

**UNIVERSIDADE DO VALE DO ITAJAÍ
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA TERRA E DO MAR
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**UM COMPARATIVO ENTRE FRAMEWORKS PARA
DESENVOLVIMENTO MULTIPLATAFORMA MÓVEL**

Daniel Borges Ribeiro

Mathias Henrique Weber, M.SC.

Orientador

São José (SC), junho de 2014

**UNIVERSIDADE DO VALE DO ITAJAÍ
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA TERRA E DO MAR
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**UM COMPARATIVO ENTRE FRAMEWORKS PARA
DESENVOLVIMENTO MULTIPLATAFORMA MÓVEL**

Daniel Borges Ribeiro

São José (SC), junho de 2014

Orientador: Mathias Henrique Weber, M.Sc.

Área de Concentração: Mobile

Linha de Pesquisa: Desenvolvimento de Software para Dispositivos Móveis

Palavras-chave: *framework*, desenvolvimento multiplataforma, *PhoneGap*, *Titanium* , *jQuery Mobile*, iOS, Android, *smartphones*, dispositivos móveis

Número de páginas: 73

RESUMO

Atualmente existe uma série de diferentes sistemas operacionais utilizados em dispositivos móveis, tais como iOS, Android, Windows Mobile e BlackBerry OS (Operation System). O desenvolvimento de aplicativos para cada um desses sistemas utiliza linguagem de programação específica e SDKs (Software Development Kits) próprios, o que geralmente dificulta a criação de aplicativos que atendam o maior número de usuários possível, com os mais diversos modelos de dispositivos móveis. Para contornar essa dificuldade, causada pela diversidade desses sistemas operacionais, algumas empresas estão oferecendo soluções que possibilitam desenvolver o código do aplicativo uma única vez, e disponibilizar sua utilização para os principais sistemas operacionais. O presente trabalho analisou e comparou as principais soluções (frameworks) para o desenvolvimento de aplicativos multiplataforma móvel. Para tal foi desenvolvido um aplicativo utilizando três dos principais *frameworks* utilizados atualmente. A análise e levantamento das características desses *frameworks* poderá contribuir com a comunidade de desenvolvedores na escolha da solução que melhor atenda suas necessidades.

ABSTRACT

Nowadays there are a great number of different operating systems used on mobile devices such as iOS, Android, Windows Mobile and BlackBerry OS (Operation System). The development of applications for each one of these systems uses specific language programming and SDKs (Software Development Kits) specific, which often difficult the creation of applications that range the largest number of users that is possible, with very different models of mobile devices. To solve this difficulty, caused by the diversity of these operating systems, some companies are offering solutions that enable developing application code once, and make available its use for major operating systems. This study analyzed and compared the main solutions (frameworks) for the development of cross platform mobile applications. To this end, it was developed an application using three major frameworks currently used. The survey and analysis of the characteristics of these frameworks can contribute to the community of developers in choosing the solution that best meets your needs.

LISTA DE TABELAS

Tabela 1. Principais características dos Frameworks Multiplataforma Móvel	16
Tabela 2. Classificação dos <i>framewoks</i> segundo seu tipo de abordagem	20
Tabela 3. APIs disponíveis pelo <i>framework</i> PhoneGap para cada plataforma	22
Tabela 4. Análise de Funcionalidades.....	38
Tabela 5. Visual Não-Nativo x Nativo.....	44

LISTA DE QUADROS

Quadro 1. Abordagens dos frameworks.....	20
Quadro 2. Dados dos dispositivos onde foram feitos os testes	38
Quadro 3. Comportamento anômalo do recurso de escolher imagem no Android.....	39
Quadro 4. Descrição do resultado obtido nos diferentes navegadores	47

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ARM	Architecture Reference Manuals
ASF	Apache Software Foundation
CSS	Cascading Style Sheets
DOM	Document Object Model
EBSCO	Elton B. Stephens Co.
FMM	Framework(s) Multiplataforma Móvel
GPS	Global Positioning System
GUI	Graphic User Interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IET	Institution of Engineering and Technology
MV	Máquina Virtual
PC	Personal Computer
QML	Qt Modeling Language
RWD	Responsive Web Design
SDK	System Development Kit
SMS	Short Message Service
TCC	Trabalho de Conclusão de Curso
UNIVALI	Universidade do Vale do Itajaí
VM	Virtual Machine
W3C	World Wide Web Consortium

LISTA DE FIGURAS

Figura 1. Vendas de smartphones (2009 - 2016)	2
Figura 2. Resultado esperado: 6 instâncias do mesmo aplicativo protótipo.	4
Figura 3. Nível de conhecimento dos <i>frameworks</i> multiplataforma	15
Figura 4. Visão Geral do Sistema	29
Figura 5. Diagrama de Casos de Uso do Aplicativo	30
Figura 6. Tela de cadastro desenvolvida no jQuery Mobile	39
Figura 7. Botão de voltar do aparelho Samsung	40
Figura 8. Aplicativo no Opera Mini.....	40
Figura 9. Aplicativo no Google Chrome.....	40
Figura 10. Printscreen do comportamento anômalo do aplicativo gerado pelo jQuery Mobile	41
Figura 11. Visão Geral do iOS.....	42
Figura 12. Visão Geral do Android OS.....	43
Figura 13 Protótipo desenvolvido pelo jQuery Mobile rodando no Samsung Galaxy SIII e iPhone 4s	62
Figura 14 Protótipo desenvolvido pelo PhoneGap rodando no Samsung Galaxy SIII e iPhone 4s	63
Figura 15 Protótipo desenvolvido pelo Titanium rodando no iPhone 4s.....	64

SUMÁRIO

INTRODUÇÃO	1
1.1 PROBLEMA DE PESQUISA.....	2
1.1.1 Solução Proposta	3
1.1.2 Delimitação de Escopo	3
1.1.3 Justificativa.....	4
1.2 OBJETIVOS	5
1.2.1 Objetivo Geral.....	5
1.2.2 Objetivos Específicos	5
1.3 METODOLOGIA.....	6
1.3.1 Metodologia da Pesquisa	6
1.3.2 Metodologia da Aplicação	7
1.4 ESTRUTURA DO TRABALHO.....	9
2 FUNDAMENTAÇÃO TEÓRICA.....	10
2.1 PLATAFORMA.....	10
2.2 FRAMEWORK	10
2.2.1 Características de Frameworks	10
2.3 FRAMEWORK MULTIPLATAFORMA	11
2.4 PRINCIPAIS FRAMEWORKS MULTIPLATAFORMA MÓVEL.....	12
2.5 PHONEGAP.....	20
2.5.1 Visão Geral	20
2.5.2 Características	21
2.6 TITANIUM	22
2.6.1 Visão Geral	23
2.6.2 Características	23
2.7 JQUERY MOBILE	23
2.7.1 Visão Geral	23
2.7.2 Características	24
3 TRABALHOS RELACIONADOS	24
3.1 CROSS-PLATFORM DEVELOPMENT TOOLS FOR SMARTPHONE APPLICATIONS.....	25
3.2 DEVELOPING MOBILE APPS USING CROSS-PLATFORM FRAMEWORKS: A CASE STUDY	26
4 DESENVOLVIMENTO.....	28
4.1 VISÃO GERAL DO SISTEMA	29
4.2 MODELAGEM DO SISTEMA.....	30
4.2.1 Diagrama de Casos de Uso	30
4.2.2 Descrição dos Casos de Uso.....	30
4.3 DETALHAMENTO DO DESENVOLVIMENTO.....	32

4.4 DESCRIÇÃO DOS EXPERIMENTOS.....	34
4.5 RESULTADOS	38
4.5.1 Análise das Funcionalidades	38
4.5.2 Análise do Resultado da Tabela.....	38
4.6 CONSIDERAÇÕES	45
4.6.1 jQuery Mobile	45
4.6.2 PhoneGap.....	48
4.6.3 Titanium.....	50
4.7 A ESCOLHA DA MELHOR SOLUÇÃO.....	52
5 CONCLUSÕES.....	54
5.1 TRABALHOS FUTUROS	54
REFERÊNCIAS BIBLIOGRÁFICAS.....	56
APÊNDICE A – TELAS DO PROTÓTIPO AGENDA	60
APÊNDICE B – IMAGENS DO APLICATIVO.....	62

INTRODUÇÃO

Tem-se observado o crescimento do uso de dispositivos móveis em todo o mundo. Dispositivos como *tablets* e *smartphones* estão chegando ao mercado com preços mais acessíveis. No caso mais específico dos *smartphones*, a queda dos preços aliada ao aumento na qualidade dos serviços oferecidos pelas operadoras têm contribuído para esse crescimento.

Estima-se que foram vendidas 958,8 milhões de unidades desses dispositivos no mundo em 2013 (FRAMINGHAM, 2013) e que haja um crescimento de vendas médio anual de 13,3% entre 2013 e 2017. A previsão de crescimento de vendas nos Estados Unidos é de 32,7% em 2013, em relação a 2012 (IDC WORLDWIDE SMARTPHONE TRACKER, 2013).

Os usuários desses dispositivos demandam cada vez mais por aplicativos que lhes proporcione melhores experiências, seja para entretenimento, atividades do cotidiano ou atividades profissionais, o que gera oportunidades para desenvolvimento de aplicativos tanto para uso pessoal quanto para uso corporativo. A necessidade de profissionais que desenvolvam aplicativos para dispositivos móveis cresce proporcionalmente à tal demanda.

Esse novo nicho de mercado tem trazido desafios significativos para os desenvolvedores de software. Eles se veem diante de uma nova realidade em que as soluções computacionais são muito diferentes das até então conhecidas.

Um estudo feito pela empresa Vision Mobile (JONES, 2012) apontou mais de cem diferentes *frameworks* para desenvolvimento multiplataforma móvel. Se por um lado a grande disponibilidade proporciona maior liberdade ao desenvolvedor, por outro dificulta-se a escolha da melhor solução.

O presente trabalho foi desenvolvido com o propósito de fazer o levantamento dos *frameworks* mais utilizados e desenvolver um protótipo de aplicativo nos *frameworks* jQuery Mobile, PhoneGap e Titanium, para compará-los quanto às suas vantagens, desvantagens, limitações e eficiência no desenvolvimento de aplicativos para diferentes plataformas móveis.

O gráfico apresentado na Figura 1 comprova o crescimento acelerado das vendas de *smartphones*. Ele aponta um crescimento médio anual de 43% no período de 2010 a 2016. “Os *smartphones*, *tablets* e dispositivos de dados se expandiram paralelamente ao aumento das conexões 3G no Brasil” (CABELLO & PHILLIPS, 2012). Cabello e Phillips (2012), ao analisar dados

fornecidos pela Wireless Intelligence (2012), concluíram que “houve uma queda de 39% no custo efetivo da banda larga móvel dos anos de 2009 a 2012 devido ao aumento da concorrência e da oferta de serviços para esses dispositivos”.

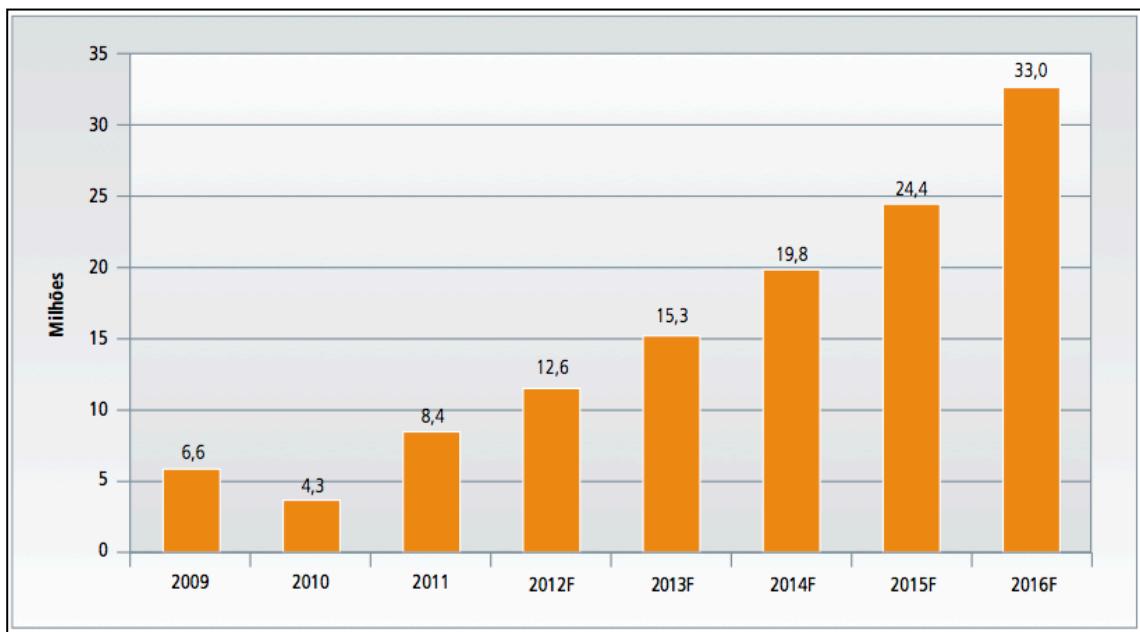


Figura 1. Vendas de smartphones (2009 - 2016)

Fonte: Adaptado de Cabello e Phillips (2012)

1.1 PROBLEMA DE PESQUISA

Como consequência do aumento de dispositivos móveis, Sistemas Operacionais (SO) são criados para gerenciar tais dispositivos. Os principais são: Android, bada, BlackBerry, iOS, MeeGo, Symbian, webOS e Windows Phone (Ohrt & Turau, 2012). Cada SO oferece ferramentas de desenvolvimento próprias (ou nativas), com características específicas, tais como: linguagem de programação, IDEs (*Integrated Development Environment*), frameworks e bibliotecas nativas. Portanto, disponibilizar o aplicativo para mais de um dispositivo móvel requer um desenvolvimento específico para cada sistema operacional, o que demanda muito tempo e gera um custo de produção alto.

Estudos apontam que o desenvolvimento de aplicativos utilizando frameworks multiplataforma geralmente facilita a abrangência do maior número de usuários, dos mais diversos dispositivos, de maneira mais rápida, prática e competitiva (Ohrt & Turau, 2012); (Humayoun, Ehrhart, & Ebert, 2013). Sua utilização possibilita menores custos de desenvolvimento, disponibilização e entrega do

produto em diversas plataformas, com diferentes sistemas operacionais, com qualidade satisfatória para o usuário final. Porém, necessitam melhorias (OHRT & VOLKER, 2012).

Uma pesquisa feita pela empresa Vision Mobile (JONES, 2012) apontou mais de cem diferentes *frameworks* para desenvolvimento multiplataforma móvel. Se por um lado a grande disponibilidade proporciona maior liberdade ao desenvolvedor, por outro dificulta-se a escolha da melhor solução.

- Será que os *frameworks* multiplataforma, na prática, realmente são capazes de disponibilizar a mesma aplicação nas diferentes plataformas de maneira eficiente e eficaz?

1.1.1 Solução Proposta

O presente trabalho foi desenvolvido com o propósito de fazer o levantamento dos *frameworks* mais utilizados e desenvolver um protótipo de aplicativo nos *frameworks* jQuery Mobile, PhoneGap e Titanium, para compará-los quanto às suas vantagens, desvantagens, limitações e eficiência no desenvolvimento de aplicativos para diferentes plataformas.

1.1.2 Delimitação de Escopo

Dentre as plataformas mais populares, segundo resultados de uma pesquisa que envolveu aproximadamente 1500 profissionais experientes da área de desenvolvimento para *mobile* (Web Directions, 2011), estão as plataformas que utilizam os sistemas iOS, Android, Blackberry e Windows Phone. Dentre elas escolheu-se para utilizar no presente trabalho as duas principais. São elas: o iPhone 4s, rodando iOS 7.x da Apple e o Samsung Galaxy SIII, rodando Android 4.x da Google. As plataformas da Apple e da Google lideram o mercado, seguidas pelas plataformas da BlackBerry com o BlackBerry OS e da Microsoft com Windows Phone OS (Web Directions, 2011), porém não foram contemplados os sistemas BlackBerry e Windows Phone.

Foram utilizados os *frameworks* jQuery Mobile, PhoneGap e Titanium. Cada um deles irá gerar duas instâncias do protótipo para executar nas plataformas escolhidas.

O protótipo implementado foi uma agenda telefônica que irá abranger os principais recursos utilizados nos *smartphones* como: câmera fotográfica, persistência de dados, geolocalização, manipulação de listas, envio de SMS, de e-mails, realização ligações, manipulação de imagens, entre

outros. Ao final do desenvolvimento gerou-se seis instâncias do mesmo aplicativo que foram avaliadas nos diferentes ambientes operacionais.

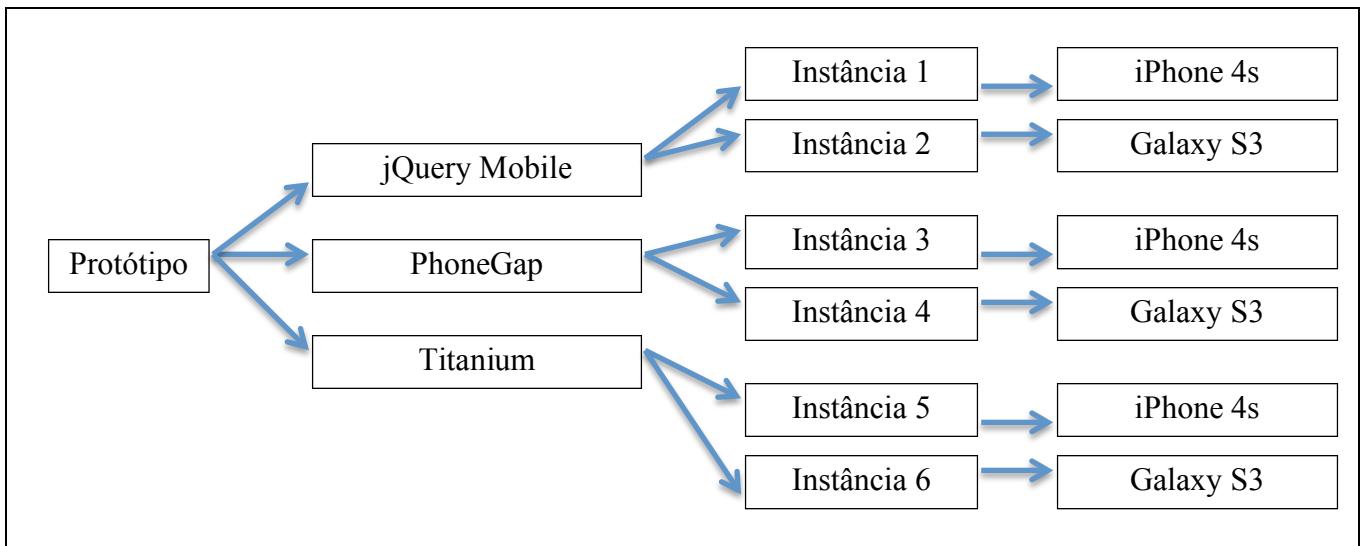


Figura 2. Resultado esperado: 6 instâncias do mesmo aplicativo protótipo.

