = "*Senha não confere";
29         }
30         else
31         {
32             usuario.contatos = new ArrayCollection();
33
34             usuarioCadastroController.useRemoteObject(usuario);
35         }
36     }
37

```

Quadro 6 - View chamando controller

```

23     private var userCadView:UsuarioCadastroView;
24
25     public var usuarioService:RemoteObject;
26
27     public function UsuarioCadastroController(user:UsuarioCadastroView)
28     {
29         this.userCadView = user;
30     }
31
32     public function useRemoteObject(user:Usuario):void
33     {
34         usuarioService = new RemoteObject();
35         usuarioService.destination = "usuarioService";
36         usuarioService.endpoint = ConfigService.END_POINT;
37         usuarioService.save.addEventListener("result", userCadResultHandler);
38         usuarioService.addEventListener("fault", faultHandler);
39         usuarioService.save(user);
40     }
41
42     public function userCadResultHandler(event:ResultEvent):void
43     {
44         this.userCadView.usuario = new Usuario;
45
46         this.userCadView.navigator.pushView(UsuarioLoginView);

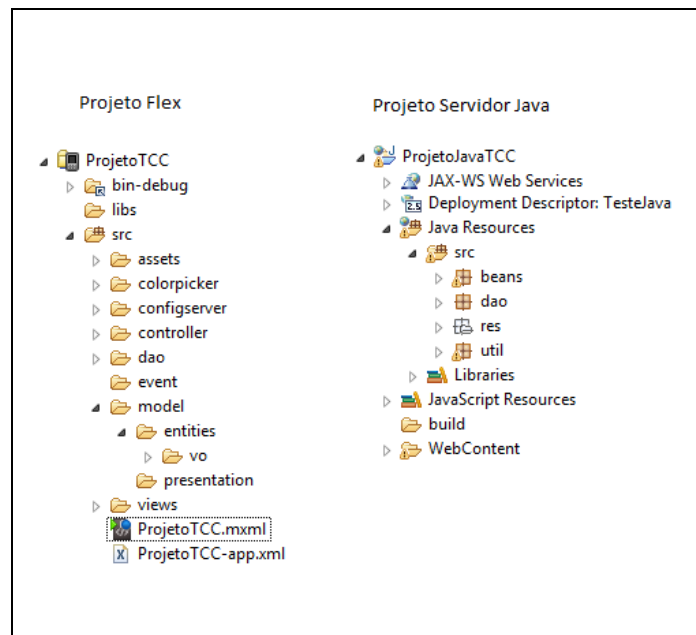
```

Quadro 7 - Controller tratando a chamada da view



### 5.3. CRIAÇÃO DO PROJETO

A aplicação possui um projeto para o *front-end* que é projeto desenvolvido em *Adobe Flex* para dispositivo móvel. E também há um projeto para o servidor da aplicação em *JavaEE* com *Tomcat* para ser possível executar todos serviços utilizados pelo dispositivo móvel. Foi desenvolvido projetos distintos em IDEs distintas, o projeto Java foi desenvolvido no Eclipse, já o projeto Flex foi desenvolvido no *Flash Builder*. A estrutura dos dois projetos pode ser visto no Quadro 8.



Quadro 8 - Estrutura do projeto Flex e Java

### 5.4. ARQUIVOS DE CONFIGURAÇÕES

#### 5.4.1. Arquivo de configuração do *hibernate*

A configuração do servidor foi com base em um arquivo do *Hibernate* de conexão ao banco de dados através do arquivo *Hibernate.cfg.xml*. Foram adicionadas as propriedades daquela sessão. Este arquivo também configura a definição da base de dados da aplicação (Quadro 9).

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.password"></property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/agendaonline</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

    <mapping class="beans.Usuario"/>
    <mapping class="beans.ContatosUsuario"/>

  </session-factory>
</hibernate-configuration>

