UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

SAMUEL SANTO DE MATOS

ANÁLISE COMPARATIVA DE APLICAÇÕES MOBILE NATIVA E APLICAÇÕES MOBILE CROSS-PLATAFORMA

CRICIÚMA

2016

SAMUEL SANTOS DE MATOS

ANÁLISE COMPARATIVA DE APLICAÇÕES MOBILE NATIVA E APLICAÇÕES MOBILE CROSS-PLATAFORMA

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. MSc. Luciano Antunes

CRICIÚMA

2016

SAMUEL SANTOS DE MATOS

ANÁLISE COMPARATIVA DE APLICAÇÕES MOBILE NATIVA E APLICAÇÕES MOBILE CROSS-PLATAFORMA

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Criciúma, 20 de junho de 2016.

BANCA EXAMINADORA

Prof. MSc. Luciano Antunes - UNESC - Orientador

Prof. MSc. Gustavo Bisognin - UNESC

Prof. Esp. Fabrício Giordani - UNESC

Dedico a minha família amigos e colegas que me incentivaram a fazer este trabalho.

“Cada descoberta nova da ciência é uma porta nova pela qual encontro mais uma vez Deus, o autor dela.”

Albert Einstein

RESUMO

Os dispositivos moveis estão cada vez mais evoluindo e se popularizando e trazendo consigo a demanda de desenvolvimento de aplicativos para tais dispositivos. Hoje no mercado existem inúmeros sistemas operacionais para estes dispositivos. Com esta variedade de dispositivos surgem duas formas de desenvolver um novo aplicativo entre elas estão as nativas e as cross-plataforma. As plataformas Android e Windows Phone atualmente estão entre os lideres deste seguimento e disponibilizam uma série de ferramentas para o desenvolvimento de novas aplicativos nativas. A Adobe por outro lada lidera o mercado de desenvolvimento cross-plataforma com maior compatibilidade e um simples desenvolvimento com seu *framework free* PhoneGap. No desenvolvimento de qualquer software, seja para dispositivos moveis ou não, é importante avaliar o custo de desenvolvimento e desempenho das tecnologias utilizadas. As métricas de testes são uma das maneiras de avaliar o custo de desenvolvimento e desempenho de um *software.* A avaliação de custo de desenvolvimento e desempenho é justamente o foco deste trabalho que tem o intuito de mostrar o custo de desenvolvimento e desempenho de aplicativos *mobile* cross-plataforma comparado com aplicativos *mobile* nativos. O aplicativo desenvolvido foi o aplicativo Achando, que foi desenvolvido nativamente em Windows Phone e Android e cross-plataforma em PhoneGap, a aplicação desenvolvida é um aplicativo de achados e perdidos onde o usuário pode efetuar os testes de desempenho e cadastras e consultar itens achados e perdidos. As métricas utilizadas para efetuar as medidas foram a métrica orientada ao tamanho para medir o custo de desenvolvimento de cada aplicativo e método de sobrecarga para avaliar o tempo gasta por cada funcionalidade, assim avaliando o desempenho. Com a análise comparativa da mescla das métricas definidas nesta pesquisa foi possível verificar a melhor utilidade de cada tecnologia e seus pontos fortes. Também foi possível identificar o custo de desenvolvimento para cada aplicação e o desempenho, assim os resultados obtidos pela mescla de métricas e análise comparativa mostram que se faz necessário avaliar o custo de desenvolvimento e desempenho em qualquer desenvolvimento de *software.*

Palavras-chave: Análise Comparativa. Cross-plataforma. Desenvolvimento. Desempenho. Métrica Orientada ao tamanho. PhoneGap. Plataforma Android. Plataforma Windows Phone.

ABSTRACT

The mobile devices are increasingly evolving and popularizing and bringing the application development demand for such devices. Today in the market there are numerous operating systems for these devices. With this variety of devices come two ways to develop a new application among them are native and cross-platform. The Android and Windows Phone platforms are currently among the leaders of this follow-up and provide a number of tools for the development of new native applications. Adobe on the other lada leads the cross-platform development market with greater compatibility and simple development with its free PhoneGap framework. In the development of any software, whether for mobile devices or not, it is important to assess the cost of development and performance of the technologies used. Metric tests are one way to assess the cost of development and performance software. Evaluation development and performance cost is precisely the focus of this work aims to show the development cost and performance of mobile applications cross-platform compared to native mobile applications. The application developed was the Finding application that was developed natively on Windows Phone and Android and cross-platform PhoneGap, the developed application is a found application and lost where the user can make performance testing and cadastras and consult items found and lost. The metrics used to make the measurements were metric-oriented size to measure the cost of development of each application and overload method to evaluate the time spent for each feature, and evaluating performance. With the comparative analysis of the mix of metrics defined in this study it was possible to verify the best use of each technology and its strengths. It was also possible to identify the cost of development for each application and performance, and the results obtained by the mix of metrics and comparative analysis show that it is necessary to assess the cost of development and performance in any software development.

Keywords: Comparative Analysis. Cross-platform. Development. Performance. Oriented metrics to size. PhoneGap. Android platform. Platform Windows Phone.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADT	*Android Developer Tool*
API	*Application Programming Interfac*
AVD	*Android Virtual Devices*
CLI	*Command-Line Cnterface*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
DVM	*Máquina Virtual Dalvik*
HTML	*HyperText Markup Language*
IDC	*International Data Corporation*
IDE	*Integrated Development Environment*
JVM	*Java Virtual Machine*
KLOC	*Thousand Lines of Code*
OHA	*Open Handset Alliance*
RAD	*Rapid Application Development*
RIA	*Rich Interactive Applications*
SDK	*Software Development Kit*
SGML	*Standard Generalized Markup Language*
SO	*Sistema Operacional*
XAML	*Extensible Application Markup Language*
XML	*Extensible Markup Language*
WORA	*Write Once Run Anywhere*
WPE	*Windows Phone Emulator*
WVGA	*Wide Video Graphics Array*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, com o crescimento de tecnologias para dispositivo moveis, observou-se que a diversidade de plataformas e aplicativos para os mesmos aumentou consideravelmente, consequentemente as plataformas e *Integrated Development Environment* (IDE) evoluíram junto. Esta situação impacta diretamente nos custos de desenvolvimento, pois desta forma a complexidade de atender todas as plataformas disponíveis aumenta, pois é necessário conhecimentos específicos da linguagem de cada plataforma, e também é necessário desenvolver uma aplicação diferente para cada uma delas (PREZOTTO; BONIATI, 2014).

Segundo Kessin (2011), visando sanar estes problemas surgiu as *webapps* e as *hybrid apps*, na web as aplicações são Cross-plataforma assim como é possível escrever páginas de internet que irão funcionar no Windows, Mac OS X, Linux, iPhone/iPad, Android entre outras plataformas. Como toda plataforma mobile possui browsers que são capazes de interpretar scripts do JavaScript e reproduzir documentos HyperText Markup Language (HTML) customizados com Cascading Style Sheets (CSS), é possível desenvolver *Apps* que irão funcional em qualquer plataforma existente para dispositivos moveis.