O experimento foi avaliado através da realização de testes de funcionalidade, afim de relatar se o aplicativo responde como esperado. Além da análise das funcionalidades do sistema do ponto de vista do usuário, foram considerados os pontos positivos e negativos de cada *framework* do ponto de vista do desenvolvedor.

Não fez parte dessa avaliação testes de código ou testes automatizados.

1.1.3 Justificativa

O presente trabalho não pretende concluir qual a melhor solução (*framework*), pois o autor entende que não há uma solução que seja a melhor independentemente do contexto, mas sim que exista soluções que melhor atendam as necessidades do desenvolvedor de acordo com seu contexto de desenvolvimento. Fazem parte do contexto do desenvolvimento de um aplicativo variáveis como: tempo, custos e ferramentas disponíveis. Por isso, escolheu-se frameworks com diferentes propostas, para que ao final deste trabalho se possa listar as características de cada um com o intuito de possibilitar, como já foi dito, que o leitor, baseado no contexto de sua necessidade, seja capaz de escolher a melhor solução que o atenda.

Pode-se observar durante o levantamento bibliográfico, trabalhos semelhantes a este, porém entende-se que por haver uma grande quantidade de *frameworks*, os trabalhos já existentes

contemplam uma parcela muito pequena deles. Trabalhar com a ferramenta certa é fundamental para a produtividade do desenvolvedor. Esse fato é comprovado pelo elevado valor que empresas de análise de mercado cobram por relatórios sobre essas soluções.

Existem mais de cem *frameworks* com a proposta de desenvolver aplicativos para múltiplas plataformas móveis (© research2guidance, 2013). E frequentemente são lançados novos com novas propostas. Eles estão em constante atualização para acompanhar o ritmo de mudanças do mercado *mobile*. Trabalhos que analisem essas soluções podem contribuir com a comunidade de desenvolvedores uma vez que trazem informações atualizadas das soluções existentes no mercado.

Este trabalho contribuiu com a disponibilização gratuita e atualizada da análise e comparação de três *frameworks*, escolhidos estratégicamente com intuito de abranger uma gama maior de *frameworks* com abordagens semelhantes. O Capítulo 2 explica o conceito de abordagem da forma que é utilizado aqui.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O presente trabalho visa comparar e avaliar soluções (*frameworks*) para o desenvolvimento de aplicativos multiplataforma móvel.

1.2.2 Objetivos Específicos

Os objetivos específicos deste projeto de pesquisa são:

1. Pesquisar e analisar *frameworks* existentes;
2. Determinar os requisitos para comparação dos *frameworks*;
3. Caracterizar um aplicativo que considere diferentes recursos existentes em *smartphones*;
4. Realizar a modelagem do protótipo;
5. Implementar um protótipo em cada um dos três *frameworks* escolhidos e implantá-los nas plataformas iPhone 4s e Samsung Galaxy S3;
6. Testar e validar as funcionalidades dos protótipos; e
7. Documentar o desenvolvimento e os resultados da comparação.

1.3 METODOLOGIA

Barros e Lehfeld (2000) definem metodologia como “[...] um conjunto de procedimentos a serem utilizados na obtenção do conhecimento. É a aplicação do método através de processos e técnicas que garante a legitimidade do saber obtido” (BARROS & LEHFELD, 2000).

Galliano (1979, p. 6) conceitua método como “um conjunto de etapas ordenadamente dispostas a serem vencidas na investigação da verdade, no estudo de uma ciência ou para alcançar determinado fim”. No contexto de trabalhos de ciência da computação, os métodos científicos são utilizados, em geral, com o intuito de criar uma base para a especificação de um sistema de informação (Wainer, 2007).

“O método estabelece de modo geral o que fazer, e a técnica nos dá o como fazer, isto é, a maneira mais hábil, mais perfeita de fazer uma ação.” (BARROS & LEHFELD, 2000, p. 4)

Para melhor compreensão dos métodos que foram utilizados neste projeto esta seção foi dividida em metodologia da pesquisa e metodologia da aplicação.

1.3.1 Metodologia da Pesquisa

O trabalho baseia-se na revisão bibliográfica e aplicação de estudos empíricos. A revisão bibliográfica foi aplicada para os principais temas necessários para execução deste trabalho, com objetivo de cobrir os principais conceitos de cada área, que são: Sistemas Operacionais, Desenvolvimento *Mobile* e Banco de Dados. Com a aplicação dos estudos empíricos pretende-se apresentar uma contribuição na verificação da efetividade do uso de *frameworks* multiplataforma para desenvolvimento *mobile*.

Kelinger (1980) definiu empírico como “guiado pela evidência obtida em pesquisa científica sistemática e controlada” e Ruiz (1996) definiu pesquisa científica como “[...] investigação planejada, desenvolvida e redigida de acordo com as normas da metodologia consagradas pela ciência”. Wazlawick (2009) observando sob o ponto de vista mais específico da Ciência da Computação afirma que “a computação, enquanto ciência, fundamenta suas pesquisas no empirismo [...] na maioria das vezes o que importa são as conclusões objetivas obtidas empiricamente”.

O método utilizado para esta pesquisa é o indutivo, método que parte de observações particulares constatadas em um grupo para inferir e definir uma verdade geral provável (KAPLAN,

1980). A indução é baseada nas leis do determinismo, definindo que sob as mesmas circunstâncias, as mesmas causas produzem os mesmo efeitos (HEGENBERG, 1976). Por meio da avaliação proposta foram verificadas as hipóteses de pesquisa. Com isso esta pesquisa ganha um caráter indutivo experimental, uma vez que a indução sobre os resultados particulares será sobre um experimento formal.

Quanto à sua natureza esta pesquisa é classificada como aplicada, dado que se objetiva gerar conhecimentos para aplicação prática dirigida à solução de problemas específicos. Neste contexto, esta pesquisa busca verificar a utilização de *frameworks* multiplataforma móvel(FMM) como resposta à parte dos desafios enfrentados no desenvolvimento mobile.

Em relação à abordagem do problema, este teve um tratamento qualitativo, visto que a avaliação foi realizada por meio de uma verificação das funcionalidades do aplicativo gerado, se elas respondiam como esperado, ou seja, a qualidade do aplicativo gerado.

Do ponto de vista de seus objetivos, este trabalho pode ser caracterizado como pesquisa exploratória, já que “visa proporcionar maior familiaridade com o problema com intuito de torná-lo explícito ou a construir hipóteses” (Menezes & Silva, 2005). Sendo assim, utilizou-se o procedimento técnico de pesquisa bibliográfica com o objetivo de construir a fundamentação teórica necessária para este trabalho. Foram utilizados essencialmente materiais publicados, como artigos, relatórios desenvolvidos por empresas de telecomunicação, de tecnologia, de pesquisa de mercado, além de livros relacionados a Sistemas Operacionais, Desenvolvimento de Software para Dispositivos Móveis e à documentação disponibilizada pelas empresas que desenvolvem os *frameworks*.

Quanto aos procedimentos técnicos da pesquisa, esta é classificada como bibliográfica e experimental. A pesquisa bibliográfica tem como objetivo compreender melhor os temas abordados, enquanto que a face experimental desta pesquisa visa verificar as hipóteses e responder as perguntas de pesquisa levantadas neste trabalho.

1.3.2 Metodologia da Aplicação

Para o cumprimento dos objetivos estabelecidos neste trabalho foram aplicadas as etapas:

1. **Estudo bibliográfica:** para atender o Objetivo Específico 1, foi feita uma revisão bibliográfica sobre os assuntos *frameworks*, plataformas móveis, desenvolvimento multiplataforma e desenvolvimento mobile. Desta forma foi possível criar uma base para

estudar as características dos FMM, tendo em vista que este trabalho não irá propor um novo *framework* e sim utilizar os já existentes no mercado.

2. **Levantamento das características dos frameworks:** para atender o Objetivo Específico 2, com base no estudo bibliográfico foram estabelecidas as principais características dos *frameworks* multiplataforma móveis afim de compará-los e analisá-los.
3. **Caracterização do protótipo:** para atender o Objetivo Específico 3, utilizando como referência os principais recursos de um *smartphone* tendo como base trabalhos correlatos durante o estudo bibliográfico, foi caracterizado um protótipo que contemple tais recursos afim de analisar e comparar seu comportamento na prática ao ser gerado por diferentes frameworks e implantado em múltiplas plataformas.
4. **A modelagem do protótipo:** para atender o Objetivo Específico 4, a modelagem considerou basicamente os casos de uso. Por se tratar de um aplicativo muito simples, decidiu-se não aprofundar na documentação da modelagem através de diagramas. Uma visão geral do aplicativo foi apresentada utilizando como artifício o diagrama de atividades. Como não existe um diagrama de visão geral, optou-se por essa alternativa que é considerada e utilizada por alguns autores da área de Engenharia de Software.
5. **Desenvolvimento do Protótipo:** para atender o Objetivo Específico 5, considerando que os frameworks escolhidos são capazes de gerar instâncias do mesmo aplicativo para diferentes plataformas, foram gerados dois aplicativos por framework, um para iPhone 4s e outro para Samsung Galaxy SIII. O usuário final terá seis aplicativos com as mesmas funcionalidades, porém com algumas modificações no *layout*.
6. **Os testes e avaliações:** para atender o Objetivo Específico 6, utilizando seis aplicativos, três em cada plataforma, foram realizados testes de funcionalidade, afim de relatar se o aplicativo responde como esperado. Além da análise das funcionalidades do sistema do ponto de vista do usuário, foram considerados os pontos positivos e negativos de cada framework do ponto de vista do desenvolvedor.
7. **Relatar o resultado:** para atender o Objetivo Específico 7, tendo o resultado do teste das funcionalidades do sistema e as considerações do desenvolvedor sobre as características de cada *framework*, esses serão compilados e apresentados como conclusão do trabalho.

1.4 ESTRUTURA DO TRABALHO

O trabalho está organizado em cinco capítulos correlacionados. O Capítulo 1, Introdução, apresentou por meio de sua contextualização o tema proposto neste trabalho. Da mesma forma foram estabelecidos os resultados esperados por meio da definição de seus objetivos e apresentadas as limitações do trabalho permitindo uma visão clara do escopo proposto.

No Capítulo 2 é apresentada a fundamentação teórica, que aborda os temas relacionados ao desenvolvimento para dispositivos móveis e apresenta os principais FMM. A fundamentação teórica contempla os conceitos fundamentais para a análise dos principais frameworks atualmente utilizados. Esses conceitos formam a base para a caracterização do protótipo.

O Capítulo 3 apresenta o estudo baseado em trabalhos já realizados, permitindo, através da análise desses, o conhecimento das principais dúvidas da comunidade científica sobre o desenvolvimento *mobile* para múltiplas plataformas, e também conhecer o que já foi tratado e o que não foi, além dos critérios que geralmente se utiliza para comparar os frameworks.

No Capítulo 4 é apresentado o desenvolvimento do aplicativo. Inicialmente é feita a modelagem do protótipo, onde são levantados os casos de uso que lista as funcionalidades do sistema. Em seguida é apresentado o fluxo do sistema, onde são mostradas as telas do aplicativo, assim como seu fluxograma. Este capítulo também apresenta o método utilizado para implementar e implantar o protótipo nas diferentes plataformas, através dos *frameworks* escolhidos.

Ainda no Capítulo 4 são apresentados os resultados dos testes e análises realizados. Uma avaliação qualitativa do sistema foi feita através de questionários respondidos pelos usuários que experimentaram os aplicativos.

No Capítulo 5 são feitas as conclusões do trabalho relacionando os objetivos identificados inicialmente com os resultados alcançados. São ainda propostas possibilidades de continuação da pesquisa desenvolvida a partir das experiências adquiridas com a execução do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo tem por objetivo contemplar conceitos fundamentais para a análise dos principais *frameworks* atualmente utilizados.

2.1 PLATAFORMA

Neste trabalho foram contempladas as duas principais plataformas utilizadas por desenvolvedores de aplicativos para *smartphones*. São elas: iPhone 4s da empresa americana Apple e Samsung Galaxy S3 da empresa coreana Samsung.

“Uma plataforma computacional consiste em uma coleção de recursos de hardware, como o processador, memória principal, módulos de entrada e saída, *timers*, drives de disco e assim por diante” (Stallings, 2012, p. 108) .

Bishop e Horspool definiram o termo plataforma como uma restrição da linguagem de programação, do sistema operacional ou do computador, ou seja, do hardware (Bishop, Horspool, & Pretoria, 2006).

2.2 FRAMEWORK

Sendo o *framework* o principal componente das soluções para desenvolvimento *mobile*, sobre o qual se baseiam os demais recursos, um bom entendimento de seus princípios torna-se fundamental.

No presente trabalho foram utilizados os *frameworks* jQuery Mobile, PhoneGap e Titanium.

“Um *framework* é um conjunto de classes cooperativas que constroem um projeto reutilizável para uma determinada categoria de software” (Gamma, Helm, Johnson, & Vlissides, 2000, p. 41).

2.2.1 Características de Frameworks

Para melhor caracterizar a forma de atuação de um *framework*, o trecho abaixo foi extraído do livro Padrões de Projeto (Gamma, Helm, Johnson, & Vlissides, 2000):

“Um *framework* predefine parâmetros de projetos, de maneira que os desenvolvedor, possa se dedicar aos aspectos específicos da sua aplicação”.

Um *framework* captura as decisões de projeto que são comuns ao seu domínio de aplicação. Assim, *frameworks* enfatizam a reutilização de projetos em relação à reutilização de código, embora um *framework*, geralmente, inclua subclasses concretas que se pode utilizar diretamente. A reutilização neste nível leva a uma inversão de controle entre a aplicação e o software sobre o qual ela está baseada. Inversão de controle é uma arquitetura de software, na qual, por exemplo, o aplicativo registra funções de *callback* que são chamadas para determinados eventos, em vez de só chamar funções de uma biblioteca diretamente (Sommer, 2012).

Quando se utiliza um *toolkit* (ou uma biblioteca convencional de sub-rotinas), escreve-se o corpo principal da aplicação e chama o código que se quer reutilizar. Por outro lado, ao se usar um *framework*, reutiliza-se o corpo principal e escreve-se o código que o *framework* chama. São escritas operações com nomes e convenções de chamada já especificadas; porém isso reduz as decisões de projeto que devem ser tomadas.

Como resultado, pode-se não somente construir aplicações mais rapidamente, como também construí-las com estruturas similares. Elas são mais fáceis de se manter e parecem mais consistentes para os usuários. Por outro lado, perde-se alguma liberdade criativa, uma vez que muitas decisões de projeto já terão sido tomadas.

Os *frameworks* estão se tornando cada vez mais comuns e importantes. São a maneira pela qual os sistemas orientados a objetos conseguem a maior reutilização. Aplicações orientadas a objetos maiores terminarão por consistir-se de camadas de *frameworks* que cooperam uns com os outros. A maior parte do projeto e do código da aplicação virá dos *frameworks* que ela utiliza ou será influenciada por eles.”

2.3 FRAMEWORK MULTIPLATAFORMA

Os *frameworks* aqui abordados se restringirão àqueles que contemplam múltiplas plataformas.

Um *framework* multiplataforma tem como principal objetivo acelerar o desenvolvimento de aplicações para as diversas plataformas existentes de forma que o desenvolvedor escreva o código uma vez e com pequenas alterações rode nas diferentes plataformas. Esta técnica evitaria com que o desenvolvedor precisasse saber Objective-C para desenvolver para as plataformas da Apple, Java para as plataformas contempladas pelo sistema operacional Android, C# para Windows Phone e assim por diante. Para criar aplicações para as diversas plataformas utilizando PhoneGap, Titanium e jQuery

Mobile, o desenvolvedor precisa conhecer principalmente as linguagens HTML e JavaScript e poderá distribuir sua aplicação para as plataformas citadas anteriormente.

2.4 PRINCIPAIS FRAMEWORKS MULTIPLATAFORMA MÓVEL

Dentre as plataformas computacionais existentes como: notebooks, tablets, consoles de *games*, computadores desktop, celulares, entre outros, este trabalho se restringe às plataformas móveis, mais especificamente aos *smartphones* e consequentemente abordará somente os FMM.

Um estudo feito pela Vision Mobile (Jones, 2012) apontou mais de cem diferentes *frameworks* para desenvolvimento multiplataforma móvel. Com a crescente demanda de soluções que acelerem a produtividade e portabilidade das aplicações, novos frameworks estão sendo lançados com diferentes propostas. Os principais para desenvolvimento multiplataforma mobile, de acordo com Dubray (2011), são: PhoneGap(Apache Cordova), Sencha Touch, Mono, Appcelerator, Adobe AIR, Qt, Rhomobile, Marmelade, Corona, MoSync, jQuery Mobile e jQTouch.

De acordo com Dio-Synodinos (2012) os mais importantes e maduros *frameworks* para desenvolvimento multiplataforma são:

1. PhoneGap (Apache Cordova) (PhoneGap, 2013a) é um *framework open source* para a criação rápida de aplicações multiplataforma móveis usando HTML5, Javascript e CSS.
2. Sencha Touch (Sencha, 2013) é um *framework* para desenvolvimento de aplicações HTML5 móveis. Tem uma estrutura que permite aos desenvolvedores criar aplicativos que funcionam nos SOs: iOS, Android, BlackBerry, Windows Phone, e muito mais.
3. Xamarain¹ (Xamarain, 2013) disponibiliza 100% das APIs do iOS e Android através de C#. Os aplicativos são escritos em C#, o qual possibilita acessar as APIs nativas diretamente dele. O compilador Xamarin utiliza uma *runtime* .NET que gera executáveis nativos para o processador ARM, os quais são empacotados como aplicações para iOS ou Android.
4. Titanium (Appcelerator, 2013) é um *framework* de desenvolvimento *open source*. Permite a criação de aplicações nativas em diferentes dispositivos móveis e sistemas operacionais,

¹ No artigo de Dio-Synodinos (2012) é chamado de Mono iOS/Android.

incluindo iOS, Android, Windows e BlackBerry, bem como híbridos e HTML5. Ele inclui um SDK open source com mais de 5.000 APIs para sistemas operacionais de dispositivos móveis.