```

Quadro 9 - Arquivo de configuração *Hibernate.cfg.xml*

#### 5.4.2. Arquivo de configuração de permissão do Android

Para criar a aplicação é necessário ter em mente as funcionalidades da que a aplicação irá utilizar, pois existe um arquivo padrão em todo projeto em *Android* conhecido como *manifest.xml* que possui todas as permissões possíveis que a aplicação poderá usufruir (Quadro 10). É possível definir as propriedades no momento da criação do projeto (Figura 13).

```

220
221 @<android>
222   <manifestAdditions><![CDATA[
223     <manifest android:installLocation="auto">
224       <!--See the Adobe AIR documentation for more information about setting Google Andr
225       <!--Removing the permission android.permission.INTERNET will have the side effect
226       of preventing you from debugging your application on your device-->
227       <uses-permission android:name="android.permission.INTERNET"/>
228       <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
229       <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
230       <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
231       <!--The DISABLE_KEYGUARD and WAKE_LOCK permissions should be toggled together
232       in order to access AIR's SystemIdleMode APIs-->
233       <uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
234       <uses-permission android:name="android.permission.WAKE_LOCK"/>
235       <uses-permission android:name="android.permission.CAMERA"/>
236       <uses-permission android:name="android.permission.RECORD_AUDIO"/>
237       <!--The ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE permissions should be toggled
238       together in order to use AIR's NetworkInfo APIs-->
239       <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
240       <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
241     </manifest>
242   ]]></manifestAdditions>
243 </android>
244

```

Quadro 10 - Arquivo de permissões *manifest.xml* da aplicação móvel

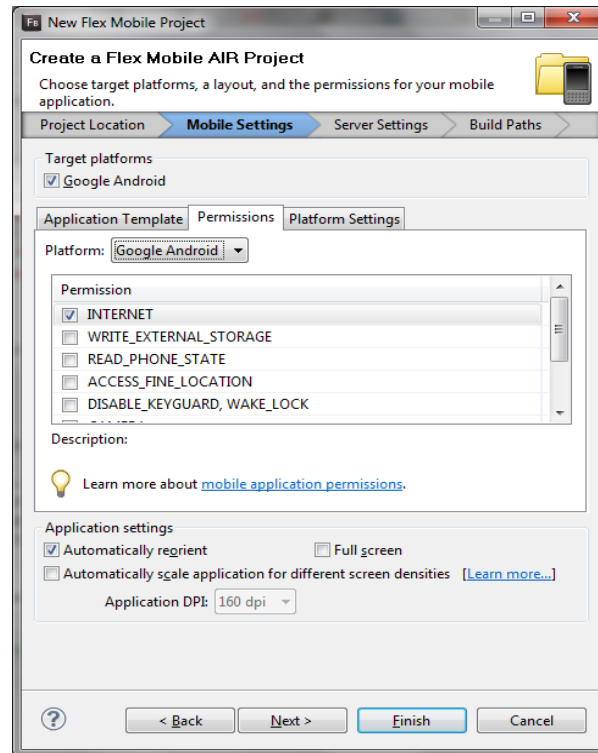


Figura 13 - Definindo permissões no momento de criação da aplicação

#### 5.4.3. Arquivo de configuração do BlazeDS

Para o *front-end* acessar os serviços do *back-end* foi necessário utilizar o BlazeDS, que faz justamente essa ponte entre a camada de visualização e camada de negócios. O BlazeDS fornece um arquivo de configuração chamado de *remoting-service*, que é fundamental para mapear todas classes dos serviços que serão utilizados (Quadro 11).