As *webapps* são basicamente websites otimizados para rodar em dispositivos moveis, as *webapps* rodam em uma *webview*, componente visual que representa um browser no aplicativo. Botões, campos de texto, rótulos e tabelas são conhecidos como *widgets*, estes *widgets* fazem parte da interface do aplicativo, sendo responsáveis pela interação com o usuário. Nos *webapps* a aparência destes *widgets* é bem próxima, por exemplo, dos *widgets* do iOS ou Android, isso varia de acordo com o *framework* que está sendo utilizado (STARK, 2010).

Os *hybrid apps* são como uma aplicação onde toda a lógica de negócios e a UI são feitas utilizando HTML, CSS e JavaScript e que roda em um *native wrapper*. Ele também afirma que os aplicativos híbridos possuem acesso limitado ao hardware e certas funcionalidades dos aparelhos, entretanto podem acessar mais funcionalidades do que os *webapps* (POULSEN, 2012).

Outra característica dos aplicativos híbridos é a possibilidade de publicá-los nas lojas de aplicativos oficiais e não requererem conexão ativa com a internet, assim podendo ser tendo a mesma funcionalidade que uma aplicação nativa de cada plataforma (BUDIU, 2013).

Em resumo, as aplicações cross-platform para dispositivos moveis são aplicativos desenvolvidos utilizando tecnologias web que suportam múltiplas plataformas. Eles podem ser *webapps* ou *hybrid apps*.

Então consequentemente surgiu a necessidade de desenvolver novos frameworks para gerar aplicações cross-plataforma, segundo Allen, Graupera e Lundrigan (2010), estes novos frameworks são influenciados pelo *Rapid Application Development* (RAD), modelo de processo de desenvolvimento de software muito utilizada no desenvolvimento de aplicações web.

Na utilização do modelo de desenvolvimento RAD que visa um prazo em media de 60 a 90 dia para entrega do produto, o conceito *Write Once, Run Anywhere* (WORA) é se faz necessário quando se trata de aplicações cross-plataforma *mobile*, pois quando se desenvolve uma aplicação que seja cross-plataforma utiliza o conceito WORA, em português (Escreva Uma Vez, Rode Em Qualquer Lugar) (CAFÉ, 2012).

WORA é um conceito que fortalece a reutilização do código, interoperabilidade entre diversas plataformas e o desenvolvimento de aplicativos cross-platform, seja nas plataformas *desktop*, *web* ou *mobile* (SHACKLES, 2012).

Assim o produto final tem um padrão único para todas as plataformas, um custo menor para a sociedade e compatibilidade do *app* com todas as plataformas desde mais cara com o dispositivo mais caro até a plataforma mais barata, pois todas as plataformas terão o mesmo *app*, facilitando no suporte ao usuário e manutenção, gerando menos custos para os desenvolvedores (ALLEN; GRAUPERA; LUNDRIGAN, 2010).

Hoje as plataformas possuem características distintas que tornam elas mais atrativas para os usuários e desenvolvedores, normalmente os usuários procuram os dispositivos mais fáceis de serem manipulados, que tenham mais funcionalidades e um bom custo-benefício. Mas os desenvolvedores procuram plataformas com uma grande quantidade de dispositivos ativos, que tenha uma boa documentação e suporte técnico, de fácil aprendizado, baixo custo de desenvolvimento e tecnologias familiares (CAFÉ, 2012).

Desta forma, a pesquisa aqui proposta, ressalta a necessidade de desenvolver um aplicativo, usando tecnologia nativa e cross-plataforma, e efetuar testes e análise comparativa de desempenho e custo entre elas, visando identificar qual aplicação terá o maior desempenho, também evidenciar quais as dificuldade

que os desenvolvedores terão ao iniciar o desenvolvimento usando tecnologia cross-plataforma.

1.1 OBJETIVO GERAL

Analisar comparativamente aplicações mobile nativa e aplicações mobile cross-plataforma.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) analisar e definir métricas de teste comparativo de Software;
b) compreender o conceito de desenvolvimento de aplicativos cross-plataforma e nativa;
c) compreender *framework* Phonegap;
d) desenvolver aplicação nativas e *cross-plataforma* para Android e Windows Fone;
e) aplicar testes comparativos entre aplicações desenvolvidas *cross-plataforma* e nativa;
f) aplicar métrica de teste definida;
g) analisar os resultados dos testes e medidas empregados.

1.4 JUSTIFICATIVA

Segundo Prezotto e Boniati (2014), existem muitas plataformas disponíveis no mercado hoje, como Android, iOS, Windows Phone, Firefox OS, BlackBerry, Ubuntu Touch, Fire OS, entre outros. Essas plataformas exigem que o desenvolvedor possua conhecimento em diferentes linguagens de programação, como por exemplo, o Android utiliza Java como linguagem de programação, Firefox OS utiliza linguagens web (HTML, CSS, JavaScript), iOS utiliza Objetive C.

Cabe ao desenvolvedor ou a equipe de desenvolvimento optar pela plataforma e pela quantidade de sistemas diferentes que a aplicação deve dar suporte e assim escolher a forma de desenvolvimento que melhor vai se adequar. Ao fazer esta análise deve-se levar em conta, os conhecimentos dos

desenvolvedores, a quantidade de plataformas que deverão ser suportadas, o tempo para desenvolvimento, e a quantidade de recursos que a aplicação vai consumir (ALLEN; GRAUPERA; LUNDRIGAN, 2010).

Com a ideia de criar uma única aplicação para qualquer plataforma, utilizando uma única ferramenta e linguagem especifica, traz duvidas do desempenho e custo da aplicação por parte dos desenvolvedores (CAFÉ, 2012).

Segundo Café (2012), o desenvolvimento de um aplicativo cross-plataforma traz vários benefícios, mas também para que as aplicações sejam desenvolvidas com sucesso será necessário um esforço adicional e fazer alguns sacrifícios, pois um aplicativo cross-plataforma deve seguir os padrões de cada plataforma, muitas vezes será necessário escrever trechos de códigos especifico para cada um.

Mas em contra partida essa solução terá um custo menor no desenvolvimento, pois não será necessário muitos desenvolvedores e desenvolver várias aplicações para atender duas ou mais plataforma (MOBILE, 2012).

Dentre os fatores que levaram a necessidade de efetuar testes comparativo de desempenho e custo entre aplicações cross-plataforma e Nativas, foi a dúvida que desenvolvedores possuem para decidir qual será a melhor forma de desenvolver aplicativos que rode em várias plataformas, que o custo de manutenção e suporte a esta aplicação seja baixo, assim disponibilizando a sociedade aplicações baratas.