5. O Adobe® AIR® 3 (Adobe®, 2013) disponibiliza uma *runtime* que permite que desenvolvedores usem HTML, JavaScript, Adobe Flash® software, e ActionScript® para criar aplicações web que funcionam como aplicativos clientes independentes, sem as limitações de um navegador.
6. Qt (Qt Project, 2013), pronunciado como "cute" é um *framework* para desenvolvimento de aplicações para plataformas embarcadas, desktop e *mobile*. Utiliza as tecnologias C++ ou QML, CSS e JavaScript. Permite o desenvolvimento de aplicações web uma vez e implantá-los em desktop, sistemas operacionais móveis e embarcados sem reescrever o código fonte. Permite também a criação de aplicações nativas de alto desempenho, bem como híbridas(nativa e web) possibilitando ao desenvolvedor escolher qual abordagem fornece a melhor experiência ao usuário.
7. RhoMobile (Motorola, 2013a) é uma suíte de ferramentas, integrada por RhoConnect, RhoStudio e RhoElements (Motorola, 2013b). Os aplicativos RhoMobile são independentes do sistema operacional como: Windows® Embedded Handheld, Windows® CE, Windows® Phone 7, Apple® iOS, Android® e BlackBerry®. Como parte da Suíte RhoMobile, RhoElements é uma estrutura de desenvolvimento de aplicações HTML5 que suporta os sistemas operacionais móveis mais utilizados como: Windows® Embedded Handheld, Windows® CE, Windows® Phone 7, Apple® iOS, Android® e BlackBerry. (Motorola, 2013b; Motorola, 2013a)
8. Marmalade (2013) é um *framework* que permite aos desenvolvedores atingir mais plataformas e mais dispositivos com uma única base de código, poupando tempo e esforço de desenvolvimento e deixando o desenvolvedor focado no seu problema de negócio. Através dele pode-se optar por desenvolver aplicações nativas utilizando C++, híbrida com HTML5 e nativo, ou ainda utilizar Lua. Aborda os sistemas: iOS, Android, Windows Phone 8, BlackBerry 10, Windows e Mac, bem como Smart TVs e plataformas de *streaming* de TV.
9. jQuery Mobile (jQuery Foundation, 2013a) é um *framework* de interface de usuário baseada no jQuery que funciona nos *smartphones*, *tablets*, *e-readers*, e plataformas

desktop. Construído com acessibilidade e acesso universal em mente, segue melhoria progressiva e princípios de Web Design Responsivo(WDR).

10. Corona (Corona Labs Inc., 2013) é um *framework* que tem como objetivo permitir com que se desenvolva o código uma vez e possa implantá-lo nos principais SOs como: iOS, Android, Kindle Fire e NOOK. Utiliza a linguagem LUA e C++.
11. MoSync (MoSync AB, 2014) é um *framework* para desenvolvimento de aplicativos nativos para várias plataformas móveis, utilizando uma única base de código. Um aplicativo desenvolvido através deste *framework* pode ser implantado em nove plataformas diferentes ao mesmo tempo. Utiliza C/C++ ou HTML5/JavaScript como linguagem de programação, ou uma combinação de ambos para criar aplicativos híbridos.

Uma pesquisa voltada à comunidade de desenvolvedores classificou os *frameworks* citados, por ordem decrescente de importância e adoção, respectivamente: PhoneGap(Apache Cordova), jQuery Mobile, Sencha Touch/jQTouch, Appcelerator, RhoMobile, Adobe Air(Flex/Flash), Mono iOS/Android, jQTouch, Corona, Qt, Marmelade e Mosync (Dio-Synodinos, 2012).

Ohrt and Turau (2012), destacaram o Flash Builder, Illumination Software Creation, LiveCode, Marmalade, MoSync, PhoneGap, RhoStudio e Titanium.

A Tabela 1 aponta as características dos principais *frameworks*. A escolha dos *frameworks* mostrados na Tabela 1 levou em consideração os mais utilizados. Deve-se ter em mente que os mais usados não necessariamente são os melhores. O gráfico da Figura 3 mostra como a maioria dos *frameworks* ainda são desconhecidos pela comunidade de desenvolvedores de aplicativos. O gráfico é resultado do estudo (© research2guidance, 2013), que contemplou 90 *frameworks*. O presente trabalho contribui com a comunidade de desenvolvedores ao se testar e analisar os *frameworks*, afim de que o desenvolvedor faça uma boa escolha, na hora de eleger um para suprir suas necessidades, o que proporcionará aos usuários experiências cada vez mais ricas e eficazes, além de fazer com que o processo de desenvolvimento, sob o ponto de vista do desenvolvedor, seja mais produtivo e com menos retrabalhos.

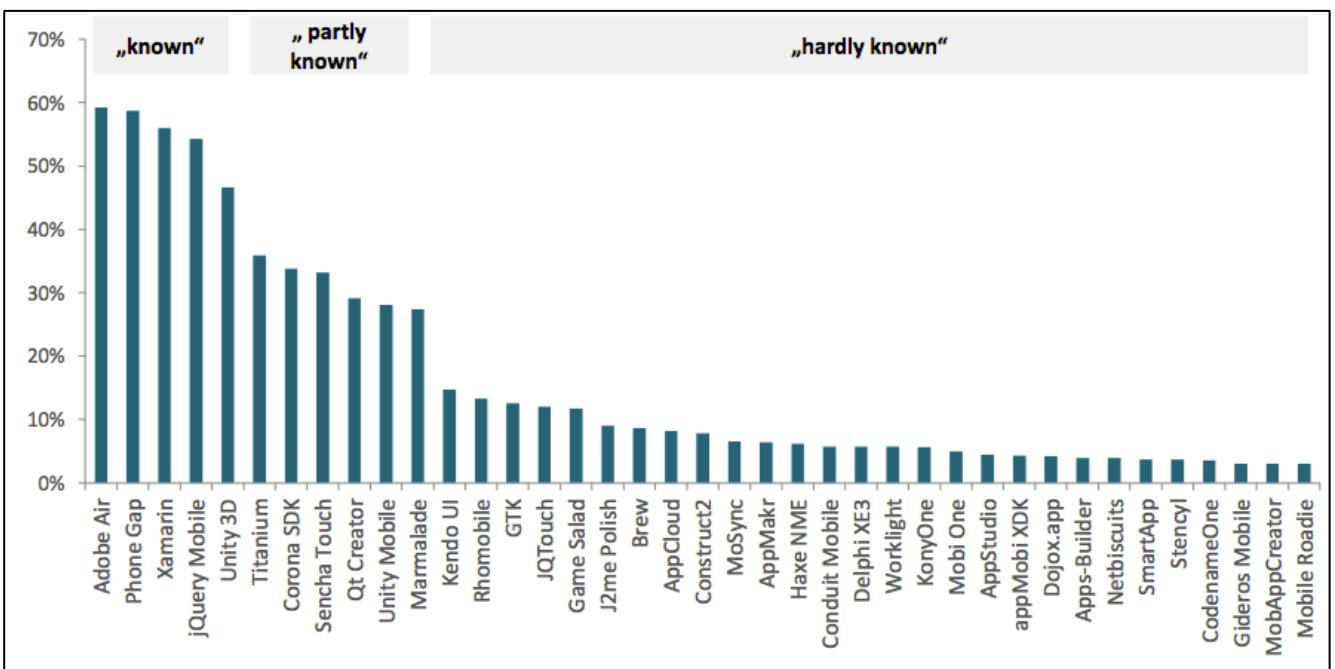


Figura 3. Nível de conhecimento dos *frameworks* multiplataforma

Fonte: Adaptado de research2guidance (2013)

Tabela 1. Principais características dos Frameworks Multiplataforma Móvel

Framework	SO Móvel Alvo	Licença	Grátis	Open Source	Linguagem	Extensível *
PhoneGap	iOS, Android, Windows Phone, BlackBerry, Symbian, WebOS, bada	MIT	sim	sim	HTML5, Javascript	sim
Apache Flex	iOS, Android	Apache License v. 2.0	sim	sim	ActionScript, HTML5, C++	sim
Sencha Touch	iOS, Android, Windows Phone, BlackBerry	GPLv3, comercial	sim	sim	HTML5, Javascript	sim
Titanium	iOS, Android, BlackBerry	Apache License v. 2.0	sim	sim	HTML5, Javascript	-
jQuery Mobile	iOS, Android, Windows Phone, BlackBerry, Symbian, WebOS, bada, MeeGo	MIT	sim	sim	HTML5, Javascript	não
DragonRad	iOS, Android, Windows Phone, BlackBerry	própria	não	não	D And D	não
MoSync	iOS, Android, Windows Phone, BlackBerry, Symbian, Moblin, MeeGo, Java	GPL v2 , Comercial	sim	sim	HTML5, Javascript, C/C++	sim
MoSync Reload	iOS, Android, Windows Phone	GPL v2 , Comercial	sim	sim	HTML5, Javascript	não
Marmalade	iOS, Android, BlackBerry, Windows Phone	community, comercial	não	parcial	HTML5, Javascript, Lua, C++	-
Canappi	iOS, Symbian, Android(Beta)	-	sim	sim	mdsl	somente a versão paga (enterprise)
Rhodes	iOS, Android, Windows Phone, BlackBerry	MIT, comercial	sim	sim	HTML, Javascript, Ruby	sim
Qt	Windows Phone, Symbian, MeeGo (Nokia N9)	GPL v3, LGPL v2 e comercial	sim	sim	C++, JavaScript	sim
Xamarain	iOS, Android, Windows Phone	comercial	não	sim	C#	sim

* Extensível à SDK Nativa

Os dados levantados na Tabela 1 serviram para se ter uma visão geral de cada *framework* e para mostrar suas diferenças. São elas:

1. Os sistemas operacionais móveis contemplados;
2. Tipo de licença a qual são distribuídos para avaliar os termos e condições de uso;
3. As linguagens de programação oferecidas aos desenvolvedores para criar aplicações;

4. Viabilidade de extensão da API do *framework* para possibilitar a implementação de algum recurso nativo não contemplado pelo *framework*;
5. O sistema operacional que o desenvolvedor poderá utilizar para executar o *framework*;
6. A estrutura do aplicativo; e
7. A estrutura dos elementos da interface gráfica.

Sistema operacional alvo: é uma das principais características que o desenvolvedor deve procurar em um FMM. De acordo com a Tabela 1, pode-se observar que a maioria dos *frameworks* contemplam os sistemas iOS e Android. A decisão de deixar de atender determinados SOs pode acarretar na perda de potenciais consumidores.

Tipo de licença: é importante para o desenvolvedor saber se as aplicações geradas podem ser comercializadas sem custo adicional, se os *frameworks* são gratuitos ou proprietários, se são *open source*, podendo ser modificado para que supra uma necessidade que o desenvolvedor necessite e o *framework* ainda não ofereça. Outra característica importante das soluções *open source* é a vantagem de ter uma vasta comunidade de desenvolvedores que as utilizam, gerando discussões em fóruns, apontamento de *bugs*, geração de *plugins* para disponibilizar novas funcionalidades, entre outras. O tipo de licença aponta também os possíveis custos para consultar o suporte técnico.

A linguagem de desenvolvimento: a tabela permite observar uma tendência em se utilizar as linguagens HTML5 e JavaScript combinadas com CSS. Se examinar-se todos os FMM disponíveis, pode ser que chegue-se a conclusão de que não é uma tendência, mas sim o motivo desses serem os FMM mais utilizados hoje. A utilização de tecnologias já amadurecidas e mais conhecidas entre os desenvolvedores diminui a curva de aprendizado. Por isso, a linguagem de programação foi aqui considerada uma característica dos FMM que merece ser analisada no processo de escolha do que será mais produtivo para cada caso.

Outra característica que a Tabela 1 proporciona inferir é que geralmente, os *frameworks* com menor número de linguagens, consegue abranger maior número de SOs móveis. Utilizando como exemplo o PhoneGap e o jQuery Mobile que utilizam JavaScript e HTML, pode-se observar que eles contemplam 7 e 8 SO móveis. Enquanto o extremo oposto pode ser visto com os *frameworks* Titanium e o Apache Flex que contempla somente os sistemas iOS e o Android. Seria preciso um estudo direcionado a essas observações para saber se a interpretação está correta.

Nas ferramentas, as quais o presente trabalho estudou em mais detalhe, pode-se concluir que a observação tende a ser verídica, pois os *frameworks* mais “simples” como PhoneGap e o jQuery Mobile conseguem atingir mais plataformas, porém contemplam recursos e performance dos *smartphones* bem mais limitados que o Titanium que consegue atingir mais recursos, com melhor desempenho. Apesar de ser intuitivo pensar que neste caso o Titanium é melhor, deve-se ter em mente que não há um melhor absoluto, mas sim, o melhor para cada caso. Por exemplo, se não se vai utilizar os recursos adicionais que o Titanium oferece, não seria melhor utilizar os outros e disponibilizar seu aplicativo em muito mais plataformas? As vantagens e desvantagens de cada um desses três *frameworks* serão apontadas no Capítulo Descrição dos Experimentos.

Extensibilidade da API: saber se a API é extensível à SDK nativa, permite ao desenvolvedor saber que se ele precisar acessar um recurso não contemplado pelo *framework*, se ele pode utilizar a SDK nativa para completar seu aplicativo.

A estrutura do aplicativo e da interface gráfica:

Segundo Ohrt e Turau (2012) para melhor entendimento do funcionamento dos FMM, três aspectos dos aplicativos devem ser observados:

A maneira como são instalados: quando se trata de sistemas operacionais móveis, na maioria das vezes, os provedores dos sistemas só disponibilizam aplicativos para o respectivo SO através de uma loja de aplicativos como a Play Store da Google para o sistema Android e a App Store da Apple para iOS. Elas possuem seus próprios critérios para avaliar se o aplicativo será publicado ou rejeitado.

Desenvolvedores que tem pretensão de distribuir para várias plataformas precisarão conhecer as regras de cada uma. Porém, os FMM já fazem esse trabalho, disponibilizando o aplicativo nas diversas plataformas seguindo suas respectivas regras de aprovação. Por esse motivo, os FMM evitam disponibilizar algumas tecnologias ou recursos que possam barrar o aplicativo.

Sabendo disso, Ohrt e Turau (2012), afirmaram que uma boa maneira de se evitar a rejeição de um aplicativo é utilizar a SDK disponibilizada pelo provedor do seu respectivo SO móvel. A estrutura do aplicativo difere dependendo da SDK nativa, consequentemente, a estrutura do aplicativo deve ser conhecida pelo desenvolvedor. No presente trabalho, será utilizado o termo abordagem da aplicação, para diferenciar as estruturas. São elas: nativa, *web-app*, *web-enabled*, interpretada. Alguns *frameworks* se classificam como híbridos, pois podem utilizar uma combinação das abordagens citadas.

O aplicativo pode ser denominado, sob o ponto de vista de sua estrutura, como integrado ou não integrado. O primeiro forma o conjunto dos aplicativos nativos e interpretados. O segundo contem os chamados *web-apps*, que se caracterizam por serem páginas web contendo funcionalidades de aplicativos. São chamados de *web-apps* e possuem restrição ao acesso dos recursos do dispositivo. Pode-se antecipar aqui um exemplo do que será visto no decorrer deste trabalho. Por exemplo: o *framework* jQuery tem uma abordagem de um *web-app*, enquanto a combinação dele com o jQuery Mobile transforma o aplicativo em *web-enabled*. Indo mais adiante, o PhoneGap encapsula esse aplicativo *web-enabled*, possibilitando que ele se comporte como um aplicativo híbrido.

Os aplicativos nativos são aqueles cujo código gerado se integra ao SO e consequentemente podem acessar diretamente a API do sistema. Podem ainda acessá-lo através de uma biblioteca, provendo um nível de abstração. Já os interpretados se caracterizam por precisar de uma máquina virtual(MV) para acessar a API do sistema. A MV pode vir empacotada com o aplicativo ou pode ser provida como um aplicativo instalado separadamente. No presente trabalho se o aplicativo é interpretado por uma máquina virtual ou por um interpretador, ele é tratado como interpretado.

Sob o ponto de vista da estrutura dos elementos da interface gráfica, os aplicativos podem ser classificados como nativos, *web-based* ou customizados. Alguns exemplos de elementos gráficos são os botões, *labels* e listas. Os aplicativos nativos utilizam os mesmos elementos usados pelo sistema operacional e são suportados pela SDK deste. Já os denominados *web-based* contam com o navegador para fornecer os componentes. Neste caso, o aplicativo incorpora uma *web-view* que processa código HTML e apresenta elementos de navegadores que podem ser modificados usando *Cascading Style Sheets* (CSS). Os customizados criam uma interface customizada que o próprio aplicativo desenha e controla.

Por motivo de simplificação os *frameworks* serão aqui tratados como mostrado no Quadro 1.

Abordagem	Definição
Web	São páginas web acessadas através de um navegador e adaptáveis para a formatação da tela e interação do usuário seguindo as expectativas oferecidas pelo dispositivo móvel. Se sobressaem pelo número de plataformas de alcance, por não precisar fazer download do aplicativo e updates instantâneos para todos usuários. Necessitam de conexão com a internet.
Híbrida	Trata-se de aplicativos que geralmente são escritos utilizando HTML, CSS e JavaScript rodando com um renderizador nativo ao invés do navegador. Têm acesso limitado ao hardware do dispositivo. Podem ser distribuídos nas App Stores e geralmente não precisam de uma conexão com a internet.
Nativa	Roda diretamente no dispositivo e tem acesso às suas funções de <i>hardware</i> . Gera código nativo, ou seja, o mesmo código gerado pelo <i>framework</i> disponibilizado pelo provedor do SO. Geralmente sobressaem em performance e experiência do usuário igual da plataforma utilizada. Adicionalmente, podem ser executados sem uma conexão com a internet.

Quadro 1. Abordagens dos frameworks

Fonte: Appcelerator Inc. (2014)

Dentre os *frameworks* levantados na Tabela 1, escolheu-se três para implementação do protótipo. Para tal escolha, foram considerados os que geram aplicações para os principais sistemas operacionais como Android e iOS. Outro fator considerado, foi a escolha de *frameworks* que utilizam abordagens distintas. No caso, PhoneGap para abordagem Híbrida, Titanium para abordagem Nativa e jQuery Mobile para abordagem Web.

Tabela 2. Classificação dos *frameworks* segundo seu tipo de abordagem

Framework	Abordagem Web	Abordagem Híbrida	Abordagem Nativa
jQuery Mobile	x		
PhoneGap		x	
Titanium			x

2.5 PHONEGAP

Nesta sessão será apresentada uma visão geral desse *framework*, bem como suas principais características.

2.5.1 Visão Geral

PhoneGap (Phonegap, 2013b) é um *framework open source* para desenvolvimento de aplicativos multiplataforma móveis. A empresa Adobe, ao perceber a complexidade de desenvolver

aplicações para as diversas plataformas criou um *framework* que gera aplicações baseadas em padrões já conhecidos de tecnologias web como HTML, CSS e JavaScript. Uma das principais funções do PhoneGap é web-habilitar as funcionalidades nativas de dispositivos se utilizando de padrões abertos e padronizados, como HTML, CSS e JavaScript.

2.5.2 Características

Para gerar aplicações o PhoneGap se utiliza das linguagens HTML5 e JavaScript, aliadas ao CSS3 que é responsável pela definição dos estilos da interface gráfica.