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <service id="remoting-service" class="flex.messaging.services.RemotingService">
3
4    <adapters>
5      <adapter-definition id="java-object"
6        class="flex.messaging.services.remoting.adapters.JavaAdapter"
7        default="true" />
8    </adapters>
9
10   <default-channels>
11     <channel ref="my-amf" />
12   </default-channels>
13
14   <destination id="loginService">
15     <properties>
16       <source>model.service.LoginService</source>
17     </properties>
18   </destination>
19
20   <destination id="usuarioService">
21     <properties>
22       <source>model.service.UsuarioService</source>
23     </properties>
24   </destination>
25
26   <destination id="contatoService">
27     <properties>
28       <source>model.service.ContatoService</source>
29     </properties>
30   </destination>
31
32 </service>

```

Quadro 11 - Arquivo XML principal do BlazeDS

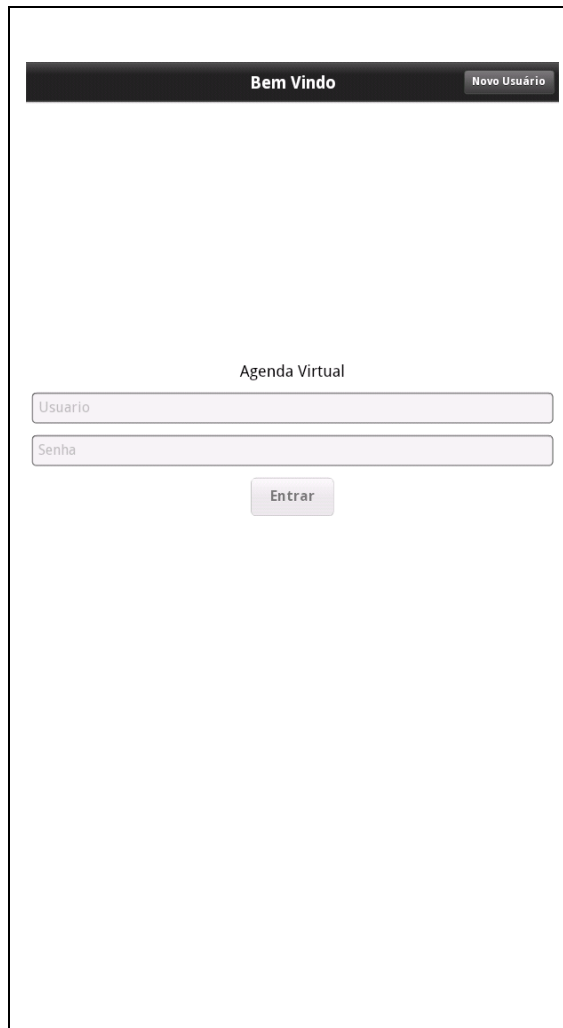
## 5.5. TELAS

As telas do sistema foram desenvolvidas utilizando a IDE Flash Builder, facilitando o desenvolvimento.

### 5.5.1. Login

Na tela de *login* o usuário necessita de um usuário e senha, caso não possua, o mesmo usuário poderá ser criado a partir do botão “Novo usuário” no canto superior à direita. (Figura 14).

Caso o *login* obtenha uma falha exibirá uma mensagem de erro (Figura 15).



The image shows a login screen for 'Agenda Virtual'. At the top, there is a dark header bar with the text 'Bem Vindo' on the left and a button labeled 'Novo Usuário' on the right. Below the header, the title 'Agenda Virtual' is centered. Underneath the title, there are two input fields: the first is labeled 'Usuario' and the second is labeled 'Senha'. Below these fields is a button labeled 'Entrar'.

Figura 14 - Tela de Login



The image shows the same login screen as in Figure 14, but with an error message. The input fields now contain the text 'Vini' and '\*\*\*\*'. Below the 'Entrar' button, there is a red error message that reads '\*Usuário Inválido'.

Figura 15 - Mensagem de erro

### 5.5.2. Cadastro usuário

Na tela de cadastro de usuário, o usuário precisa inserir os dados de forma válida caso algum dado não esteja da forma correta o sistema emite uma mensagem de erro avisando que há algum dado incorreto (Figura 16), caso contrário se executar o cadastro corretamente, automaticamente será encaminhado para tela de *login* novamente para entrar no sistema.



A imagem mostra a interface de usuário para o cadastro de um novo usuário. No topo, há uma barra de navegação com um botão 'Voltar' à esquerda e o título 'Cadastro' no centro. Abaixo, há seis campos de entrada de texto empilhados verticalmente, rotulados como 'Nome', 'Usuário', 'Senha', 'Confirmar Senha', 'Email' e 'Número Celular'. Cada campo possui uma borda arredondada e um ícone de lupa para pesquisa. No final do formulário, há um botão 'Cadastrar' centralizado.