2 DISPOSITIVOS MOVEIS

Dispositivos moveis são aparelhos que tem um tamanho reduzido, com capacidade de processamento, com bateria de longa duração e ter acesso a rede sem fio, permitindo assim a troca de informações entre um ou mais dispositivos (FIGUEIREDO; NAKAMURA, 2003).

Atualmente, conforme na figura 1, a vários dispositivos moveis de tamanho e modelos distintos (Smartphone, PDA, Telemóvel, Celular, Console portátil, Ultra Mobile PC, Ultrabook, Notebook, Netbook, Laptop, tabletes e Coletor de dados), os mais comuns são os Smartphone e tabletes, pois cerca de 80% dos dispositivos moveis conectados a internet atualmente (LEE, 2011, tradução nossa).

Figura 1 – Dispositivos moveis

Fonte: Lee (2010).

Os dispositivos moveis aparecem como destaque aos outros tipos de dispositivos quando se trata de disseminação e consumo no mercado atual. Segundo a Gartner Group (2013, nossa tradução) em 2012 foram vendidos cerca de 340 milhões de *desktops* e *notbook* no mundo e mais de 1,7 bilhões de celulares. As pesquisas realizadas pela Global System for Mobile communications (2011, nossa tradução) mostra que em 2020 haverá mais de 12 bilhões de dispositivos moveis conectados à internet.

A viabilidade da utilização destas tecnologias está aumentando devido ao fácil acesso da população. De acordo com pesquisas realizada pelo IGBE, em 2011,

existiam 70.2 milhões de pessoas de 10 anos ou mais que tinham telefone celular para uso pessoal (IBGE, 2013).

Devido a facilidade da população adquirir a tecnologia e as várias características positivas como a de possibilitar o envio de dados em movimento, a não precisar se preocupar com cabos de energia, ter acesso a redes sem fio, facilidade no transporte entre outros recursos, estes aparelhos estão se espalhando de forma acelerada.

A partir dos anos 90 foi possível identificar um grande crescimento no desenvolvimento da tecnologia móvel. Diante desta evolução obteve-se uma grande popularização destes dispositivos que passaram a permitir o acesso remoto as informações onde quer que se esteja, colaborando assim com um vasto campo de facilidades, aplicações e serviços aos usuários (FIGUEIREDO; NAKAMURA, 2003).

A área de dispositivos móveis está ficando cada vez mais ampla, e com este crescente progresso na tecnologia os profissionais tendem aprimorar seus conhecimentos para ter maior domínio sobre as ferramentas (DARIVA, 2011).

Estes aparelhos foram criados com o intuito de fornecer algum auxílio na vida das pessoas, pois atualmente são utilizados para muitas tarefas, como despertadores, meios de acesso rápido para se ler notícias, emails entre outras facilidades disponibilizadas (DARIVA, 2011). Desta forma é importante o conhecimento sobre os dispositivos analisados, para verificar qual poderá suprir melhor as necessidades que os testes e as análises necessitarão.

2.1 SISTEMA OPERACIONAL ANDROID

O Android é desenvolvido principalmente pela Google, em colaboração com a *Open Handset Alliance* (OHA), uma aliança de dezenas de organizações, mais precisamente 84 empresas, que se comprometeram em disponibilizar no mercado um telefone celular *open-source* e com um maior número de vantagens. Em poucos anos o Android cresceu e foi capaz de mudar o mercado, ganhando o respeito como escárnio dos colegas de indústria (ABLESON et al, 2012).

O Android é um SO móvel baseado em uma versão modificada do Linux. Foi originalmente desenvolvido por uma equipe com mesmo nome, Android, Inc. Em 2005 a Google comprou o Android e assumiu o seu trabalho de desenvolvimento (bem como sua equipe de desenvolvimento), isso fazia parte de uma estratégia para

entrar no espaço de dispositivos móveis (LEE, 2011, tradução nossa).

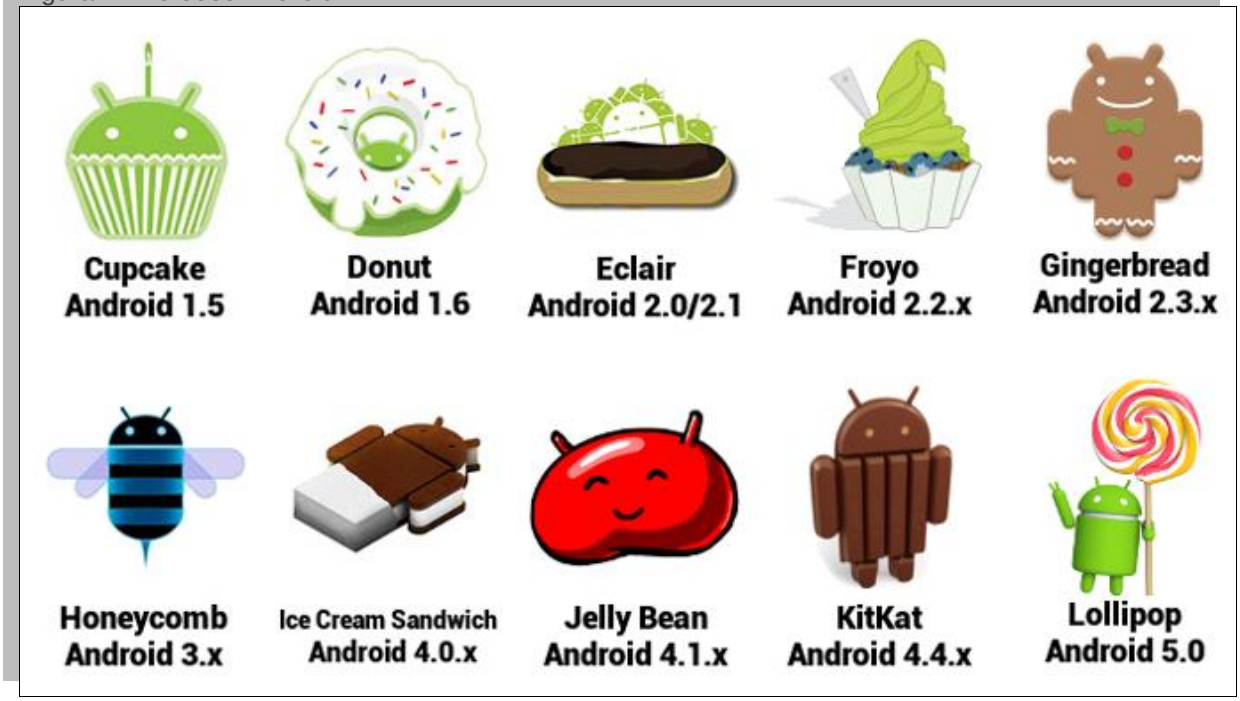
A Google queria que o Android fosse *open-source* e *software* livre, portanto, a maioria do código do Android foi liberado sob a licença Apache *open-source*, deste modo qualquer um que tivesse interesse em utiliza-lo teria fácil acesso ao download do código fonte completo, além disso as empresas podem adicionar suas próprias extensões e customiza-lo, diferenciando o seu produto dos demais (LEE, 2011, tradução nossa).