Uma aplicação gerada pelo PhoneGap é uma aplicação para a plataforma web empacotada com os mesmos recursos utilizados pelo navegador para renderizá-las. Ele possibilita que aplicações web sejam disponibilizadas como aplicativos capazes de rodar como se fossem nativos. Dessa forma, o desenvolvedor pode submeter seu aplicativo às *App Stores* para serem distribuídos e monetizados.

Ele conta com uma grande comunidade de desenvolvedores, o que significa vasta diversidade de aplicativos de demonstração, tutoriais, discussões em fóruns, muitos desenvolvedores reportando bugs, entre outras vantagens.

Segundo a empresa, o *framework* já teve 1 milhão de downloads e é usado por 400 mil desenvolvedores (PhoneGap, 2013a). Segundo a mesma fonte, o que atrai o alto número de desenvolvedores é o fato do *framework* ser totalmente gratuito, tanto para uso pessoal como comercial, além de possuir código fonte aberto. Ele é distribuído sobre a licença Apache versão 2.0, uma vez que a Adobe doou o código para a Fundação Apache.

Tabela 3. APIs disponíveis pelo *framework* PhoneGap para cada plataforma

	iPhone 3GS +	Android	Blackberry OS 5.x	WebOS	Windows Phone 7	Symbian	Bada
Accelerometer	sim	sim	sim	sim	sim	sim	sim
Câmera	sim	sim	sim	sim	sim	sim	sim
Bússola	sim	sim	não	sim	sim	sim	não
Contatos	sim	sim	sim	não	sim	sim	sim
Arquivos	sim	sim	sim	sim	sim	sim	sim
Geolocalização	sim	sim	sim	sim	sim	sim	sim
Mídia	sim	sim	não	não	sim	não	não
Internet	sim	sim	sim	sim	sim	sim	sim
Alerta	sim	sim	sim	sim	sim	sim	sim
Som	sim	sim	sim	sim	sim	sim	sim
Vibração	sim	sim	sim	sim	sim	sim	sim
Memória	sim	sim	sim	sim	sim	sim	não

Fonte: Adaptado de PhoneGap Features (2013b)

O código do PhoneGap foi entregue para a Apache Software Foundation (ASF), sob o nome Apache Cordova e passou para o status de projeto de alto nível em outubro de 2012. Através da ASF, o desenvolvimento futuro do PhoneGap irá garantir administração aberta do projeto. Ele permanecerá sempre livre e de código aberto sob a licença Apache, versão 2.0.

Os usuários desse *framework* contam com um serviço na nuvem chamado PhoneGap Build que lhes permite compilar suas aplicações sem as SDKs, compiladores e hardware nativos dos respectivos sistemas operacionais móveis. Sem esta ferramenta, o time de desenvolvimento precisaria de computadores Mac para compilar aplicativos para iOS e PCs com as SDKs do Android, BlackBerry, Java e outras.

“PhoneGap é uma tentativa de criar um ambiente de execução abstrato para permitir o desenvolvimento multiplataforma” (Whinnery, 2012).

2.6 TITANIUM

Nesta sessão será apresentada uma visão geral desse *framework*, bem como suas principais características.

2.6.1 Visão Geral

Titanium (Appcelerator, 2013) é um *framework* de desenvolvimento aberto(*open source*) e extensível para a criação de aplicações nativas em diferentes dispositivos móveis e sistemas operacionais, incluindo iOS, Android, Windows Mobile (não é Windows Phone) e BlackBerry OS (não é a versão utilizada nos *smartphones* da BB), bem como híbridos e HTML5. Ele inclui um SDK *open source* com mais de 5.000 APIs para sistemas operacionais de dispositivos móveis, Studio, uma poderosa IDE baseada no Eclipse, Alloy, um *framework* MVC e Cloud Services para um backend *mobile* pronto para uso (Sencha, 2013).

2.6.2 Características

O *framework* gera código nativo através da tradução de código JavaScript. Em seguida, utiliza técnicas para gerar o pacote final como:

1. Pré-compilador: otimiza o código JavaScript reduzindo espaços em branco, tamanho das variáveis, entre outras.
2. *Front-end* do compilador: gera o código nativo específico da plataforma apropriada.
3. Empacotador: utiliza o compilador de cada plataforma específica e gera o executável.

2.7 JQUERY MOBILE

2.7 Nesta sessão será apresentada uma visão geral desse *framework*, bem como suas principais características.

2.7

2.7.1 Visão Geral

jQuery Mobile (jQuery Foundation, 2013a) é um *framework* de interface de usuário baseada no jQuery que funciona nos *smartphones*, *tablets*, *e-readers*, e plataformas desktop. Construído com acessibilidade e acesso universal em mente, segue melhoria progressiva e princípios de Web Design Responsivo(WDR).

2.7 Web Design Responsivo (jQuery Foundation, 2013b) é uma abordagem de design e técnicas que visam adaptar o layout e interação de um site ou aplicativo para funcionar de forma ideal em uma **2.7**

ampla gama de resoluções de dispositivos, densidades de tela e modos de interação com a mesma base de código. O *framework* tem uma série de widgets responsivos: grades responsivas, tabelas de refluxo e tabelas seletoras de colunas, e painéis deslizantes.

2.7.2 Características

É compatível com as plataformas iOS, Android, Windows Phone, BlackBerry, Symbian, WebOS, bada, MeeGo.

A ênfase do jQuery Mobile em marcação semântica e prática de melhoria progressiva torna-o fácil de usar. Prática de melhoria progressiva (jQuery Foundation, 2013b) significa sempre começar com HTML semântico que irá funcionar em qualquer dispositivo, em seguida, inicia-se discretamente camadas em CSS avançado, enquanto JS é iniciado apenas em navegadores compatíveis.

Tudo isso fornece uma base sólida para as otimizações de acordo com o dispositivo através da técnica de RWD. Basta conhecer HTML básico, para começar a construir sites móveis. Além disso, a Fundação jQuery oferece várias bibliotecas para simplificar a programação JavaScript e o design do aplicativo baseado em HTML5 e CSS, oferecendo compatibilidade *cross-browser*.

O jQuery Mobile focado em interfaces móveis, trabalha junto com o jQuery para atingir mais funcionalidades JavaScript e Ajax e com o jQuery UI para criação de interfaces de usuário e o *framework* jQUnit para testes JavaScript.

3 TRABALHOS RELACIONADOS

Trabalhos semelhantes foram buscados em artigos publicados nos mais importantes repositórios de periódicos. Na IEEE pode-se encontrar o trabalho intitulado de Cross-Platform Development Tools for Smartphone Applications (Jones, 2012) com informações semelhantes ao presente trabalho. Já na base de dados SpringLink, encontrou-se um ebook, o qual um dos capítulos tratava do assunto em questão, Developing Mobile Apps Using Cross-Platform Frameworks: A Case Study (Humayoun, Ehrhart, & Ebert, 2013).

A pesquisa se deu através da Internet, por *sites* de procura como, Google, Yahoo e Lycos, além de bases de dados de periódicos, e-books, revistas eletrônicas de portais assinados pela Universidade

como Institute of Electrical and Electronic Engineers (IEEE), EUA, Institution of Engineering and Technology (IET), Inglaterra, EBSCO HOST, Ebrary, etc.

A pesquisa iniciou com a busca de artigos científicos que referenciem os problemas e os desafios encontrados hoje para desenvolvimento multiplataforma. Posteriormente estendeu-se a quaisquer *sites*, em português e inglês, que tratem do assunto. A análise das informações encontradas levou em conta a procedência das mesmas, sua quantidade, relevância, características desejáveis e limitações relacionadas ao problema.

3.1 CROSS-PLATFORM DEVELOPMENT TOOLS FOR SMARTPHONE APPLICATIONS

Pesquisadores da Universidade de Tecnologia de Hamburg (Ohrt & Turau, 2012), na Alemanha, compararam *frameworks* para desenvolvimento multiplataforma móvel. Os autores supracitados concluíram que tais *frameworks* satisfazem as expectativas dos usuários e diminuem o trabalho para desenvolver aplicativos para variadas plataformas móveis, porém necessitam melhorias.

Tal conclusão foi alcançada por meio de um experimento que consistiu na criação e análise de um aplicativo simples utilizando nove *frameworks* para desenvolvimento de aplicativos para múltiplas plataformas móveis. O aplicativo consistiu em uma tela com um ícone e um título.

O estudo realizado para criar o experimento, foi direcionado no sentido de testar características e funcionalidades dos *frameworks* que favorecem os desenvolvedores ao facilitar o desenvolvimento de aplicativos para diferentes plataformas. Além disso, pretendia-se também, avaliar algumas características de performance como: o tempo de instalação, o tempo de inicialização, a quantidade de memória utilizada para persistência de dados e finalmente o consumo de memória RAM.

Os autores justificaram as métricas escolhidas para avaliação do experimento como uma tentativa de avaliar o aplicativo tanto do lado do usuário final, quanto dos desenvolvedores. Segundo Ohrt e Volker (2012), a avaliação do desempenho é importante para satisfazer as expectativas do usuário. Já as características e funcionalidades oferecidas pelos *frameworks* são importantes para diminuir o custo de desenvolvimento do aplicativo como: a linguagem de programação, a capacidade do *framework* compilar sem a necessidade da SDK nativa do sistema operacional, a funcionalidade de completar o código, a oferta ou não de uma ferramenta de *design* de interface gráfica, de um depurador

de código, de um emulador, de suporte ao recurso de *bluetooth*, e finalmente, a capacidade ou não de se utilizar código nativo para suprir algum recurso necessário e não disponibilizado pelo *framework*.

3.2 DEVELOPING MOBILE APPS USING CROSS-PLATFORM FRAMEWORKS: A CASE STUDY

Em outro trabalho semelhante (Humayoun, Ehrhart, & Ebert, 2013) aplicativos foram desenvolvidos em três cenários que utilizavam os sistemas Android e iOS. Utilizou-se *frameworks* de desenvolvimento nativos e três *frameworks* de desenvolvimento multiplataforma como Appcelerator, RhoMobile Rhodes e MoSync. Um estudo de avaliação do usuário foi realizado em um ambiente controlado, o qual o foco era a verificação da resposta de interação de diferentes versões de três cenários distintos, utilizando diferentes dispositivos e sistemas operacionais.

A principal diferença entre este e o de Orth e Volker (2012) está na implementação de aplicativos móveis interativos e na forma de avaliação do desempenho. A interatividade do aplicativo proporcionou avaliar a percepção de usuários eleitos para experimentar e julgar o tempo de resposta de interação do aplicativo e também medir o grau de satisfação obtido do mesmo aplicativo desenvolvido em diferentes *frameworks* e executando em sistemas operacionais distintos.

Para realizar o experimento, três aplicativos diferentes foram desenvolvidos com o intuito de abranger os diversos recursos disponíveis nos *smartphones*. O primeiro, chamado MovePic, tinha o intuito de testar o suporte e processamento de gestos multitoque, como *tap*, *drag* e *pinch*.

O segundo, BubbleLevel, foi desenvolvido com intuito de testar o suporte e processamento do acelerômetro. Este simulou uma ferramenta de “nível de bolha” (do inglês “*bubble level*”), utilizada para medir se um objeto é paralelo ou perpendicular ao chão.

Finalmente, o terceiro, AnnotatePic, utilizou recursos como acesso a câmera, ao sistema de arquivo e serviços específicos do telefone. Basicamente o aplicativo carrega uma imagem da galeria de fotos da câmera, adiciona informações como nome, descrição e data, e salva a imagem com as informações na memória.

O objetivo foi analisar se usuários de diferentes níveis de experiência com dispositivos móveis sentiriam alguma diferença se o mesmo aplicativo fosse lhes apresentado, sendo ele construído através de diferentes estruturas e para diferentes plataformas.

Os autores do referido trabalho concluíram que onde o rápido tempo de resposta não é tão importante, a melhor opção é usar o desenvolvimento multiplataforma, pois economiza-se tempo e custo para quase o mesmo nível de satisfação do usuário. O mesmo acontece em aplicativos onde o tempo de resposta da interação é importante, porém não muito crítico. Caso contrário, os *frameworks* nativos alcançam melhores resultados. De modo geral, concluiu-se que as aplicações móveis desenvolvidas utilizando as estruturas multiplataforma fornecem aproximadamente o mesmo nível de satisfação do usuário como as desenvolvidas nos ambientes nativos.

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

3.3

4 DESENVOLVIMENTO

O presente trabalho, afim de testar os frameworks, desenvolveu um estudo voltado ao levantamento e à análise das características dos principais *frameworks* para desenvolvimento de aplicativos multiplataforma móveis. Após essa análise, escolheu-se três *frameworks* para implementação de um protótipo que contemple as principais funcionalidades de um *smartphone*.

Os *frameworks* escolhidos para implementar o protótipo foram jQuery Mobile, PhoneGap e Titanium. Para tal escolha, foram considerados: os *frameworks* que geram aplicações para os principais sistema operacionais como Android e iOS e *frameworks* que utilizam abordagens distintas. As abordagens podem ser do ponto de vista da estrutura do aplicativo: *web-app*, nativa ou interpretada ou do ponto de vista da interface gráfica: nativa, web-enabled, customizada. Esses conceitos estão explicados com mais detalhes no Capítulo 2 (Fundamentação Teórica).

Uma visão geral dos *frameworks* escolhidos pode ser descrita como:

PhoneGap (Phonegap, 2013b) é um *framework open source* para desenvolvimento de aplicativos multiplataforma móveis. A empresa Adobe, ao perceber a complexidade de desenvolver aplicações para as diversas plataformas criou um *framework* que gera aplicações baseadas em tecnologias web padronizadas. Dessa forma, qualquer dispositivo capaz de rodar um navegador web suporta suas aplicações. A principal função do PhoneGap é ligar as tecnologias Web aos recursos nativos dos dispositivos.

Titanium (Appcelerator, 2013) é um ambiente de desenvolvimento aberto e extensível para a criação de aplicações nativas em diferentes dispositivos móveis e sistemas operacionais, incluindo iOS, Android, Windows e BlackBerry, bem como híbridos e HTML5. Ele inclui um SDK *open source* com mais de 5.000 APIs para sistemas operacionais de dispositivos móveis, Studio, uma poderosa IDE baseada no Eclipse, Alloy, um *framework* MVC e *Cloud Services* para um *backend mobile* pronto para uso (Sencha, 2013).

O jQuery Mobile (jQuery Foundation, 2013a) é um *framework* de interface de usuário baseada no jQuery que funciona nos *smartphones*, *tablets*, *e-readers*, dentre outros. Construído com acessibilidade e acesso universal em mente, segue melhoria progressiva e princípios de Web Design Responsivo (WDR).

Web Design Responsivo (jQuery Foundation, 2013b) é uma abordagem de design e técnicas que visam adaptar o layout e interação de um site ou aplicativo para funcionar de forma ideal em uma ampla gama de resoluções de dispositivos, densidades de tela e modos de interação com a mesma base de código. O *framework* tem uma série de *widgets* responsivos e sensíveis ao toque: grades responsivas, tabelas de refluxo e tabelas seletoras de colunas, e painéis deslizantes.

4.1 VISÃO GERAL DO SISTEMA

Na Figura 4, utilizou-se um diagrama de atividades como artifício para ilustrar uma visão geral do sistema. É imprescindível ressaltar que não se trata de um diagrama de atividades.

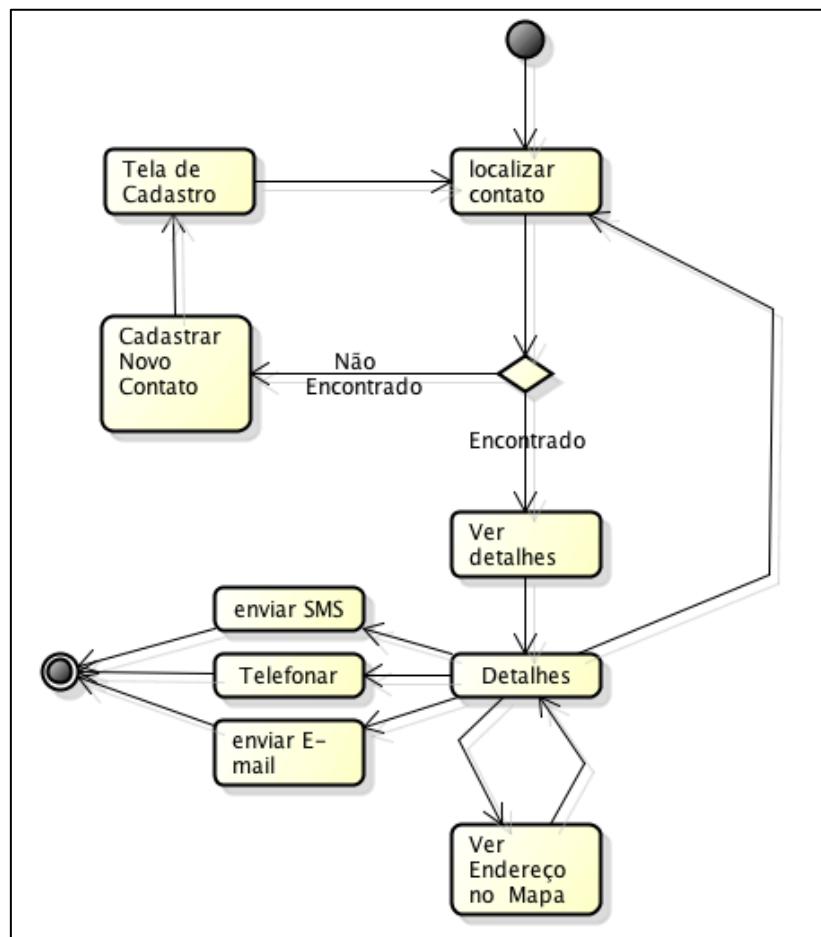


Figura 4. Visão Geral do Sistema

O aplicativo protótipo será uma agenda telefônica, pois considerou-se que é possível, com o desenvolvimento dessa, abranger os principais recursos utilizados nos *smartphones* como:

Utilização do recurso câmera fotográfica, utilização da persistência de dados, tanto local como remoto, o recurso da geolocalização, manipulação de listas, envio de SMS, de e-mails, realização ligações, utilização de touch para acessar os elementos, renderização adequada nos diversos dispositivos móveis.

No Apêndice A é possível ver as telas do aplicativo para maior entendimento.

4.2 MODELAGEM DO SISTEMA

Nesta sessão será apresentada uma visão geral do aplicativo, bem como seus casos de uso.