Figura 16 - Cadastro usuário

### 5.5.3. Tela lista contatos

Logo após ter efetuado o login no sistema e com sucesso, irá mudar para uma tela de lista de contatos, como é a primeira vez não há contato cadastrado. No canto superior a direita existe dois botões para cadastrar um contato novo ou remover o contato desejado (Figura 17).

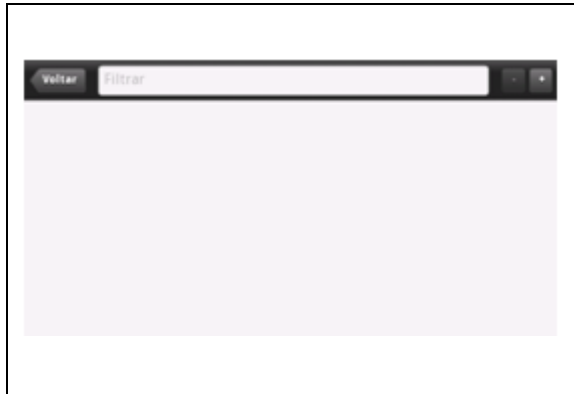


Figura 17 - Lista de contatos vazia

#### 5.5.4. Tela cadastro contato

Na tela de lista de contatos, se clicado no botão para um novo contato abrirá uma tela para inserção dos dados (Figura 18). Se os dados estiverem corretos o cadastro é efetuado e é automaticamente carregado na lista. Agora é possível ver a lista com os contatos criados em ordem alfabética (Figura 19).

**Voltar** **Cadastro**

Marlon

3525-2574

9985-1587

Número Comercial

Marlon@gmail.com

Cadast...

PT @ # 1 2 3 \_ % ( )  
q w e r t y u i o p  
& \$ 4 5 6 + ; : ' "  
a s d f g h j k l  
↑ z x c v b n m ←  
í SÍMB - \_ ' ↵

Figura 18 - Cadastro de contato

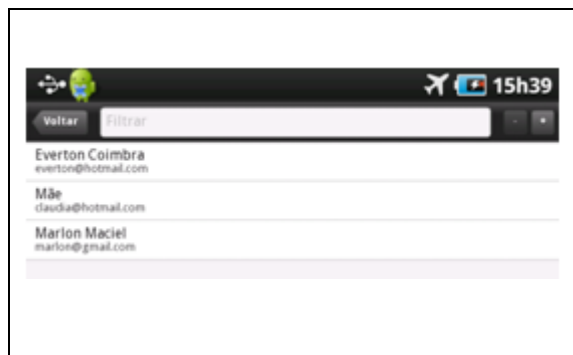


Figura 19 - Lista de contatos



#### 5.5.5. Vincular coordenada usuário.

Se um contato foi salvo pelo usuário, e este contato criar uma conta, automaticamente quando for efetuar o *login* aparecerá uma mensagem perguntando se deseja disponibilizar as coordenadas e a posição no *Google Maps* para os outros usuários que possuem ele como contato (Figura 20).

Por exemplo, se for criada uma conta com o usuário “Vinicius” e adicionar uma pessoa como contato “Everton”. Quando o contato “Everton” criar uma conta, o sistema irá vincular este usuário (“Everton”) com um contato (“Everton”) já existente por outro usuário (“Vinicius”), sendo assim possível detectar a coordenada e posição do usuário.



Figura 20 - Vincular coordenada ao contato

#### 5.5.6. Detalhar contato

Quando selecionar um contato da lista de contatos é possível detalhar e verificar os dados do contato. Se o contato for vinculado com a coordenada é possível verificar a latitude e longitude e também a posição no *Google Maps* (Figura 21). Caso escolha alguma opção o ícone à direita mostra a ação que será executada. Por exemplo, caso clique nos números de telefones, irá discar para o contato ou clique no email irá abrir algum aplicativo de email localizado no celular ou o próprio nativo passando por parâmetro o email do contato como destino.



Figura 21 - Detalhes do contato

#### 5.5.7. Google Maps

É possível verificar o local que o contato esteve pela ultima vez que entrou na aplicação. Isto é resultado da verificação que foi realizado no momento que o contato for entrar no sistema. Se ele disponibilizou a posição no mapa mostrará a posição, a data e hora do ultimo acesso realizado (Figura 22).

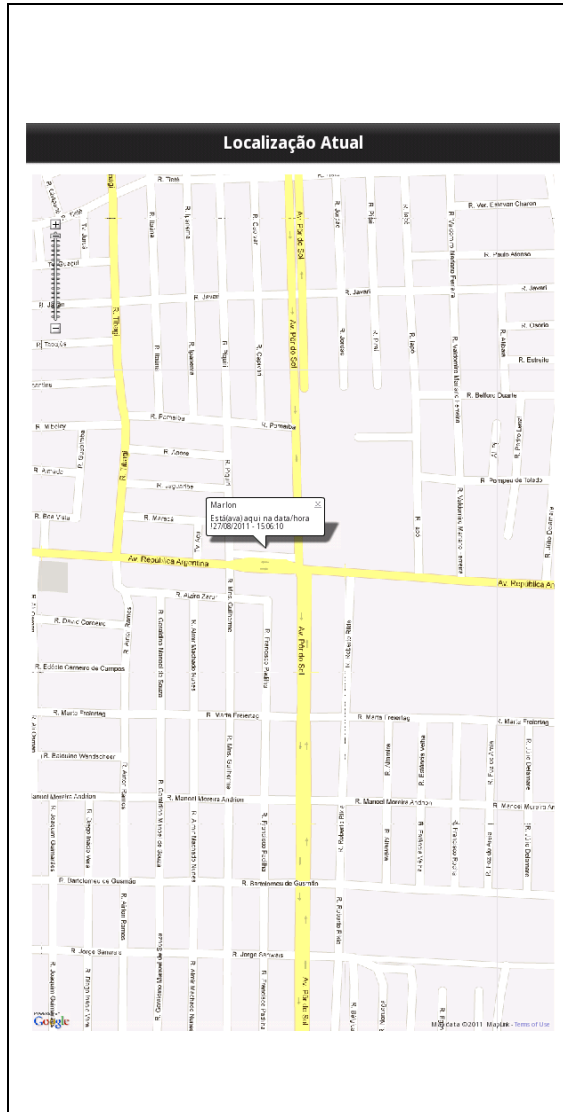


Figura 22 - Última posição do usuário no último acesso