Por ser a primeira plataforma de software que possui aplicativos para telefones celulares *open-source*, o Android vem revolucionando o mercado mundial de aplicativos para smartphones e sendo demasiadamente notado pelos maiores mercados de telefonia celular do globo (ABLESON et al, 2012).

Considerado uma plataforma para tecnologia móvel completa, o Android, desenvolvido com base no Linux, possui um *Sistema Operacional* (SO) middleware, com já citado, aplicativos instalados, uma interface com o usuário e envolve um pacote de programas para celulares. Foi construído com a intenção de permitir os desenvolvedores tirarem total proveito do aparelho, desta forma pode-se criar aplicações que acessem qualquer funcionalidade do núcleo do sistema, tais como efetuar chamadas, enviar mensagens ou qualquer possível atividade que o dispositivo venha a exercer. Com isto a tecnologia do Android sempre estará evoluindo, pois disponibiliza total controle do seu código fonte aos desenvolvedores, de tal modo que os mesmos sempre estarão trabalhando em cima do sistema incorporando novos recursos e construindo aplicações móveis inovadoras (PEREIRA; SILVA, 2009).

Atualmente conforme a figura 2 existem diversas versões do Android no mercado, desde as mais simples até as mais modernas. A plataforma está sempre em constante evolução, fornecendo aos usuários novos benefícios a cada aprimoramento ou lançamento de uma nova versão (GOOGLE, 2015, tradução nossa).

Figura 2 –Versões Android



Fonte: Google (2015).

Sabendo disso é de suma importância conhecer algumas partes essenciais que compõem o SO Android, tais como a arquitetura, Framework, SDK e emuladores.

2.1.1 Arquitetura do sistema Android

O Android foi desenvolvido inicialmente para *smartphones*, no entanto hoje é utilizado em diversos dispositivos. Seu SO foi baseado no Kernel 2.6 do Linux, responsável por realizar todo o controle da memória. Desta forma o Kernel consegue gerenciar processos e aplicativos, que podem ser executados simultaneamente, threads dos processos e a segurança dos arquivos e pastas (LECHETA, 2013).

Conforme figura 3 a arquitetura do Android é formada basicamente por cinco camadas, sendo elas as seguintes: aplicações, *framework*, bibliotecas, Android *runtime* e *linux kernel* (GOOGLE, 2015, tradução nossa).

Figura 3 – Arquitetura do Android

Fonte: Google (2015).

2.1.2 *Framework*

Frameworks são estruturas que servem tipicamente para a implementação de software de larga escala. Seu principal objetivo é possibilitar a reutilização do código e assim aumentar a produtividade durante o desenvolvimento de softwares (PEREIRA; SILVA, 2009).

Nesta camada, conhecida como *Application Framework*, são encontradas todas as APIs e recursos que são utilizados pelos aplicativos, como as classes visuais que englobam listas, grades, caixas de texto, gerenciadores de recursos, *View system*, que são componentes utilizados na construção de aplicativos, provedor de conteúdo, que possibilita que aplicações possam compartilhar suas informações, acessar e utilizar as de outra aplicação e trocar informação entre aplicativos, entre outras vantagens (PEREIRA; SILVA, 2009).

A camada de *framework* trabalha em conjunto com a de aplicações, desta forma fica mais fácil à reutilização dos componentes, ou seja, a troca de informação das aplicações. Com esta reutilização de código os desenvolvedores passam a ter

recursos prontos para serem aplicados e consequentemente menos códigos para escreverem (JOBSTRAIBIZER, 2009).

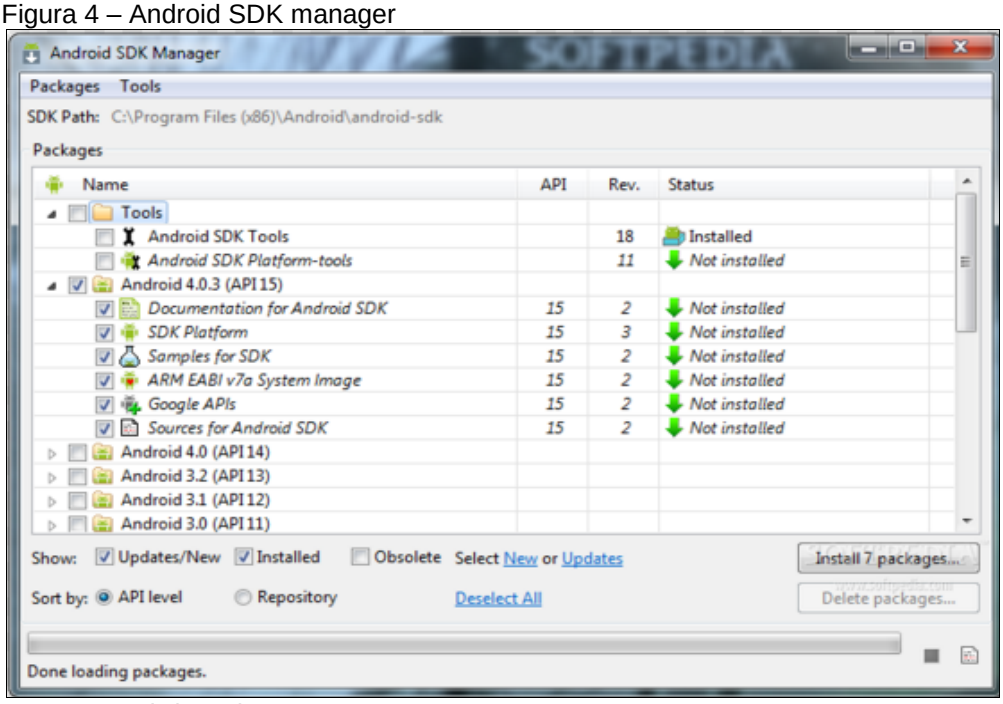
2.1.3 SDK Android

O *Software Development Kit* (SDK) do Android é utilizado para o desenvolvimento de aplicações. Para rodar as mesmas o SDK desfruta de um emulador similar a um celular, que dispõem de ferramentas utilitárias e uma API completa para a linguagem Java. O emulador do SDK pode ser executado como um aplicativo comum, assim podendo ser integrado com a IDE utilizada para desenvolvimento (LECHETA, 2013).

O SDK do Android é composto por uma plataforma, ferramentas, exemplos de códigos e documentação necessária para desenvolver aplicações Android. É construído como um add-on, ou seja, uma extensão para o Kit de desenvolvimento Java e tem um plug-in integrado para o ambiente de desenvolvimento (SCHWARZ, 2013, tradução nossa).