4.2.1 Diagrama de Casos de Uso

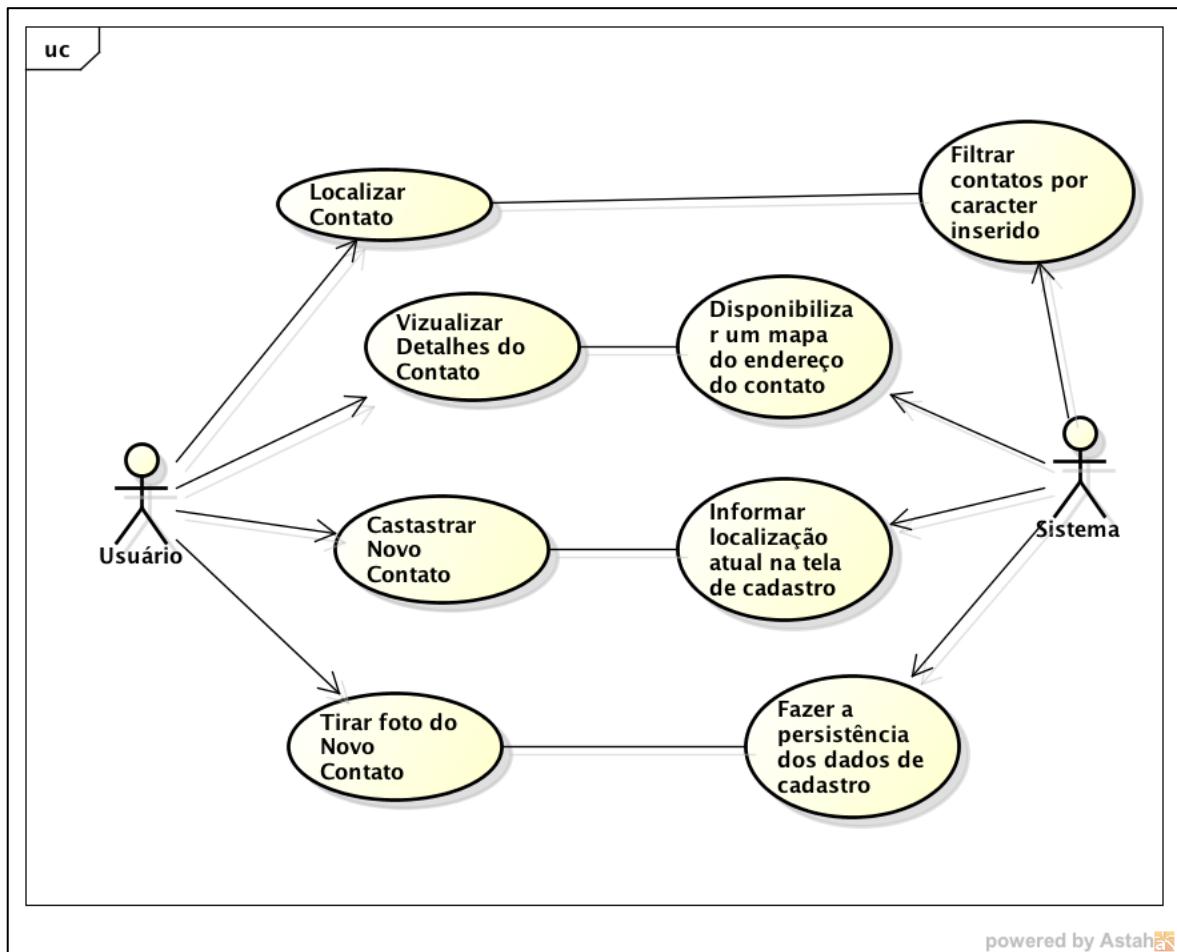


Figura 5. Diagrama de Casos de Uso do Aplicativo

4.2.2 Descrição dos Casos de Uso

Do Ponto de vista do usuário:

Localizar contato: utiliza recursos de persistência de dados, ao passo que a lista de contatos pode estar, dependendo da disponibilidade do *framework*, armazenada localmente, em um webservice, ou no servidor local. Além disso, utiliza recursos para filtrar a busca mostrando apenas os nomes que contêm os caracteres inseridos.

Visualizar detalhes do contato existente: redireciona o usuário para a tela com todos os detalhes do usuário. Nessa tela, pode-se clicar na foto para vê-la em tamanho real, clicar no campo telefone para que seja feita a ligação, assim como no campo e-mail, SMS. Ainda o sistema oferece que o endereço possa ser mostrado no mapa, em diferentes tamanhos de *zoom*.

Cadastrar novo contato: A tela de cadastro de novo contato, oferece a possibilidade do usuário tirar uma foto ou escolher dentre seus arquivos. Além disso, é possível obter através do recurso de geolocalização do *smartphone*, a localização atual para auto preenchimento dos campos. Aqui principalmente, a persistência dos dados será utilizada.

Tirar foto para colocar no cadastro: a possibilidade de tirar uma foto na hora para colocar no cadastro, utiliza mais um dos recursos mais comuns de um *smartphone*, captura e tratamento da imagem.

Do Ponto de vista do sistema:

Filtrar e mostrar contatos ao inserir parte do nome: é um recurso importante, e muito utilizado, para bom aproveitamento do espaço da tela. Só serão apresentados nomes já cadastrados. Caso não esteja cadastrado, o sistema oferece a barra de navegação com a opção de adicionar usuário.

Informar localização atual na tela de cadastro: informa a localização atual, como uma funcionalidade extra que pode ser usado, no caso do cadastro estar sendo feito no escritório do contato por exemplo. Neste caso, o GPS irá converter as coordenadas no endereço, preenchendo a maior parte dos campos, restando apenas alguns refinamentos para que o usuário informe como número do prédio, por exemplo.

Disponibilizar um mapa do endereço do contato: o sistema oferece a função de mostrar o endereço em forma de mapa, renderizável em diferentes tamanhos de *zoom*. Funcionalidades adicionais podem ser acrescentadas aí com funções como, localizar por perto, ou direções daqui até outro lugar.

Fazer a persistência dos dados de cadastro: o sistema deve ser capaz de guardar os dados do cadastro de forma que possa recuperá-los sempre que o usuário necessitar acessá-los.

4.3 DETALHAMENTO DO DESENVOLVIMENTO

O desenvolvimento se deu em três etapas, iniciando da ferramenta mais simples para a mais complexa. Todo processo de desenvolvimento seguiu a documentação do provedor do *framework*. A sequência de *frameworks* utilizados foi jQuery Mobile, PhoneGap e Titanium.

jQuery Mobile:

Para desenvolver para jQuery Mobile, deve-se ter um conhecimento prévio em jQuery que por sua vez recomenda como pré-requisito uma boa base em JavaScript, HTML e CSS3.

Sendo assim, como o autor do presente trabalho nunca havia trabalhado com essas tecnologias começou um estudo em JavaScript no site da W3Schools (W3Schools, 2014).

Uma breve descrição da interação dos elementos acima descritos e estudados poderá ajudar o leitor que também não esteja tão familiarizado com essas tecnologias.

O navegador envia uma requisição de uma página web ao servidor web que devolve a página HTML. Essa é carregada utilizando uma estrutura criada pelos navegadores denominada *Document Object Model* (DOM). Esse modelo segue o formato de árvore e organiza os elementos HTML em forma de nodos que se relacionam.

JavaScript proporciona aos desenvolvedores uma linguagem que interage com os elementos do DOM, ou seja, com JavaScript pode-se facilmente encontrar os elementos HTML do documento/DOM. Em poucas palavras, o HTML é carregado no DOM pode interagir com JavaScript. Porém, cada navegador vem com uma interface de DOM ligeiramente diferente. Os principais navegadores são: Mozilla Firefox, Internet Explorer, Google Chrome, Safari e Opera.

Uma das principais funções do jQuery, que implementa sua própria versão de JavaScript, é cuidar para que sua implementação contorne essas pequenas diferenças de cada navegador e seu código tenha o resultado esperado nos principais navegadores.

O jQuery Mobile é um *framework* de interface de usuário, por esse motivo, ele geralmente aparece juntamente com o jQuery. Ele basicamente fornece uma biblioteca de componentes personalizados para dispositivos móveis.

A persistência dos dados pode ser feita do lado cliente ou do lado do servidor. O HTML5 nos oferece dois tipos de armazenamento de dados no lado do cliente: o armazenamento permanente ou o de sessão. O primeiro guarda os dados mesmo se o dispositivo for desligado ou reiniciado. O segundo armazena os dados enquanto a sessão do aplicativo está aberta. Ambas podem coexistir em contextos diferentes. No caso do aplicativo Agenda, foi utilizada o armazenamento permanente, uma vez que os contatos devem ser mantidos se o aplicativo for fechado. Esse tipo de armazenamento é recomendado para aplicativos simples, pois a base de dados do lado cliente tem um limite máximo recomendado de 5 megabytes (W3C, 2014).

PhoneGap:

Sabendo que a principal função do PhoneGap é ligar as tecnologias Web aos recursos nativos dos dispositivos, utilizou-se o PhoneGap neste aplicativo basicamente para empacotá-lo para que possa ser instalado no dispositivo e rodar como um aplicativo nativo. O PhoneGap possibilita que ele seja submetido e comercializado nas App Stores dos principais SOs. Neste trabalho, o aplicativo não será publicado, pois para fazê-lo é preciso ter uma licença de desenvolvedor. Essa licença custa atualmente \$99 dólares anuais, o que se tornou um fator limitante para tal experiência.

Mesmo para testar o aplicativo no aparelho é preciso a licença de desenvolvedor. Neste caso, usou-se a licença gratuita concedida pelo *iOS Developer University Program* (Apple, 2014). Essa licença permite que se rode 1 aplicativo por vez no dispositivo cadastrado e é concedida à comunidade acadêmica. A licença é um fator imprescindível para se testar aplicativos nos sistemas da Apple. No caso do Android, mais flexível, pode-se desenvolver e instalar quantos aplicativos quiser, porém, caso queira distribuí-los em sua respectiva loja de aplicativos, gratuitamente ou não, deve-se obrigatoriamente comprar uma licença de desenvolvedor, o que garantirá a procedência do aplicativo. Neste caso, paga-se uma taxa também.

Outra importante característica do PhoneGap é tornar mais recursos dos dispositivos móveis acessíveis. Devido à simplicidade da nossa aplicação, previamente caracterizada para utilizar os principais recursos dos *smartphones*, não foi preciso utilizar nenhum recurso adicional do PhoneGap.

Neste trabalho, ele foi usado para transformar nosso aplicativo web, em um aplicativo executável e com características mais próximas de um aplicativo nativo.

Para se criar um aplicativo através desse *framework*, é necessário o SDK nativo. Isso significaria que não se poderia criar um aplicativo para iOS sem que se tenha uma máquina da Apple rodando o SO da Apple. Porém esse *framework* possui um serviço chamado PhoneGap Build (Adobe PhoneGap Build, 2014) que compila o aplicativo na nuvem e disponibiliza o executável para as diversas plataformas. Este serviço foi utilizado para gerar o executável do protótipo. Ele é gratuito para se utilizar com um aplicativo apenas por vez. Caso queira-se instalar mais de um aplicativo no *smartphone* utilizando esse serviço, deve-se pagar uma assinatura.

4.4 DESCRIÇÃO DOS EXPERIMENTOS

Esta sessão descreverá as dificuldades encontradas durante o desenvolvimento dos aplicativos em cada um dos três *frameworks*, assim como suas vantagens e desvantagens. Os FMM serão analisados na sequência em que foram implementados, ou seja, jQuery Mobile, PhoneGap e Titanium.

jQuery Mobile:

A primeira dificuldade encontrada durante a implementação dos experimentos foi devido a uma característica até então desconhecida do jQuery Mobile. Para criação da interface gráfica utilizou-se uma ferramenta indicada na documentação do *framework* em questão denominada Codiqa (Codiqa, 2014), onde cada página representava uma tela do aplicativo. Ao inserir um formulário era dado um id a ele, para que pudesse manipulá-lo via JavaScript. Porém, a ferramenta não tratou de uma característica do jQuery Mobile chamada “*single-page navigation model*” que permite que várias “páginas” diferentes estejam presentes no documento (ou DOM) ao mesmo tempo. Dessa forma o aplicativo final era gerado inteiramente em uma página que renderizava ou não certa parte de acordo com a regra estabelecida. Portanto, deve ser ter atenção para que cada formulário tenha um “id” único, não só na página em que está, mas único dentre todos os formulários do aplicativo.

Outra dificuldade encontrada foi a documentação fraca do jQuery Mobile. Ela é baseada em exemplos, ou “*demos*” para seguir a linguagem da página. Não há na documentação um guia que lhe conduza desde os primeiros passos. Ou seja, a documentação oficial é um ponto negativo do *framework*.

Apesar da documentação oficial ser um pouco confusa e pobre de informações, por ser gratuito e de código fonte aberto, possui uma grande comunidade que disponibiliza tutoriais e materiais de consulta. Segundo relatório apresentado pela empresa research2guidance (2013, p. 17) é o segundo *framework* mais utilizado por desenvolvedores voltados para o desenvolvimento de aplicativos para multiplataforma móvel.

Outro ponto positivo é a simplicidade e facilidade de aprendizado que o *framework* proporciona. Pode-se ainda citar como vantagem:

- a possibilidade de se poder testar e depurar o aplicativo direto no navegador;
- a quantidade de dispositivos que ele é capaz de atingir, basta que tenha um navegador; e
- não precisa ser instalado no dispositivo o que lhe proporciona acesso imediato ao aplicativo, pois não precisa passar pela inspeção das *app stores*, além de facilitar a manutenção, uma vez que não é preciso que o usuário atualize o aplicativo para ver as mudanças das novas versões.

Uma característica que depende do contexto para ser classificada como vantagem ou desvantagem é a propriedade do aplicativo gerado pelo jQuery Mobile aparecer da mesma forma em todos os dispositivos. Diferente de *frameworks* que geram aplicativos nativos, caso se queira que os estes sigam as cores da empresa ou de uma marca, pode ser considerado uma vantagem. Adicionalmente, o jQuery Mobile conta com uma ferramenta de customização de interface gráfica chamada ThemeRoller for jQuery Mobile (2014) que tem exatamente este propósito. Sob o ponto de vista de um aplicativo comum, um estudo concluiu que os usuários têm preferência pelo visual nativo de seu dispositivo (HUMAYOUN, EHRHART & EBERT, 2013).

No Apêndice B, pode-se ver o aplicativo rodando nas duas plataformas.

PhoneGap:

O PhoneGap cumpriu sua função como deveria. Utilizou-se para compilar um serviço na nuvem oferecido pelo provedor do *framework*, chamado PhoneGap Build (2014). O PhoneGap permite que se compile um aplicativo web para rodar como nativo. Basicamente ele empacota a *engine* do navegador junto com o aplicativo de modo que este não precise daquele para interpretar o código web que foi utilizado na geração do aplicativo.

Ele precisa da SDK nativa, que no caso, estava no servidor, na nuvem. O *framework* estrutura o aplicativo de modo que ele atenda as especificações exigidas pelo provedor do SO da respectiva plataforma, para que o aplicativo possa rodar no dispositivo alvo. O iPhone por exemplo, não permite a instalação de aplicativos sem uma assinatura do desenvolvedor, adquirida através da solicitação de uma licença chamada *Provisioning Profile*.

O *framework* ainda verifica se o aplicativo segue as exigências necessárias para que seja disponibilizado na *App Store*. No Capítulo 2, essa questão é tratada na descrição da estrutura da aplicação. A única maneira de se instalar um aplicativo na maioria das plataformas é através de suas respectivas lojas de aplicativos. Os *frameworks* têm um papel importante nesse quesito, pois possibilitam que o desenvolvedor se concentre no desenvolvimento do seu aplicativo, e o libera de conhecer as exigências para publicação de seu aplicativo, principalmente quando a intenção é disponibilizá-lo para múltiplas plataformas, cada uma com suas regras próprias. Dessa forma, ganha-se em produtividade e agilidade na publicação.

Insta ressaltar que não foi testado se o aplicativo gerado pelo PhoneGap foi aprovado na hora de publicar na *App Store*, pois foge do contexto deste trabalho. Porém, foi possível testar o comportamento do aplicativo no celular, “privilegio” dado ao desenvolvedor, para não contradizer o que disse antes, que das duas plataformas contempladas neste trabalho, a única maneira de se instalar um aplicativo no iPhone é através de sua loja. No caso do Samsung (Android) é permitida a instalação de aplicativos não assinados, mas esses aplicativos não podem ser disponibilizados na Play Store ou Samsung App Store.

Não foi necessária a utilização dos recursos adicionais desse *framework*, pois o aplicativo rodou como esperado. Esses recursos adicionais, permitem que se acesse recursos dos *smartphones* que um *framework* de aplicativos web não consegue acessar como acelerômetro, por exemplo.

Só se precisou acessar a documentação do PhoneGap para saber como configurar o arquivo “config.xml” onde deve-se informar o nome e descrição do aplicativo, a versão do PhoneGap, entre outras informações básicas. Para aplicativos mais complexos existem várias informações que devem ser preenchidas ali.

No Apêndice B, pode-se ver o aplicativo rodando nas duas plataformas. Pode-se observar que uma diferença visível é que o aplicativo gerado pelo jQuery Mobile está claramente rodando dentro de um navegador, pois aparece a barra de navegação. Apesar de ser uma diferença pequena, o usuário não

se sente utilizando um aplicativo, mas sim um site com recursos adicionais. Além disso, nesse caso ele precisa estar conectado à internet e o desempenho do aplicativo web ser comprometido pela qualidade da conexão.

Titanium:

Esse *framework* foi utilizado para desenvolver um aplicativo nativo. Ele possui uma SDK própria que precisou ser instalada. Neste caso precisou-se seguir as instruções da documentação para não se deparar com problemas de incompatibilidade de versões, pré-requisitos para instalação, configuração das variáveis de ambiente. Toda essa preparação do ambiente de desenvolvimento tomou um tempo considerável.

Utilizou-se para desenvolver o aplicativo a IDE oferecida pela empresa responsável pelo *framework*. Essa IDE se chama Studio e foi desenvolvida em JAVA o que aumentou a lista de pré-requisitos para começar o desenvolvimento.

A documentação do *framework* é excelente, o que facilitou muito o processo de configuração do ambiente e desenvolvimento do aplicativo. Ela possui uma estrutura bastante organizada e intuitiva, onde se pode encontrar a API, vídeo tutoriais e guias que lhe conduzem passo a passo no processo de conhecimento do ambiente, das ferramentas auxiliares e do *framework*.

Uma vantagem de se utilizar o Titanium é que ele possui todo esse suporte de qualidade e é totalmente gratuito tanto para uso pessoal, quanto para uso comercial.

Para geração de aplicativos multiplataforma, o *framework* utiliza a linguagem JavaScript. Posteriormente todo código gerado pela linguagem será convertido em código fonte nativo da plataforma como Objective C para iPhone e Java para Android.

Um ponto negativo do *framework* é que ele gera código nativo somente para iOS e Android.

Um ponto positivo é que através dele, pode-se atingir todos os recursos do *smartphone*, segundo a documentação. Este trabalhou abordou apenas algumas funcionalidades muito utilizadas pelos usuários.