## 6. CONSIDERAÇÕES FINAIS

### 6.1. CONCLUSÃO

Tendo em vista o crescimento de aplicações em *Android*, é possível ver que um dos motivos por optar por este tema é a facilidade e desempenho da aplicação em Flex para dispositivos móveis. Pode se utilizar um servidor como *back-end* obtendo uma maior segurança com um visual extremamente bonito. Este Trabalho utilizou uma aplicação para mostrar a simplicidade do desenvolvimento e a qualidade final do aplicativo.

Portanto os resultados obtidos, com a criação do projeto móvel e as pesquisas sobre o Android para o Adobe Flex comprovam a viabilidade da utilização do *Adobe Flex* com *Flash Builder* para a criação de aplicações em *Android*, pois desenvolvedores podem criar aplicações, utilizando a mesma linguagem que se utiliza para criar aplicações em *Flex* para *desktop* ou para *web*, obtendo o mesmo desempenho e qualidade. O *Adobe Flex* suporta muito bem várias APIs, principalmente com as APIs da Google permitindo aplicações com qualidade e segurança.

E com esse crescimento de dispositivos móveis e do Android há também a necessidade de mais aplicações, essas aplicações estão em constante crescimento, sendo o *Adobe Flex* é uma forma muito viável para a construção de aplicativos para *Android*.

## REFERÊNCIAS BIBLIOGRÁFICAS

**ADOBE. Adobe Flash Builder 4.5 and Flex 4.5 Accelerate Mobile App Development for Android, BlackBerry, Tablet OS and IOS**, 2011. Disponível em <<http://eon.businesswire.com/news/eon/20110410005103/en/Flash-Platform/Flex/Flash-Builder>>. Acesso em 20/04/2011.

**ADOBE. Flex SDK**, 2011. Disponível em <<http://opensource.Adobe.com/wiki/display/flexsdk/Flex+SDK>>. Acesso em 10/02/2011.

**ADOBE. AdobeFlash Platform, Packager for Iphone**, 2010. Disponível em <[http://help.Adobe.com/pt\\_BR/as3/iphone/index.html](http://help.Adobe.com/pt_BR/as3/iphone/index.html)>. Acesso em 10/02/2011.

**ADOBE. Mobile List, Scroller and Touch - Functional and Design Specification**, 2011. Disponível em <<http://opensource.Adobe.com/wiki/display/flexsdk/Mobile+List,+Scroller+and+Touch>>. Acesso em 20/04/2011.

**ANDROID. Download de Android SDK**, 2011. Disponível em <<http://developer.Android.com/sdk/index.html>>. Acesso em 03/05/2011.

**ANDROID. Upload Applications**, 2011. Disponível em <<http://www.google.com/support/Androidmarket/developer/bin/answer.py?hl=en&answer=113469>>. Acesso em 20/05/2011.

Apple Computer, I. (2006). **The Webkit Open Source Project**. Disponível em: <<http://webkit.org/projects/>>. Acessado em 07/05/2011.

AUGUSTO, Thiago. **Historia dos SmartPhones**, 2010. Disponível em <<http://tecnologia.culturamix.com/eletronicos/a-historia-da-smartphones>>. Acesso em 20/03/2011.

CHAIZE, Michael. **Create Custom Layouts for Mobile Apps in Flex 4.5**, 2011. Disponível em <<http://tv.Adobe.com/watch/adc-presents/create-custom-layouts-for-mobile-apps-in-Flex-45/>>. Acesso em 10/02/2011.