Para auxiliar o desenvolvedor a obter o SDK do Android, conforme na figura 4, a Google disponibiliza o Android SDK manager, neste painel e possível efetuar o *doawload* de todas as versões possíveis do SDK android (GOOGLE, 2015).

Figura 4 – Android SDK manager



Fonte: Google(2015).

2.1.4 Emulador Android

A partir da versão 1.6 do Android SDK os desenvolvedores obtiveram um melhor controle sobre o emulador do Android, podendo fazer downloads das plataformas que despertam um maior interesse, o que não era possível nas versões anteriores (ABLESON, 2012).

O *Android Virtual Devices* (AVD) é a ferramenta perfeita para testar e depurar aplicativos. Uma implementação da DVM que simula os softwares construídos em emuladores de diferentes dispositivos, assim é possível testar um aplicativo em uma variedade de plataformas de hardware, sem a necessidade de comprar tais dispositivos (MEIER, 2010, tradução nossa).

Para facilitar a inicialização do emulador existe o *Android Developer Tool* (ADT) plug-in, que integra o emulador no ambiente de desenvolvimento, com isto é possível inicia-lo automaticamente a cada compilação. Cada AVD criado é configurado com um nome e com a versão SDK requerida, onde é determinada a capacidade de cartão SD, resolução de tela ou qualquer característica da versão selecionada. Conforme figura 5, o emulador simula um telefone com uma ótima realidade, ele além de possuir o poder de testar as aplicações criadas pode fazer simulações com os próprios aplicativos integrados na versão escolhida, como mensagens de texto, chamadas de voz, entre outros (MEIER, 2010, tradução nossa).

Figura 5 – Emulador Android

Fonte: Meier (2010).

2.1.5 Linguagem Java

Java é a linguagem de programação orientada a objetos, desenvolvida pela *Sun Microsystems*, capaz de criar tanto aplicativos para desktop, aplicações comerciais, softwares robustos, completos e independentes, aplicativos para a Web e aplicações Android. Ela é muito parecida com C++, eliminando as características consideradas complexas, dentre as quais ponteiros e herança múltipla (CLARO; SOBRAL, 2000).

Seu lançamento em 1995 foi focado inteiramente em disponibilizar uma linguagem totalmente portável da maneira que ela fosse apropriada para implementação de aplicativos baseados na Internet e na *World Wide Web* (WWW), após isso disponibilizaram a linguagem gratuitamente (DEITEL, 2003).

O Java hoje é uma das plataformas mais importantes do mundo, estando presente em vários ramos da tecnologia, dispondo de amplos pontos positivos é uma das principais linguagens de desenvolvimento para dispositivos móveis, incluindo para o Android (SILVEIRA et al, 2012).

2.2 SISTEMA OPERACIONAL WINDOWS FONE

O sistema operacional Windows Mobile da família Windows, desenvolvido pela Microsoft, apesar de ter o nome do Windows, não é um sistema derivado ou uma versão colhida do mesmo, mas um novo sistema projetado especificamente para dispositivos móveis, isso não significa que ele não apresenta semelhanças aos sistemas operacionais para PC desenvolvido pela Microsoft, pois desta forma o SO permite que o usuário tenha uma familiaridade e uma experiência agradável com o dispositivo (CARDOSO ,2014).

Anteriormente conhecido como Windows Mobile, o Windows Phone foi desenvolvido para concorrer diretamente com Android da Google e o IOS da Apples, mas diferente dos concorrentes que optaram por ter uma interface e experiência com usuário muito parecidas, o Windows Phone chegou 3 anos depois com uma ideia diferente, conforme figura 6 o Windows Phone tem um aparência e similar ao SO Windows 8 e bem distinta de seus concorrentes (JIMÉNEZ; MENESES, 2013).

Figura 6 – Menus Android, IOs e Windows Phone

Fonte: Jimenez; Meneses (2013).

Na figura 6 na tela principal do sistema apresenta-se diversos quadrados ao invés de ícones, estes quadrados são conhecidos como *tiles*, esses tiles podem mostrar informações atualizadas a todo momento, tanto quanto informações locais do sistema, quanto informações extraídas da internet (CARDOSO, 2014).

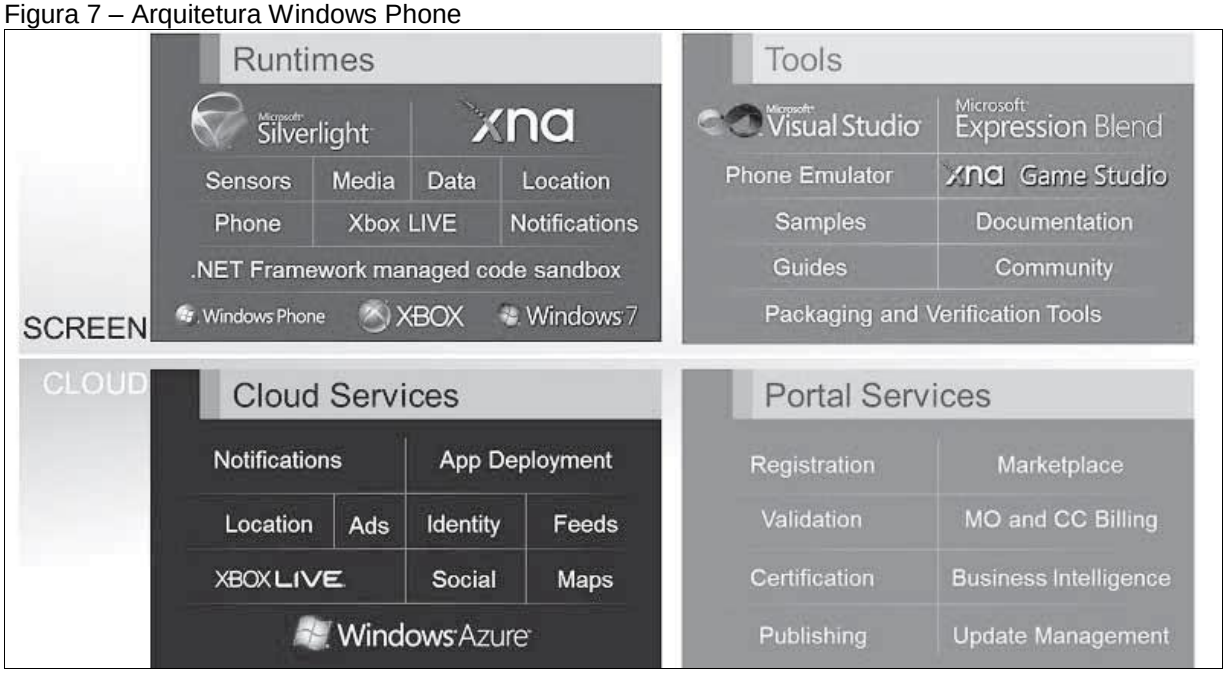
A Microsoft desenvolveu o Windows Phone com um intuito diferente, a solução foi desenvolvida para integrar todas as “telas” da Microsoft, de forma a ajudar a visão da empresa a se tornar realidade. Em qualquer apresentação da Microsoft é sempre falado das três telas, o Windows no seu PC, o Xbox na sua TV, e o Windows Phone em suas mãos, o Windows Phone em suas versões futuras será cada vez mais o hub de conexão entre todas as soluções Microsoft (MÔNACO; CARMO, 2012).