4.5 RESULTADOS

Nesta sessão será apresentado o resultado dos testes, bem como sua análise, o que irá proporcionar apontar o comportamento do aplicativo gerado o que permitirá avaliar os *frameworks* quanto às suas vantagens e desvantagens, pontos positivos e negativos, entre outros.

4.5.1 Análise das Funcionalidades

A fim de avaliar os resultados, será feita uma análise das funcionalidade dos aplicativos nas 2 plataformas: iPhone e Samsung. A configuração do ambiente onde o aplicativo foi testado é mostrada no Quadro 2, enquanto o resultado dos testes está listado na Tabela 4.

Plataforma	iPhone	Samsung
Modelo	4s	Galaxy S3
Versão do Sistema Operacional	iOS 7.1.1	Android 4.3

Quadro 2. Dados dos dispositivos onde foram feitos os testes

Tabela 4. Análise de Funcionalidades

funcionalidades	jQuery Mobile 1.3.1		PhoneGap 3.3.0		Titanium 3.2.3.GA	
	iPhone	Samsung	iPhone	Samsung	iPhone	Samsung*
Persistência dos Dados	sim	sim	sim	sim	sim	-
Utilização do recurso de escolher imagem	sim	não	sim	sim	sim	-
Utilização do recurso de geolocalização	sim	sim	sim	sim	sim	-
Integração com aplicativos pré-instalados	não	não	sim	sim	sim	-
Efetuou ligação	sim	sim	sim	sim	sim	-
Enviou SMS	sim	sim	sim	sim	sim	-
Enviou e-mail	sim	sim	sim	sim	sim	-
Renderizou como esperado	não	não	sim	sim	sim	-

* No momento em que este trabalho foi entregue, por causa de um problema ainda não identificado, não foi possível testar o aplicativo no aparelho.

4.5.2 Análise do Resultado da Tabela

Nesta sessão serão reportados os resultados que não funcionaram como esperado, ou seja, somente os que receberam “não” na Tabela 4.

Funcionalidade: Utilização do recurso de escolher imagem

O recurso de escolha da imagem não funcionou como esperado com o aplicativo feito utilizando o *framework* jQuery Mobile. Esperava-se que ao clicar no botão de escolher arquivo (Figura 6), o aplicativo oferecesse a opção de tirar uma foto ou pegar da galeria. Porém, essa funcionalidade respondeu de diferentes maneiras em diferentes navegadores. O Quadro 3 descreve o comportamento obtido em cada navegador.

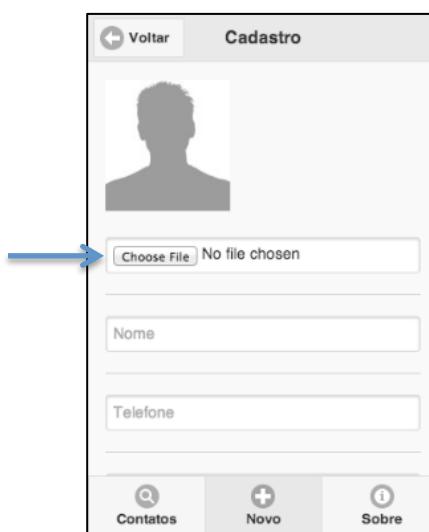


Figura 6. Tela de cadastro desenvolvida no jQuery Mobile

No iPhone funcionou como esperado nos navegadores Safari e Google Chrome. Porém, no Android, respondeu de maneira diferente em cada navegador.

Navegador	Comportamento
Google Chrome	Entrou direto na câmera fotográfica, sem dar a opção de selecionar foto na galeria.
Firefox	Funcionou como esperado e ofereceu recursos adicionais como escolher o arquivo de aplicativos de armazenamento na nuvem como dropbox e afins.
Opera Mini	Ofereceu somente o recurso de escolher da galeria.

Quadro 3. Comportamento anômalo do recurso de escolher imagem no Android

Funcionalidade: Integração com aplicativos pré-instalados

No iPhone, ao acessar aplicativos pré-instalados como e-mail, sms e discador, o aplicativo era deixado em segundo plano ao acessar outro aplicativo. Mas ao voltar, o aplicativo era recarregado e

voltava para a tela inicial. Enquanto no Android, bastava clicar no “botão” de voltar (Figura 7) que se poderia continuar utilizando o aplicativo do ponto em que parou.



Figura 7. Botão de voltar do aparelho Samsung

Fonte: Google Imagens (2014)

Funcionalidade: Renderizou como esperado

As Figuras 8 e 9 mostram o mesmo aplicativo gerado pelo jQuery Mobile nos navegadores Opera Mini e Google Chrome, respectivamente. Pode se observar a diferença entre o resultado esperado que é o da Figura 9 e o resultado obtido na Figura 8, somando mais um problema no aplicativo gerado pelo jQuery Mobile tanto no iOS quanto no Android.

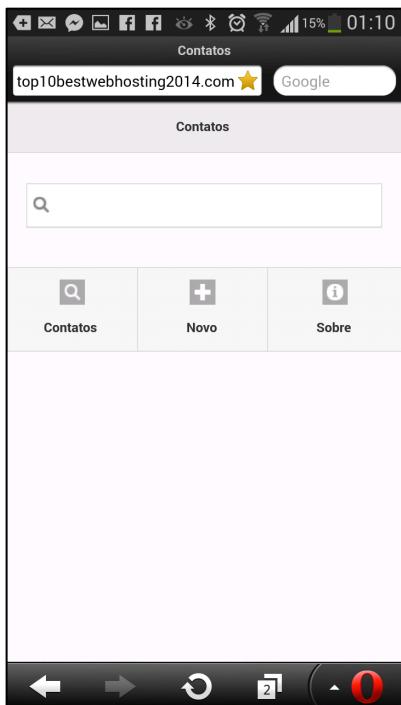


Figura 8. Aplicativo no Opera Mini

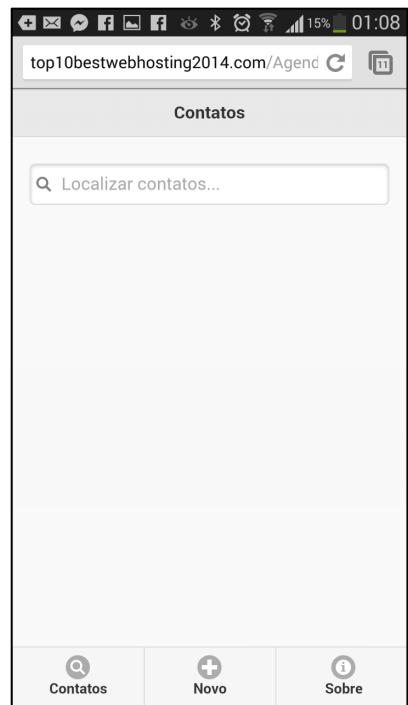


Figura 9. Aplicativo no Google Chrome

Outro problema encontrado ao rodar o aplicativo no Opera Mini é a falta de suporte às requisições utilizando Ajax.

A Figura 10 apresenta o aplicativo renderizado de maneira incorreta, ao distorcer a imagem da foto. Observe que no navegador Chrome e no aplicativo gerado pelo PhoneGap, a imagem aparece da maneira correta, enquanto no navegador nativo do Android ela aparece distorcida.

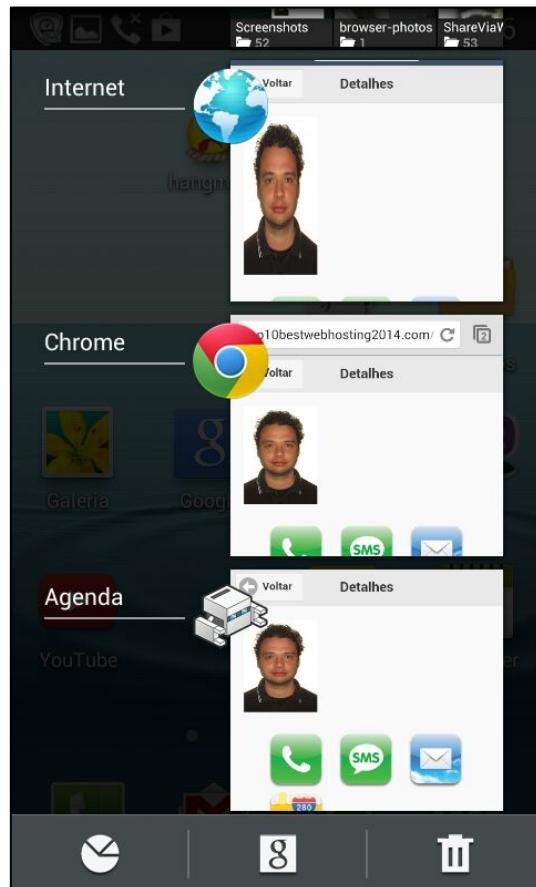


Figura 10. Printscrean do comportamento anômalo do aplicativo gerado pelo jQuery Mobile

A Tabela 5, apresenta um *printscrean* de uma das telas do aplicativo rodando no iPhone para que se possa observar as diferenças entre aplicativos não nativos e nativos. Existem várias diferenças como: desempenho, usabilidade, integração, entre outras, porém a Tabela 5 tem o objetivo de ilustrar como o aplicativo se comporta do ponto de vista do usuário. O aplicativo nativo aproveita a familiaridade do usuário com o sistema de seu dispositivo, para tornar o aplicativo mais intuitivo e melhorar a experiência do usuário.

O aplicativo apresentado na Tabela 5 foi criado por um desenvolvedor acostumado com a interface do iOS, por isso, não há muita diferença entre as telas. Mas um usuário do sistema Android pode achar o aplicativo pouco intuitivo.

Uma diferença que fica muito evidente é que no iOS existe o botão de voltar, enquanto no Android para voltar ele conta com um botão no aparelho (veja Figura 7).

Outra diferença são os ícones e as cores do aplicativo que são típicos de casa SO.

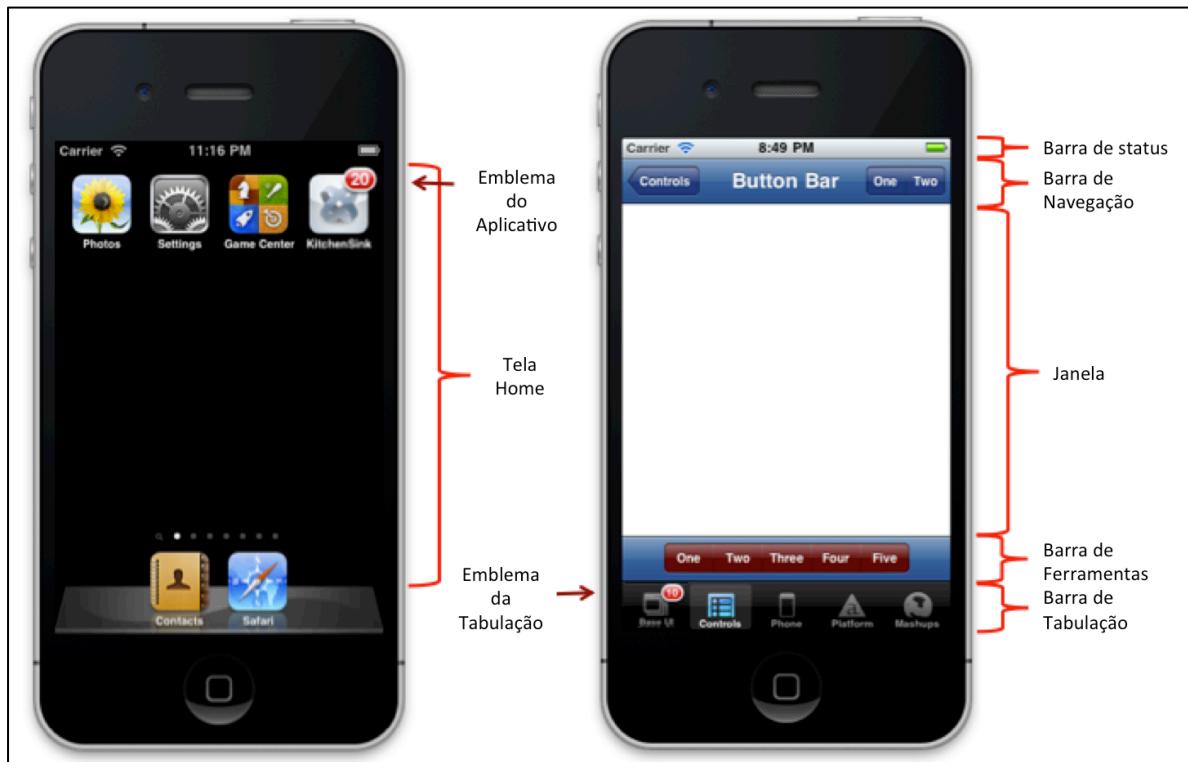


Figura 11. Visão Geral do iOS

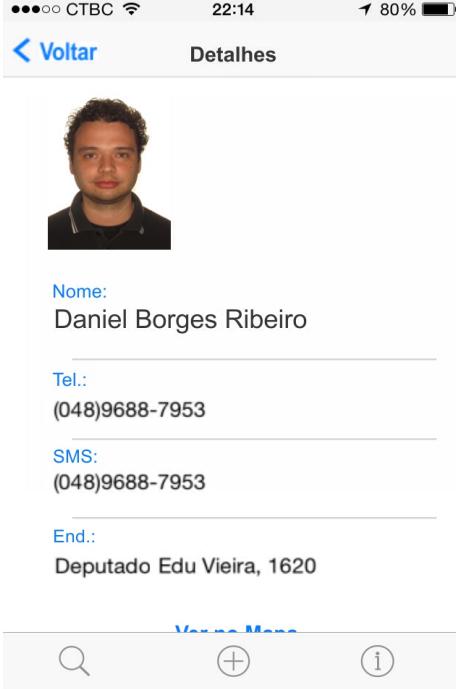
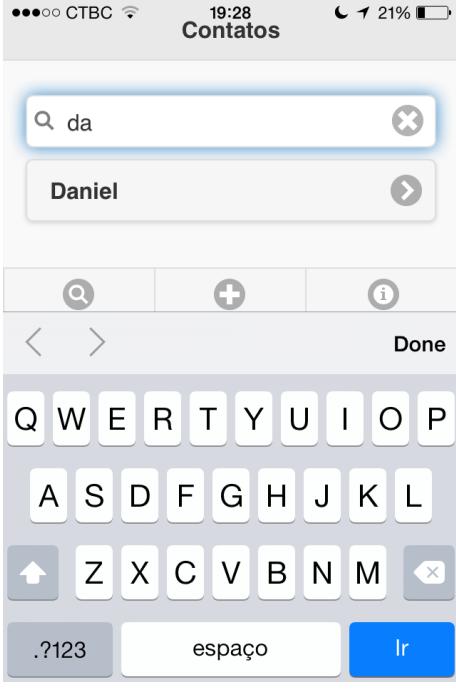
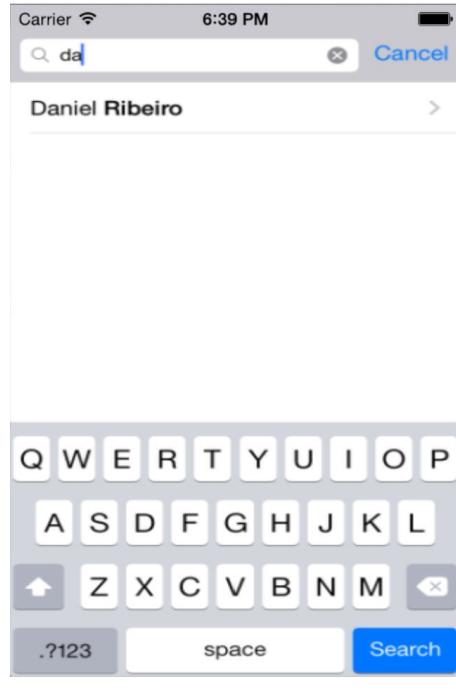
Fonte: Appcelerator (2014)



Figura 12. Visão Geral do Android OS

Fonte: Appcelerator (2014)

Tabela 5. Visual Não-Nativo x Nativo

Aparência do Aplicativo Não Nativo	Aparência do Aplicativo Nativo
 <p>Daniel Borges Ribeiro</p> <p>Tel: (048)9688-7953</p> <p>SMS: (048)9688-7953</p> <p>danielbribeiro@gmail.com</p> <p>Rua Deputado Edu Vieira, 1620</p> <p>Ver no Mapa</p> <p>Contatos Novo Sobre</p>	 <p>Nome: Daniel Borges Ribeiro</p> <p>Tel.: (048)9688-7953</p> <p>SMS: (048)9688-7953</p> <p>End.: Deputado Edu Vieira, 1620</p> <p>Ver no Mapa</p> <p>Search Novo Sobre</p>
 <p>Contatos</p> <p>da</p> <p>Daniel</p> <p>Done</p> <p>Q W E R T Y U I O P A S D F G H J K L Z X C V B N M</p> <p>.?123 espaço Ir</p>	 <p>Carrier 6:39 PM</p> <p>da Cancel</p> <p>Daniel Ribeiro</p> <p>Q W E R T Y U I O P A S D F G H J K L Z X C V B N M</p> <p>.?123 space Search</p>

4.6 CONSIDERAÇÕES

Esta sessão tem o objetivo de compilar os resultados e análise dos experimentos de modo sucinto e representá-los em uma tabela e em seguida apresentar um *score board* para eleger a melhor solução para o contexto do experimento do presente trabalho.

4.6.1 jQuery Mobile

O jQuery Mobile é um *framework* multiplataforma móvel que gera aplicativos Web. Nesta sessão serão relacionadas as características dos aplicativos gerados por ele com os resultados obtidos nos experimentos. Adicionalmente, serão apresentadas suas vantagens e desvantagens.

Característica 1 de um aplicativo Web

- Atinge o maior número de dispositivos, pois qualquer dispositivo que tenha um navegador é capaz de rodar o aplicativo.

Vantagens e desvantagens:

- Vantagem: atinge maior número de usuários.
- Desvantagem: necessita estar conectado à internet e a performance pode ser prejudicada pela qualidade da conexão.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: O aplicativo deve renderizar da maneira como foi configurado.
- Resultado Obtido: O aplicativo apresentou resultados diferentes dependendo do navegador. Como pode ser visto na Figura 10, a imagem apresentou distorção em suas dimensões no navegador padrão do Android. O teste foi efetuado nos navegadores Google Chrome, Firefox, Opera Mini e no navegador padrão do Android e do iOS (Safari). Pode-se observar na Figura 8 que a barra de tabulação aparece no centro da página, enquanto foi configurada para aparecer no rodapé da janela.
- Resultado Esperado: Na tela de buscar os contatos ilustrada nas Figuras 8 e 9, é feita uma filtragem a cada letra digitada, refinando a lista de contatos somente com as respectivas letras. A atualização da lista é feita utilizando a tecnologia Ajax.