DEV MEDIA, **Dispositivos móveis e telefonia para 2011**, 2010. Disponível em <<http://www.devmedia.com.br/post-21157-Dispositivos-moveis-e-telefonia-para-2011.html>>. Acesso em 05/07/2011.

Google Inc. (2008). **Android - an open handset alliance project**. Disponível em: <<http://code.google.com/android/toolbox/custom-components.html>>. Acesso em 07/04/2011.

Google Inc. (2008). **Android open source**. Disponível em: <<http://source.android.com>>. Acesso em 15/05/2011.

Google Inc. (2009). **Application Fundamentals**. Disponível em: <<http://developer.android.com/guide/topics/fundamentals.html>>. Acesso em 03/07/2011.

HERRINGTON, Jack; KIM, Emily. **Getting Started with Flex 3**:  
*Adobe Development Team*: O'Really Media, 2008.

JARAMILLO, Narciso. **Mobile Development using Adobe Flex 4.5 SDK and Flash Builder 4.5**, 2011. Disponível em <<http://www.Adobe.com/devnet/Flex/articles/mobile-development-Flex-FlashBuilder.html>>. Acesso em 15/04/2011.

**JARAMILLO, Narciso**. Build Your First Mobile *Flex* Application - Twitter Trends , 2011. Disponível em < <http://www.Adobe.com/devnet/Flex/articles/twitter-trends.html>>. Acesso em 15/04/2011.

JESPERS, Serge. **Build your First Flex Application**, 2011. Disponível em < <http://tv.Adobe.com/watch/adc-presents/build-your-first-Flex-45-application/>>. Acesso em 07/02/2011.

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

LOOKOUT. **Platform Wars: Growth of Apps in The Android Market Outpaces Apple App Store**, 2011. Disponível em <<http://www.mylookout.com/appgenome#app-stores>>. Acesso em 17/04/2011.

RASMUSSEN, Bruna. **Android: o sistema móvel que conquistou o mundo**, 2011. Disponível em < <http://www.tecmundo.com.br/9010-Android-o-sistema-operacional-movel-que-conquistou-o-mundo.htm> >. Acesso em 20/03/2011.

RIBEIRO, Ramon. **Android: Um Novo paradigma de desenvolvimento móvel**. Publicado em: Revista WebMobile - edição 18.

SCHINSKY, Holly. **Flex 4.5 - Using Mobile View Transitions**, 2011. Disponível em < <http://devgirl.org/2011/05/12/Flex-4-5-using-mobile-view-transitions/>>. Acesso em 22/05/2011.

SCHMITZ, Daniel. **Dominando Flex e Java**. São Paulo: Daniel Schmitz, 2010.

SUBRAMANIAM, Deepa. **Introducing Adobe Flex 4.5 SDK**, 2011. Disponível em <<http://www.Adobe.com/devnet/Flex/articles/introducing-flex45sdk.html>>. Acesso em 03/05/2011.

TAPPER, Jeff; LABRIOLA, Michael; BOLES, Matthew. **Adobe Flex 3 Treinamento direto da Fonte**:  
Rio de Janeiro: Alta Books, 2009.

Tanenbaum, A. S. (2003). *Sistemas Operacionais Modernos*. Editora Pearson, 2 edition. São Paulo.

WIKIMEDIA FOUNDATION: **Wikipédia - The Free Encyclopedia**. *Dalvik* virtual machine. Disponível na Internet em: <[http://pt.wikipedia.org/wiki/Dalvik\\_virtual\\_machine](http://pt.wikipedia.org/wiki/Dalvik_virtual_machine)>. Acesso em 02/03/2011.