Windows Phone é totalmente integrado com o .NET Compact Framework fornecendo assim uma série de bibliotecas para desenvolvimento das aplicações. Estas aplicações poderão ser desenvolvidas a partir do próprio Visual Studio, ou pelo Visual Studio Express for Windows Phone, sendo que em ambos os produtos as aplicações poderão ser criadas utilizando Silverlight ou XNA (PETZOLD, 2010, tradução nossa).

2.2.1 Arquitetura do Sistema Windows Phone

A arquitetura do Windows Phone esta construída com base em quatro componentes principais, sendo eles os seguintes: *Runtimes*, *Tools*, *Coud services* e *Portals services* (MONACO, 2015).

Figura 7 – Arquitetura Windows Phone



Fonte: Monaco (2015).

Na figura 7, *Runtimes* e *Tools* são linguagens e ferramentas que através de códigos irão gerar a aplicação que será executada no aparelho, já *Coud* e *Portal Services* são serviços e soluções que podemos acessar via aplicativos (MÔNACO; CARMO, 2012).

A arquitetura utilizada hoje para o desenvolvimento de aplicativos para Windows Phone é o *Model-View-ViewModel* (MVVM), o MVVM é um modelo de arquitetura de aplicativos que pode ser utilizado na plataforma .NET. Ele consistem em três camadas principais: *Model*, que é a camada de dados, *View* que é a acamada de interface e *Viewmodel* que é uma abstração de um modelo que possui interfacção com a interface da aplicação (CARDOSO, 2012).

2.2.2 Framework

Microsoft Silverlight é um *framework* de desenvolvimento para a plataforma .NET, o *framework* e muito poderoso e viza criar aplicaçoes com o modelo *Rich Interactive Applications* (RIA), Silverlight usa o *Extensible Application Markup Language* (XAML) para facilitar o desenvolvimento de interface do usuário como controles, animações, gráficos, *layout* (MICROSOFT, 2015, tradução nossa).

XAML é linguagem declarativa baseada no *Extensible Markup Language* (XML) utilizada para implementação de interfaces durante do desenvolvimento. Com

o XAML o desenvolvedor pode separar o telas da aplicação da camada de negócio. Além de simplificar a construção dos controles das telas (DURÃES, 2015).

2.2.3 SDK Windows Phone

O SDK do Windows Phone é composto por quatro componentes, sendo eles os seguintes (MÔNACO; CARMO, 2012):

a) **Visual Studio for Windows Phone:** versão do Visual Studio própria para o desenvolvimento para o Windows Phone;
b) **Windows Phone Emulator Resources:** emulador do Windows Phone onde e possível testar as aplicações simulando o ambiente do dispositivo;
c) **Silverlight Tools for Visual Studio:** *templates* para desenvolvimento de aplicações baseadas em *Silverlight*;
d) **XNA Game Studio 4.0:** *templates* para desenvolvimento de aplicações baseadas em XNA;
e) **Microsoft Expression Blend for Windows Phone:** ferramenta gráfica de auxílio ao *design* das interfaces das aplicações**;**

Estes componentes são integrados com o ambiente de desenvolvimento e tornam possível o desenvolvimento de aplicativos para o Windows Phone.

2.2.4 Emulador Windows Phone

Windows Phone Emulator (WPE) é uma aplicação desktop que emula um dispositivo Windows Phone. Ele fornece um ambiente virtualizado em que você pode depurar e testar aplicativos do Windows Phone sem um dispositivo físico. Ele também fornece um ambiente isolado para seus protótipos de aplicativos (MICROSOFT, 2015).

WPE foi construído para proporcionar um desempenho e layout compatível a um dispositivo real, assim o desenvolvedor pode testar seu aplicativo em diversas versões e resoluções disponíveis para Windows Phone. O emulador padrão no Visual Studio é emulador *Wide Video Graphics Array* (WVGA) de 512 MB, que já está incluso no SDK do Windows Phone, a figura 8 ilustra o *layout* do WPE (JIMÉNEZ; MENESES, 2013).

Figura 8 – Emulador Windows Phone

Fonte: Monaco (2015).

2.2.5 Linguagem C#

O C sharp ou mais conhecido como C# é uma linguagem orientada ao objeto derivada do C e C++, ela possibilita os programadores de forma rápida construir um ampla gama de aplicações para as plataformas Microsoft .NET (HICKSON, 2002).

Embora seja possível usar várias linguagens na plataforma .NET, hoje o C# é a recomendada para iniciar o desenvolvimento na plataforma, pois o C# oferece o mesmo poder que o C++ e a mesma facilidade de programação que o Visual Basic, além de ser a linguagem nativa para a nova plataforma da Microsoft, a linguagem C# também tem como objetivo permitir o desenvolvimento de qualquer tipo de aplicação: *Web service*, aplicação *Windows* convencional, aplicações para serem executadas num *palmtop* ou *handheld*, aplicações para *Internet* e aplicações *mobile* (LIMA, 2002).

3 PHONEGAP CROSS-PLATAFORMA FRAMEWORK

Nas primeiras versões o PhoneGap era utilizado apenas para criação de *templates* para o Xcodo e Eclipse, ele era propriedade da Nitobi Software, em 2011 com a compra da Nitobe pela Adobe o projeto começou a ser integrado com o Apache Cordova, a Adobe manteve o nome de PhoneGap, mas contudo desde a versão 1.x o PhoneGap era muito problemático até a o projeto começar a utilizar o *Command-Line Cnterface* (CLI) que permite os desenvolvedores criarem projetos multi-plataforma facilmente utilizando poucas linhas de comando. Em 2013 com a nova CLI ficou mais simples ainda a criação e desenvolvimento de projetos mult-plataforma a instalação de plug-in ficou mais fácil e menos problemáticas que antes (SHOTTS, 2014).

Basicamente o PhoneGap é um conjunto bibliotecas que permite aos desenvolvedores interagir diretamente com um dispositivo móvel através da utilização de JavaScript. Com o grande número de plataformas móveis é muito difícil e caro para criar múltiplas aplicações em Java, Objective-C, ou outras línguas nativas. Através da biblioteca PhoneGap, a maioria dos desenvolvedores web podem utilizar seu conhecimento existente de HTML, CSS e JavaScript para desenvolver aplicativos moveis cross-plataforma (MUNRO, 2012).