- Resultado Obtido: No navegador Opera Mini as requisições via Ajax não funcionaram, fazendo com que a cada letra digitada, para se ter o resultado esperado, foi preciso enviar uma requisição (*request*) HTML para receber uma resposta (*response*) que reconstrói todos os elementos da página ao invés de atualizar apenas aquele campo, perdendo toda a vantagem que a tecnologia Ajax trás para o desempenho das aplicações Web.

Característica 2 de um aplicativo Web

- O aplicativo não precisa ser instalado no dispositivo.

Vantagens e desvantagens:

- Vantagens:
 1. Não consome memória do dispositivo, o que permite atingir os *smartphones* mais simples e com limitações de memória de armazenamento.
 2. As atualizações são feitas no servidor e aparecem automaticamente para todos os usuários.
- Desvantagens:
 1. O espaço de armazenamento local é limitado a 5Mb conforme definição da especificação do HTML5.
 2. O aplicativo não pode ser distribuído e monetizado nas App Stores.
 3. Devido a restrições de segurança o acesso aos recursos de hardware é bastante limitado.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado esperado: acessar o recurso da câmera ou o recurso de acesso aos arquivos do sistema para coletar uma imagem para atribuir ao contato na tela de cadastro ilustrada na Figura 6.
- Resultado obtido: no iOS, utilizando os navegadores disponíveis (Google Chrome e Safari), ao clicar no botão para escolher a imagem as opções “tirar uma foto” e “escolher da galeria” apareceram corretamente conforme esperado. Porém no Android a resposta foi diferente para cada navegador testado (Google Chrome, Firefox, Opera Mini, Navegador

Nativo). Em alguns casos entrava direto na câmera, não dando a opção de escolher da galeria, em outros casos só dava a opção de escolher da galeria.

Navegador	Resultado obtido
Google Chrome	Entrou direto na câmera. Não ofereceu a opção para escolher na galeria de fotos.
Opera Mini	Entrou direto no sistema de arquivos. Não ofereceu a opção de tirar foto.
Firefox	Funcionou além do esperado. Ofereceu acesso à camera, a galeria de fotos e a outros aplicativos instalados como Dropbox, Samsung Link, Google Drive, Fotos e OneDrive.
Navegador Padrão denominado “Internet”	Entrou direto na câmera. Não ofereceu a opção para escolher na galeria de fotos.

Quadro 4. Descrição do resultado obtido nos diferentes navegadores

Característica 3 de um aplicativo Web

- O acesso aos recursos de hardware é bastante limitado.

Vantagens e desvantagens:

- Vantagem: não há vantagem.
- Desvantagem: a característica em si já é uma desvantagem.

No contexto do experimento (Resultado Experado x Resultado Obtido):

- Resultado esperado: era esperado que o aplicativo pudesse acessar os contatos já disponíveis no dispositivo.
- Resultado obtido: devido a restrições de segurança o acesso a esse recurso não é disponibilizado pelo *framework*.

Outras vantagens:

1. Como não utiliza a SDK nativa e não precisa ter licença de desenvolvedor não é preciso ter um computador com sistema operacional da Apple para disponibilizar o aplicativo. O que não se aplica aos demais *frameworks*. Isso implica numa economia significativa em relação aos pré-requisitos para desenvolvimento.

2. Não precisa de licença de desenvolvedor. Para desenvolver aplicativos para iOS é preciso adquirir uma licença de desenvolvedor no valor de \$99 anuais. Caso contrário, não será possível sequer testar o aplicativo no aparelho. A alternativa seria utilizar o simulador que possui limitações em relação ao dispositivo real.

Outras desvantagens:

1. Documentação baseada em exemplos. Pouco intuitiva e desorganizada.

4.6.2 PhoneGap

O PhoneGap é um *framework* multiplataforma móvel que gera aplicativos Híbridos. Nesta sessão serão relacionadas as características dos aplicativos gerados por ele com os resultados obtidos nos experimentos. Adicionalmente, serão apresentadas suas vantagens e desvantagens.

Característica 1 de um aplicativo híbrido

- Trata-se de aplicativos que geralmente são escritos utilizando HTML, CSS e JavaScript rodando com um renderizador nativo ao invés do navegador.

Vantagens e desvantagens:

- Vantagem: permite com que desenvolvedores web aproveitem seu conhecimento prévio e produzam aplicativos móveis com uma curva menor de aprendizado.
- Desvantagem: o aplicativo se comporta como nativo, porém com visual web. Essa característica não aproveita a familiaridade do usuário com a posição dos componentes do seu dispositivo, podendo prejudicar a experiência do usuário em relação ao aplicativo.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: um aplicativo com características bem próximas a um aplicativo nativo.

Resultado Obtido: conforme ilustrado na Tabela 5 o aplicativo gerado ficou muito parecido com um aplicativo nativo do sistema iOS, pois o desenvolvedor está familiarizado com este sistema. Provavelmente um usuário do sistema Android não teria uma adaptação tão rápida ao aplicativo, pois está acostumado com outro design, o qual não existe por exemplo um

botão voltar na barra de navegação, pois os dispositivos que utilizam Android já vêm com um botão voltar no próprio aparelho. As Figuras 11 e 12 mostram com mais detalhes as diferenças entre o design dos dois sistemas.

Característica 2 de um aplicativo híbrido

- Têm acesso limitado ao hardware do dispositivo.

Vantagens e desvantagens:

- Vantagem: não se aplica.
- Desvantagem: o aplicativo não tira todo proveito de todas as funcionalidades dos dispositivos.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: para o aplicativo idealizado no presente trabalho, esse *framework* satisfez todas as funcionalidades requeridas.
- Resultado Obtido: o resultado obtido foi o resultado esperado.

Característica 3 de um aplicativo híbrido

- Podem ser distribuídos nas App Stores.

Vantagens e desvantagens:

- Vantagem: pode ser monetizado, pode ser instalado por todos os usuários do sistema. Abstrai do desenvolvedor a responsabilidade de conhecer as regras de cada App Store dos diversos sistemas, possibilitando com que ele se foque no desenvolvimento do aplicativo e consiga disponibilizá-lo para as diversas plataformas com um risco menor de rejeição e de maneira mais rápida e competitiva.
- Desvantagem: o desenvolvedor precisa possuir uma licença de desenvolvedor que possui um custo relativamente elevado. No caso do iOS essa licença custa \$99 por ano. Além disso, o *framework* para gerar um aplicativo com as características necessárias para distribuí-lo nas App Stores precisa utilizar o SDK nativo que no caso do iOS só é disponibilizado em sistemas operacionais da Apple, o que implica que o desenvolvedor seja

obrigado a possuir um computador da Apple. Isso implica em um investimento adicional de no mínimo \$599 que é o preço do computador mais barato da Apple, o Mac mini.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: Não fez parte do escopo deste trabalho a disponibilização do aplicativo nas App Stores.
- Resultado Obtido: não se aplica.

Característica 4 de um aplicativo híbrido

- Geralmente não precisam de uma conexão com a internet.

Vantagens e desvantagens:

- Vantagem: aumenta a mobilidade do aplicativo.
- Desvantagem: não há desvantagem nessa característica.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: O aplicativo deve rodar independente de uma conexão com a internet.
- Resultado Obtido: o resultado obtido foi o resultado esperado.

4.6.3 Titanium

O Titanium é um *framework* multiplataforma móvel que gera aplicativos Nativos. Nesta sessão serão relacionadas as características dos aplicativos gerados por ele com os resultados obtidos nos experimentos. Adicionalmente, serão apresentadas suas vantagens e desvantagens.

Característica 1 de um aplicativo nativo

- Roda diretamente no dispositivo e tem acesso as suas funções de hardware.

Vantagens e desvantagens:

- Vantagem: o aplicativo terá acesso a quase todos recursos de hardware, podendo proporcionar uma experiência mais rica ao usuário, uma vez que pode-se combinar os recursos do dispositivo para enriquecer as funcionalidades do aplicativo.
- Desvantagem: aumenta a complexidade do desenvolvimento do *framework*, fazendo com que disponibilize o aplicativo para um número bem menor de plataformas alvo que os demais *frameworks*.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: possibilitar a interação do aplicativo com os contatos pré existentes do dispositivo, utilizar a memória do *smartphone*, entre outros.
- Resultado Obtido: os resultados esperados foram atingidos, porém somente no iPhone, pois devido às adequações ao código que devem ser feitas para cada sistema, o prazo de desenvolvimento acabou antes que a implementação para o Samsung fosse finalizada.

Característica 2 de um aplicativo nativo

- Gera código nativo, ou seja, o mesmo código gerado pelo *framework* disponibilizado pelo provedor do SO.

A análise do código gerado não faz parte do escopo deste trabalho.

Característica 3 de um aplicativo nativo

- Geralmente sobressaem em performance e experiência do usuário igual da plataforma utilizada.

Vantagens e desvantagens:

- Vantagem: aproveita a familiaridade que o usuário tem com o design do software e hardware do seu dispositivo para proporcionar uma experiência mais intuitiva ao usuário, ou seja, ele não precisará aprender como usar as funções do aplicativo, pois serão muito parecidas com as que ele já utiliza nos outros aplicativos.
- Desvantagem: pode ser que o desenvolvedor fique atrelado ao *framework*.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: gerar o aplicativo com os componentes nativos do sistema.
- Resultado Obtido: o resultado esperado foi alcançado conforme ilustrado na Tabela 5.

Característica 4 de um aplicativo nativo

- Podem ser executados sem uma conexão com a internet.

Vantagens e desvantagens:

- Vantagem: aumenta a mobilidade do aplicativo.
- Desvantagem: não há desvantagem nessa característica.

No contexto do experimento (Resultado Esperado x Resultado Obtido):

- Resultado Esperado: O aplicativo deve rodar independente de uma conexão com a internet.
- Resultado Obtido: o resultado obtido foi o resultado esperado.

4.7 A ESCOLHA DA MELHOR SOLUÇÃO

Como já foi explicado anteriormente, a escolha da melhor solução depende do contexto do desenvolvimento. Portanto, será descrito nesta sessão o contexto em que o experimento desse trabalho foi desenvolvido para que se eleja a melhor solução. As variáveis analisadas serão: pré requisitos para o desenvolvimento, prazo para completar o experimento, recursos dos *smartphones* que o aplicativo precisará utilizar.

Pré requisitos para o desenvolvimento: essa variável está diretamente ligada ao custo de produção. Tem-se disponível para o desenvolvimento do aplicativo um computador rodando sistema operacional OS X e a licença de desenvolvedor obtida através do iOS Developer University Program. Caso contrário, somente o jQuery Mobile proporcionaria o desenvolvimento do aplicativo.

Tempo disponível para o desenvolvimento: 2 semanas para aprender usar o *framework* e desenvolver o aplicativo. Somente o *framework* Titanium não proporcionou a execução da tarefa dentro do prazo. Esse fato elimina a possibilidade desse *framework* ser a melhor opção.

Recursos dos *smartphones* que o aplicativo precisará utilizar: persistência dos dados, utilização do recurso de escolher imagem, utilização do recurso de geolocalização, integração com aplicativos

pré-instalados, efetuar ligação, envio de SMS, envio de e-mail, renderização adequada da interface gráfica. De acordo com os testes relatados na Tabela 4, o PhoneGap respondeu conforme esperado em relação aos recursos utilizados pelo aplicativo. As deficiências do jQuery Mobile em relação a esses recursos são explicadas no Capítulo 4.

Considerando a análise acima, conclui-se que o *framework* que melhor atingiu as necessidades do desenvolvimento do aplicativo do presente trabalho foi o PhoneGap.

5 CONCLUSÕES

Pode-se concluir com a realização do presente trabalho que existe um mercado de dispositivos móveis em acelerado crescimento. A comunidade de desenvolvedores tem buscado soluções que lhes proporcione atender as expectativas dos usuários desse mercado. Os *frameworks* que geram aplicativos para múltiplas plataformas móveis parecem ser uma boa solução, visto a grande quantidade de *frameworks* existentes com essa proposta.

Esse trabalho analisou e comparou três desses *frameworks*. O jQuery Mobile que representou o grupo de *frameworks* que geram aplicativos Web, o PhoneGap que representou o grupo que geram aplicativos Híbridos e o Titanium representando os que geram aplicativos Nativos.

O conhecimento prévio das características desses *frameworks* permitirá aos desenvolvedores escolher o que melhor atenda suas necessidades, fazendo com que eles ganhem em produtividade, uma vez que evitam o retrabalho no caso de uma escolha equivocada.

Dos *frameworks* analisados pode-se concluir que o jQuery Mobile é mais indicado pra aplicativos simples, que não têm a pretenção de utilizar muitos recursos dos *smartphones*. É também uma opção para quem possui recursos financeiros limitados. O Titanium é indicado aqueles que pretendem aproveitar ao máximo os recursos dos *smartphones*, mas que o tempo de aprendizado de sua utilização não seja um fator crítico, uma vez que ele se mostrou mais demorado na geração do experimento. O PhoneGap foi o que melhor atendeu às necessidades do aplicativo aqui desenvolvido, uma vez que atendeu as funcionalidades do aplicativo dentro do prazo determinado.

O levantamento e teste dos *frameworks* ajudou conhecer melhor as diferentes abordagens de FMM existentes no mercado e o desenvolvimento do protótipo permitiu compará-los na prática e levantar suas vantagens e desvantagens, a abrangência e limitações de cada um.

O trabalho foi conduzido utilizando-se referências bibliográficas de qualidade, ou seja, tem forte embasamento teórico e prático, e portanto, pode-se concluir que foi elaborado visando a qualidade do conteúdo exposto. Dessa forma, pode ser utilizado pela comunidade de desenvolvedores que visam a qualidade do produto que desenvolvem.

5.1 TRABALHOS FUTUROS

Um item não abordado no presente trabalho, foi o processo de submissão e admissão do aplicativo nas *app stores*, pois o valor da licença de desenvolvedor que permitiria tal teste foi fator impeditivo para o autor. Porém, como foi dito no Capítulo 2, a maneira como os aplicativos são instalados é um fator importante para melhor entendimento dos FMM, este experimento poderia fazer parte de um trabalho futuro.

Trabalhos futuros podem ser desenvolvidos dando continuidade a este projeto no que diz respeito a contemplar os outros sistemas operacionais existentes e as outras funcionalidades oferecidas pelos *smartphones* que não foram contempladas.

REFERÊNCIAS BIBLIOGRÁFICAS

© RESEARCH2GUIDANCE. **Global Cross Platform Tool Benchmarking 2013.** © research2guidance. Berlin. 2013.

ABLESON, F. **DeveloperWorks.** ibm.com, 12 Maio 2009. Disponivel em: <<http://www.ibm.com/developerworks/br/library/os-android-devel/>>. Acesso em: 30 Março 2013.

ADOBE®. **Learn more - Adobe Air.** Adobe, 01 June 2013. Disponivel em: <<http://get.adobe.com/air/?promoid=JOPDE>>. Acesso em: 01 June 2013.

ADOBE PHONEGAP BUILD. **Home: Adobe PhoneGap Build,** 2014. Disponivel em: <<https://build.phonegap.com/>>. Acesso em: 2014.

APPCELERATOR. **Titanium Mobile Application Development | Appcelerator Inc.** Appcelerator.com, 01 June 2013. Disponivel em: <<http://www.appcelerator.com/platform/titanium-platform>>. Acesso em: 01 June 2013.

APPCELERATOR. Documentation: Titanium. **iOS Platform Overview,** 2014. Disponivel em: <http://docs.appcelerator.com/titanium/latest/#!/guide/iOS_Platform_Overview>. Acesso em: 2014.

APPCELERATOR. Documentation: Titanium. **Android Platform Overview,** 2014. Disponivel em: <http://docs.appcelerator.com/titanium/latest/#!/guide/Android_Platform_Overview>. Acesso em: 2014.

APPCELERATOR INC. Documentation: Appcelerator Titanium. **Titanium - Titanium 3.X - Appcelerator Docs,** 2013. Disponivel em: <<http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.UI.ListView>>. Acesso em: 15 May 2014.

APPLE. **iOS Developer University Program,** 2014. Disponivel em: <<https://developer.apple.com/programs/ios/university/>>. Acesso em: 2014.

BARROS, A. J. D. S.; LEHFELD, N. A. D. S. **Fundamentos de metodologia científica: um guia para a iniciação científica.** São Paulo: Makron Books, 2000.

BISHOP, J.; HORSPOOL, N.; PRETORIA, U. O. **Cross-Platform Development: Software that Lasts.** Published in: Computer, 39, n. 10, 09 Oct 2006. 26-35.

CABELLO, S.; PHILLIPS, T. **Relatório: Observatório Móvel Brasil 2012.** GSMA.com, Londres, p. 84, 2012. Disponivel em: <<http://www.gsmworld.com/publicpolicy/public-policy-resources/mobile-observatory-series>>. Acesso em: 08 abr. 2013.

CODIQA. **About: Codiqa,** 2014. Disponivel em: <<https://codiqa.com/about>>. Acesso em: 2014.

CORONA LABS INC. **Corona Labs: About.** Corona, 2013. Disponivel em: <<http://coronalabs.com/about/>>. Acesso em: 05 out. 2013.

DIO-SYNODINOS. **InfoQ: Research - What are the Most Important and Mature Cross Platform Mobile Tools?** InfoQ, 21 Aug 2012. Disponivel em: <<http://www.infoq.com/research/cross-platform-mobile-tools#>>. Acesso em: 3 May 2013.

GALLIANO, G. A. **O Método Científico: teoria e prática.** São Paulo : [s.n.], 1979. p. 6.

GAMMA, E. et al. **Padrões de Projeto - Soluções reutilizáveis de software orientado a objetos.** Tradução de Luiz A. Meirelles Salgado. Porto Alegre: bookman, 2000. 41-43 p.

HEGENBERG, L. **Etapas da investigação científica.** São Paulo: E.P.U./EDUSP, v. 2, 1976. Capítulo 4.

HUMAYOUN, S. R.; EHRHART, S.; EBERT, A. Developing Mobile Apps Using Cross-Platform Frameworks: A Case Study. In: INTERACTION, T. 1. I. C. O. H. **Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments.** Las Vegas: Springer Berlin Heidelberg , v. 8004, 2013. p. 371-380. ISBN 978-3-642-39231-3.

JONES, S. Cross-Platform Developer Tools. **VisionMobile**, 2012. Disponível em: <<http://www.visionmobile.com/blog/2012/02/crossplatformtools/>>. Acesso em: 24 out. 2013.

JQUERY FOUNDATION. **jQuery Mobile: Introduction.** jQuery Mobile Web Site, 09 April 2013a. Disponível em: <<http://view.jquerymobile.com/1.3.1/dist/demos/intro/>>. Acesso em: 09 April 2013.

JQUERY FOUNDATION. **jQuery Mobile: Going Responsive.** jQuery Web Site, 10 June 2013b. Disponível em: <<http://view.jquerymobile.com/1.3.1/dist/demos/intro/rwd.html>>. Acesso em: 10 June 2013.

KAPLAN, A. **A conduta na pesquisa.** São Paulo: EDUSP, 1980.

KELINGER, F. N. **Metodologia da Pesquisa em Ciências Sociais - Um tratamento conceitual.** Tradução de Helena Mendes Rotundo. São Paulo: EPU, 1980.

MARMALADE. **SDK Overview**, 09 June 2013. Disponível em: <<https://www.madewithmarmalade.com/sdk>>. Acesso em: 09 June 2013.

MENEZES, E. M.; SILVA, E. L. **Metodologia da pesquisa e elaboração de dissertação.**, 2005. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CC8QFjAA&url=https%3A%2F%2Fprojetos.inf.ufsc.br%2Farquivos%2FMetodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes_4ed.pdf&ei=RIN0U4eJFtPkASvgIGoDA&usg=AFQjCNGI-LZ5rkj7EV5xsattKMyssorfqA&sig2=LBBelw7CLwGwu0GPghoflg&bvm=bv.66917471,d.cWc>. Acesso em: 14 maio 2014.

MOSYNC AB. **Native mobile app development for multiple platforms using a single code base.** Mosync, 2014. Disponível em: <<http://www.mosync.com>>. Acesso em: 04 jun. 2014.

MOTOROLA. **Suite RhoMobile.** Motorola Solutions, May 2013a. Disponível em: <<http://www.motorolasolutions.com/XL-PT/Produtos+e+Servicos+para+Empresas/Software+e+Aplicativos/Suite+RhoMobile>>. Acesso em: 13 May 2013.

MOTOROLA. **RhoElements.** Motorola Solutions, May 2013b. Disponível em: <<http://www.motorolasolutions.com/XL-PT/Produtos+e+Servicos+para+Empresas/Software+e+Aplicativos/Suite+RhoMobile/RhoElements>>. Acesso em: 13 May 2013.

OHRT, J.; TURAU, V. **Cross-Platform Development Tools for Smartphone Applications.** Computer, 45, n. 9, Sept 2012. 72 - 79.

PHONEGAP. **About us: PhoneGap.com**, May 2013a. Disponível em: <<http://phonegap.com/about/>>. Acesso em: 02 May 2013.

PHONEGAP. **PhoneGap Features**, 2013b. Disponível em: <<http://phonegap.com/about/feature/>>. Acesso em: 01 June 2013.

QT PROJECT. **About us: Qt Project.** Qt Project, 05 May 2013. Disponível em: <<http://qt.digia.com/About-us/#.UbT-MhY1fdk>>. Acesso em: 05 May 2013.

RUIZ, J. Á. **Metodologia científica: guia para eficiência nos estudos.** São Paulo: Atlas, 1996.

SENCHA. **Products: Sencha Touch.** Sencha Web Site, 10 June 2013. Disponível em: <<http://www.sencha.com/products/touch/>>. Acesso em: 10 June 2013.

SOMMER, A. **Comparison and evaluation of cross-platform frameworks for the development of mobile business applications.** Technische Universität München. München. 2012.

STALLINGS, W. **Operating systems: internals and design principles.** 7ed. ed. [S.l.]: Prentice Hall, 2012. 768 p.

THEMEROLLER. **ThemeRoller for jQuery Mobile.** ThemeRoller for jQuery Mobile, 2014. Disponível em: <<http://themeroller.jquerymobile.com>>. Acesso em: 2014.

W3C. **W3C: Web Storage.** W3C Editor's Draft, 2014 May 2014. Disponível em: <<http://dev.w3.org/html5/webstorage/#disk-space>>. Acesso em: June 2014.

W3SCHOOLS. W3Schools.com. **W3Schools Online Web Tutorials**, 2014. Disponível em: <<http://www.w3schools.com>>. Acesso em: 2014.

WAINER, J. **Métodos de pesquisa quantitativa e qualitativa para a ciência computação.** Atualização em informática 2007. Rio de Janeiro: Sociedade Brasileira de Computação. 2007. p. p. 221-262.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação.** Rio de Janeiro: Elsevier Editora Ltda, 2009. ISBN ISBN 978-85-352-3522-7.

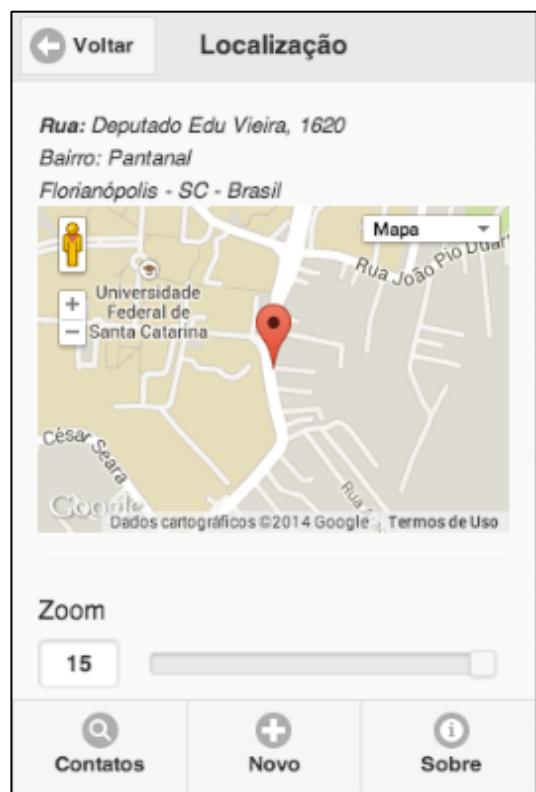
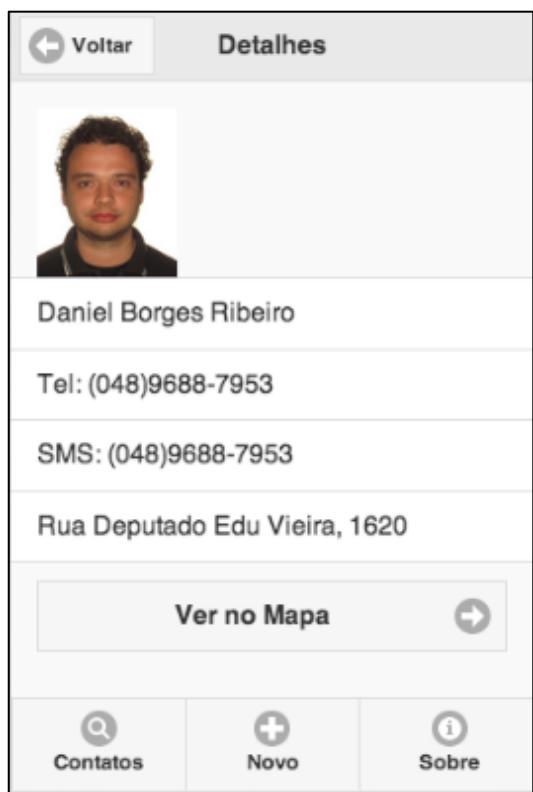
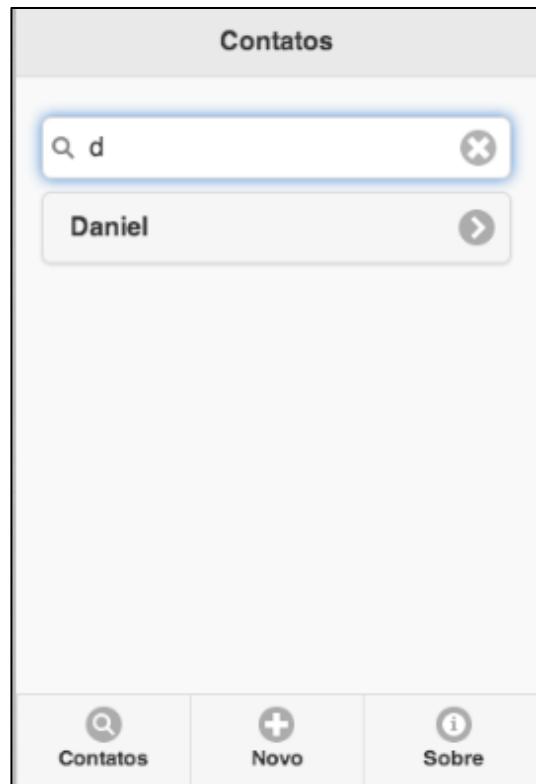
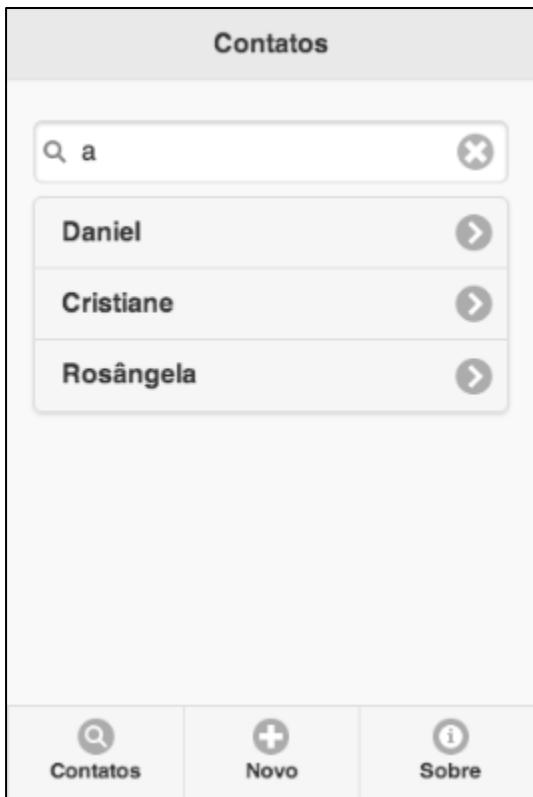
WEB DIRECTIONS. **A Survey on Mobile Development.** InfoQ.com, 05 Maio 2011. Disponível em: <http://www.infoq.com/news/2011/05/A-Survey-on-Mobile-Development?utm_source=infoq&utm_medium=related_content_link&utm_campaign=relatedContent_interviews_clk>. Acesso em: 29 Março 2013.

WHINNERY, K. **Comparing Titanium and PhoneGap.** Appcelerator Developer Blog, 12 May 2012. Disponível em: <<http://developer.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap.html>>. Acesso em: 04 April 2013.

WIRELLES INTELLIGENCE, 2012. Disponível em: <<https://wirelessintelligence.com>>. The industry's definitive source of data analysis.

XAMARIAN. **How it works:** Xamarian, 06 June 2013. Disponivel em: <<http://xamarin.com/how-it-works>>. Acesso em: 06 June 2013.

APÊNDICE A – TELAS DO PROTÓTIPO AGENDA



Voltar **Cadastro**

No file chosen

Contatos Novo Sobre

Sobre

Desenvolvido
por
Daniel Borges Ribeiro
danielbribeiro@gmail.com

Contatos Novo Sobre

APÊNDICE B – IMAGENS DO APLICATIVO



Figura 13 Protótipo desenvolvido pelo jQuery Mobile rodando no Samsung Galaxy SIII e iPhone 4s

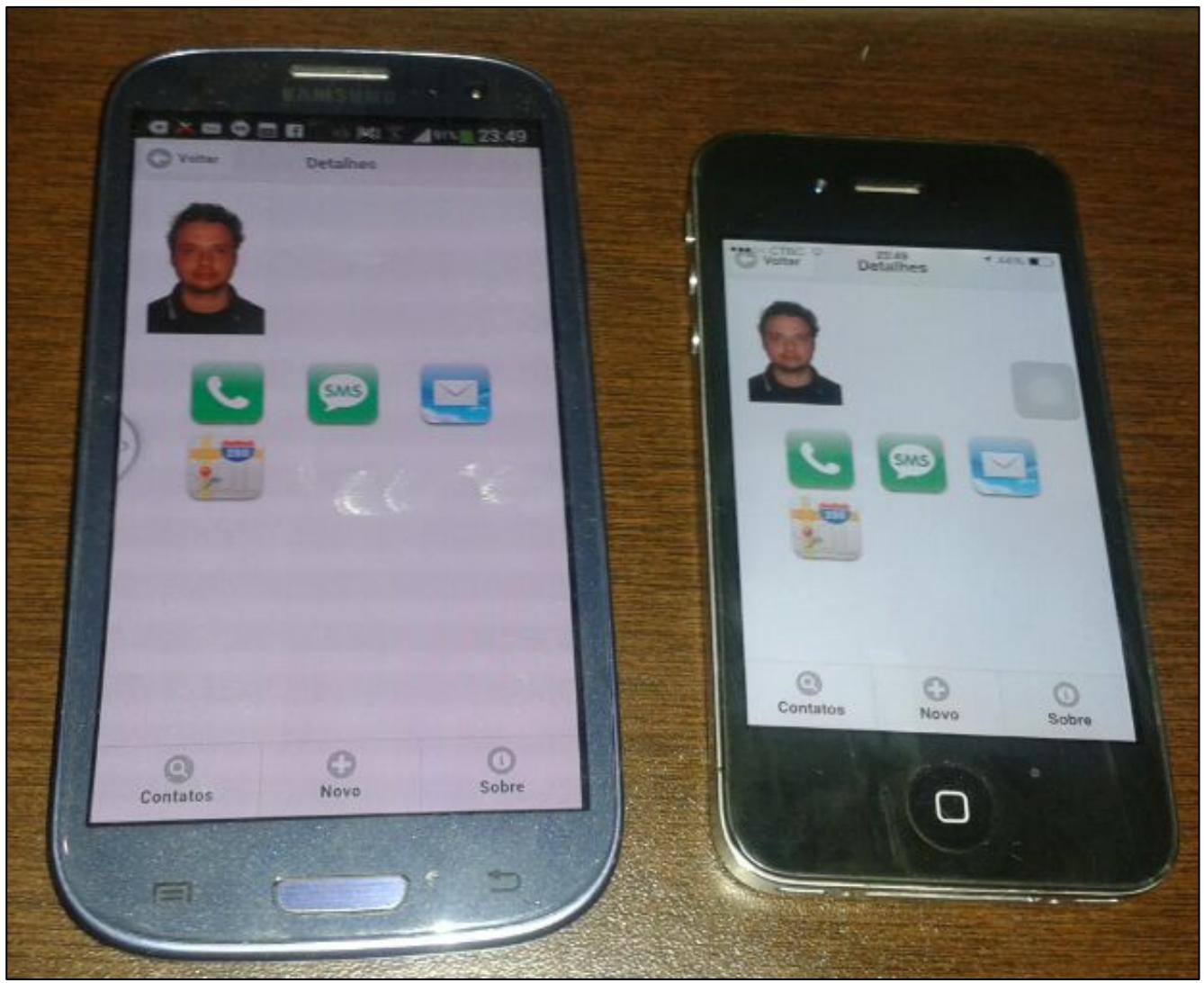


Figura 14 Protótipo desenvolvido pelo PhoneGap rodando no Samsung Galaxy SIII e iPhone 4s

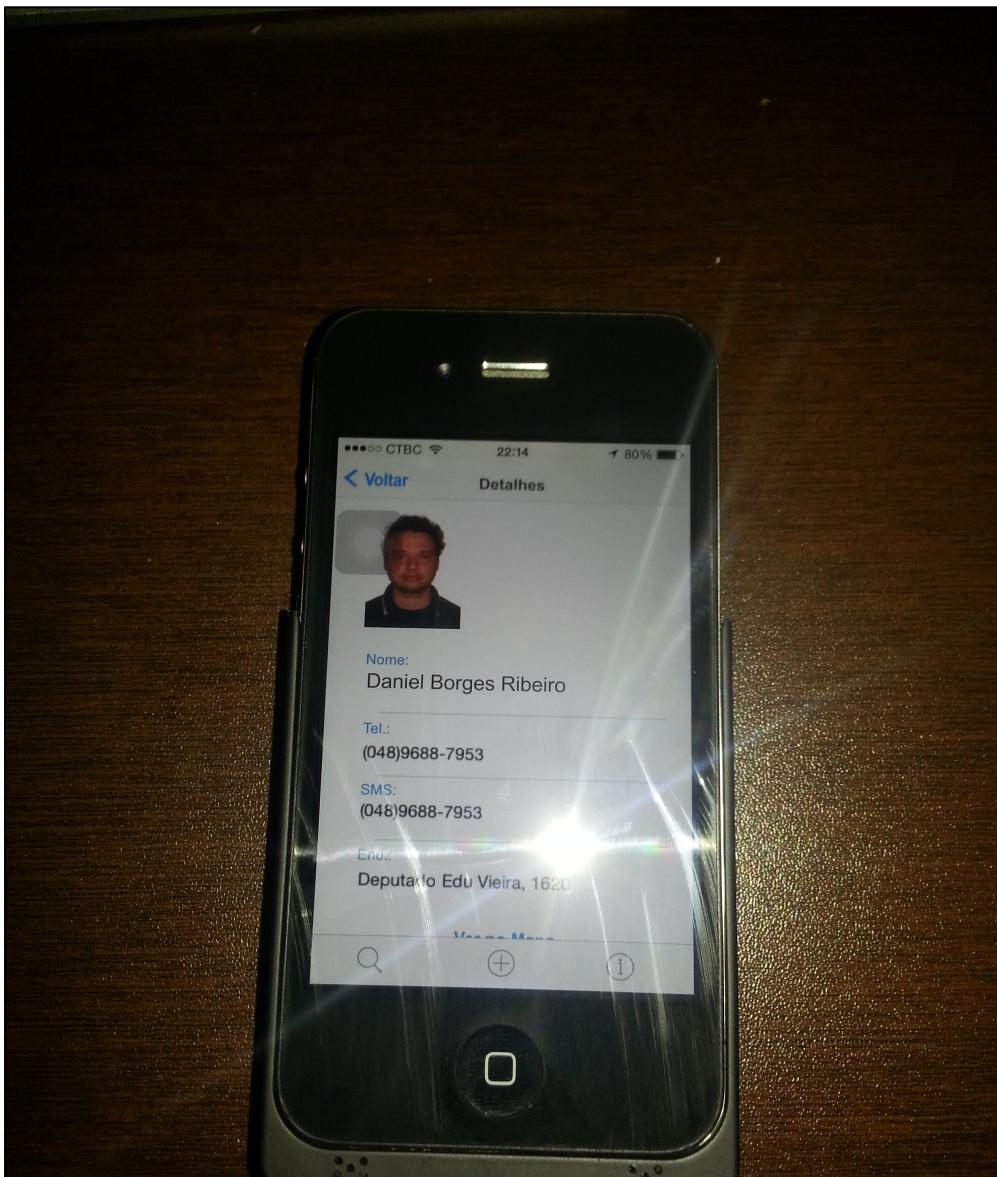


Figura 15 Protótipo desenvolvido pelo Titanium rodando no iPhone 4s