Esta bibliotecas específicas fornece recursos de comunicação que as tecnologias da web não têm, tais como acesso ao módulo Bluetooth e Wi-Fi para enviar e receber informações. Mas em qualquer caso, as aplicações construídas com esta tecnologia podem ser considerados como aplicações *webapps* híbrida (ESPADA, 2013).

O PhoneGap fornece aos desenvolvedores acesso ao recursos nativos do dispositivo móvel do usuário, ele também fornece as tecnologias necessárias para interagir com os sistemas externos localizado em outras redes. Isso significa que os usuários podem ser altamente móvel, sem perder o acesso a altamente valiosa e oportuna dados da empresa (SHOTTS, 2014).

Hoje o PhoneGap e compatível com os maiores SOs disponíveis no mercado, dentre eles Windows Phone, Android e IOS, pois seus aplicativos são construídas por tecnologia web aberto, ilustrado na figura 9 (PHONEGAP, 2015).

Figura 9 – Tecnologia PhoneGap

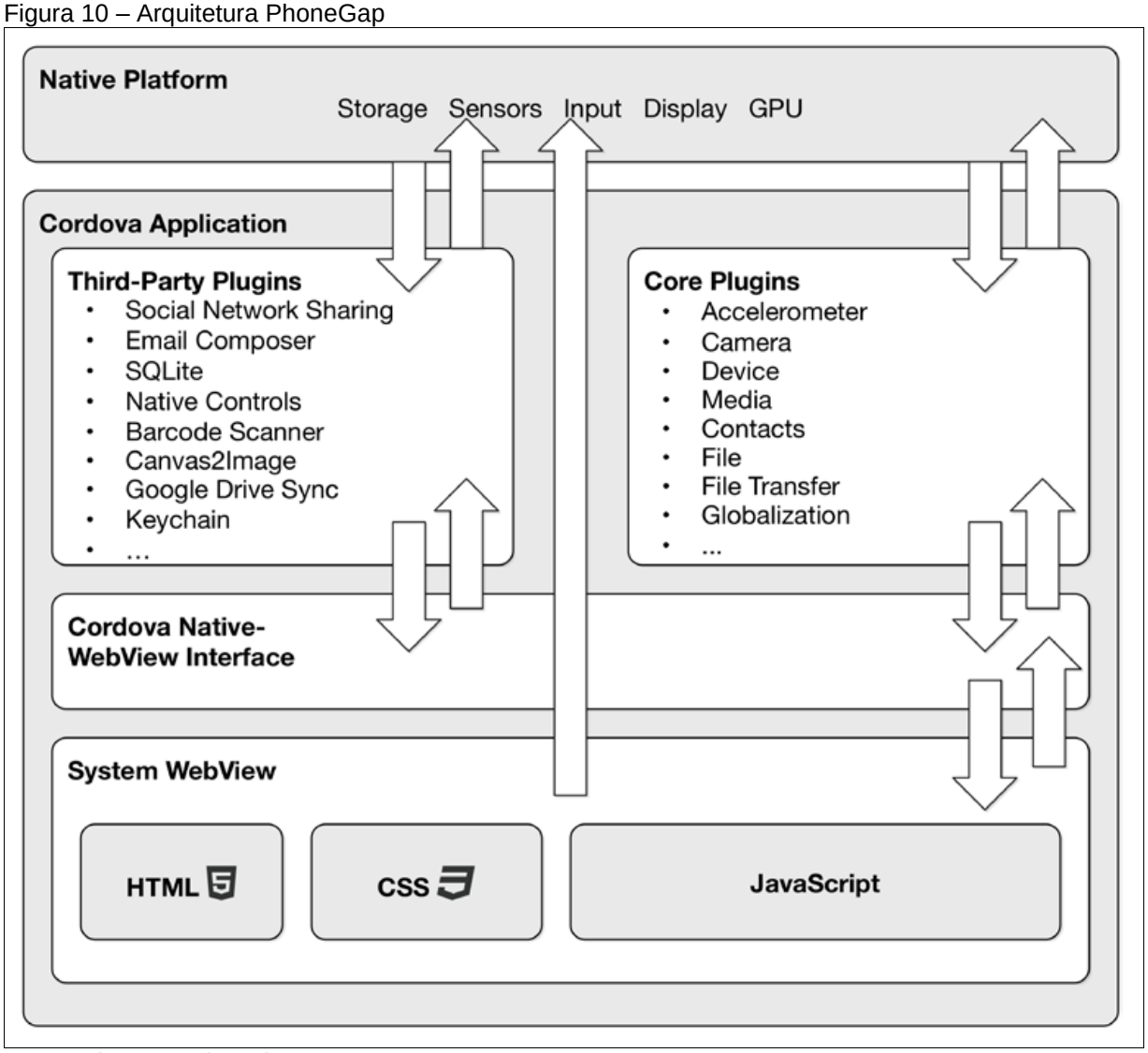
Fonte: Phonegap (2015).

3.1 ARQUITETURA PHONEGAP

A arquitetura do PhoneGap e baseada em uma mistura de programação Web e nativa, código WWW ou web é totalmente multi-plataforma e deve ser o mesmo em todas as plataformas. Sempre que necessário podem ser escrito trechos de códigos para dispositivos específicos, embora isso normalmente só é necessária quando utilizamos *plugins* de terceiros que são específicos para certas plataformas (MYER, 2012).

Conforme figura 10, o código web multi-plataforma é executado em cima de um código nativo específico para cada plataforma, o código nativo serve para iniciar o aplicativo e instanciar um único navegador de tela cheia sistema no qual o código *web* é executado, o código nativo também inicializa quaisquer *plugins* nativos, os mesmos podem ser *plugins* centrais fornecida pela equipe de Cordova, *plugins* de terceiros (PHONEGAP, 2015).

Figura 10 – Arquitetura PhoneGap



Fonte: Phonegap (2015).

Por causa deste trecho nativo gerado pelo framework torna a programação simples e objetiva usando linguagem Web (HTML, CSS JavaScript), permitindo todas as funcionalidades e qualidade de uma aplicação nativa.

3.2 LIMGUAGENS WEB

As principais linguagens Web são HTML, CSS e JavaScript, com essas três linguagem o desenvolvedor pode criar aplicações e Web sites totalmente responsivos e seguros (CARVALHO, 2001).

3.2.1 HTML

Hoje o HTML é a estrutura básica de todas as páginas da internet, assim quando mencionamos, ou utilizamos qualquer tecnologia WWW estamos falando de HTML, ele é derivado do Standard *Generalized Markup Language* (SGML), ele foi criado no início da década de 1990 com o objetivo de tornar possível o acesso e troca de informações e pesquisas entra universidades, mas em menos de 20 anos o HTML já era presente em todo mundo (SILVA, 2011).

Os documento HTML consistem basicamente por *tags*, palavras entre parênteses angulares (< e >), todo *tags* tem sua função especifica e são os comandos de formatação da linguagem, existem diversas as tags disponíveis, com elas é possível, por exemplo, criar parágrafos, cabeçalhos e blocos (*divs*) que contenham mais *tags,* entre outros comandos (SILVA, 2011).

Segundo Carvalho (2001) a estrutura básica de um documento HTML é a seguinte:

```
<!DOCTYPE HTML>
<html lang=”pt-br”>
        <head>
            <title></title>
                <meta charset=”UTF-8”>
        </head>
        <body>
        </body>
</html>
```

3.2.2 CSS3

O CSS foi lançado em de 1998 com o intuito de facilitar a estilização das páginas HTML, hoje ele é uma linguagem de estilização responsável por definir a apresentação visual de um praticamente todas as páginas HTML disponíveis no mundo, adicionando elementos como cores, sombras e animações (SILVA, 2012).

O CSS interage diretamente com o HTML, esse processo é feito através dos seletores que são associados as *tags* do HTML, por meio dos seletores é possível referenciar trechos específicos do documento HTML, estilizando-os individualmente ou coletivamente através do atributo *class* do HTML. O CSS possui

seletores pré-definidos como *div*, *span*, *table* e *p*, os seletores class devem ser utilizados por elementos HTML únicos (MAZZA, 2013).

3.2.3 JavaScript

Criado em 1995, o JavaScript é uma linguagem de programação que pode ser adicionado a uma página HTML para torná-la mais interativa e conveniente para o usuário, assim tornando a experiência do usuário agradável e menos cansativa. Ela é orientada a objetos, de tipagem fraca e dinâmica, com funções de primeira classe e uma das poucas linguagens *client-side* disponíveis (SILVA, 2010).

Por muito tempo o JavaScript possuiu uma má reputação por causa da baixa performance, linguagem peculiar que acompanhava bugs misteriosos e as diversas dependências por causa do *Document Object Model* (DOM) eram um dos diversos motivos que impediam o crescimento desta linguagem, mas 10 anos se passaram e com o surgimento do *Asynchronous Javascript and XML* (AJAX), o JavaScript passou a ser visto com outros olhos e a partir deste momento diversas ferramentas e API começaram a ser desenvolvidas para ele (DEITEL, 2003).

Algumas das API que elevaram o status do JavaScript tornando-o essencial no dias atuais são o AJAX, *JQuery*, *ExtJS*, *Node*.js, GWT, *MooTools* e *RaphaëlJS*. Ferramentas como o *FireBug* e *JSLint* também ajudaram em sua popularização entre os desenvolvedores e as páginas da HTML da internet (SILVA, 2010).

4 METRICA DE SOFTWARE

As métricas de *software* são discutidas há mais de 20 anos dentro engenharia de *software*, pesquisas realizadas em empresas de software indicam que mais da metade de grandes projetos de software se deparam com algum tipo de problema, excesso de custo ou prazo ou algum fracasso na execução quando implantado (CHOU, 2015).

As métricas de software têm como objetivo especificar as funções de coleta de dados de avaliação e desempenho, custo, prevenindo problemas na exclusão do projeto ou na implantação do mesmo. Assim quando vemos aplicação de métricas temos que ter em mente que se trata de avaliação de dados números quantitativos que irão auxiliar na decisão e mostrar indicadores do estado atual do projeto ou software (PRESSMAN, 1995).

Uma métrica de software é qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada e quando coletadas essas medições, as questões em relação ao software poderão ser respondidas e confirmações poderão ser feitas, de que as melhorias do software alcançaram, ou não, a meta desejada (SOMMERVILLE, 2003).

Métricas de *Software* é uma medição de propriedades ou características de um determinado processo ou recurso, utilizadas para determinar o custo previsto e desempenho do software que são divididas em várias categorias (GOMES, 2015).

As métricas de software tem como o principal objetivo de especificar métodos de coleta de dados e avaliação de custo e desempenho, assim toda a equipe de análise envolvida no desenvolvimento pode verificar os dados obtidos e compará-los com os dados do histórico de projetos atribuindo esta responsabilidade a todos os analistas. Quando se fala de métricas deve-se ter em mente que se trata de dados, números quantitativos que irão mostrar a viabilidade e o custo do *software* a ser desenvolvido.

Conforme Chou (2015) a aplicação das métricas de *software* devem ser aplicada em todas as fases do ciclo de vida do projeto, e não somente na fase de desenvolvimento, isso fará com que a própria equipe do projeto fique mais confiante diante dos resultados obtidos e melhore cada vez mais o processo pelo qual o projeto é desenvolvido, evoluindo sempre para a qualidade do processo e do produto.

A medição tem um papel importante dentro da engenharia de software, especialmente quando se trada da gerência de projetos de software, independente da metodologia utilizada e essencial que os gerentes de projeto analisem os dados obtidos pelos engenheiros de software e equipe de analistas (SOMMERVILLE, 2003).

As métricas são aplicadas em três fases: coleta de dados, cálculo dos dados e análise dos dados, é importante saber escolher a metodologia que melhor se encaixa no projeto, trazendo resultados mais precisos, sejam eles bons ou ruins. Tendo então realizada a medição, poderão ser feitas estimativas de custos e prazos de término do projeto ou entrega do produto final (GOMES, 2015).

Para um melhor entendimento sobre métrica de software e necessário saber que as definições de cada parte de uma métrica de software (VASCONSELO, 2005).

a) **Medida**: Fornece uma indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo de um produto ou processo.
b) **Medição**: Ato de determinação de uma medida.
c) **Métrica**: Medida quantitativa do grau em que um sistema se encontra em relação a um determinado atributo.
d) **Indicadores**: Métrica ou combinação de métricas que fornece uma compreensão de um processo/projeto/produto

As métricas podem ser categorizadas de maneiras diferentes, tais como métricas diretas e indiretas, ou métricas orientadas a tamanho, ou funções, entre outras conforme tabela 1.

Tabela 1 – Métricas de testes

Nome	Técnica	Exemplos
Métricas diretas	Medidas obtidas a partir de outras métricas	Ex.: complexidade, eficiência, confiabilidade, facilidade de manutenção
Métricas indiretas	Medidas obtidas a partir de outras métricas	Ex.: complexidade, eficiência, confiabilidade, facilidade de manutenção
Métricas orientadas a tamanho	São medidas diretas do tamanho dos artefatos de software associados ao processo por meio do qual o software é desenvolvido.	Ex.: esforço, custo, no. KLOC, no. páginas de documentação, no. erros
Métricas orientadas por função	Consiste em um método para medição de software do ponto de vista do usuário, determinando de forma consistente o tamanho e a complexidade de um software.	
Métricas de produtividade	Concentram-se na saída do processo de engenharia de software.	Ex.: no. de casos de uso/iteração.
Métricas de qualidade	Oferecem uma indicação de quanto o software se adequa às exigências implícitas e explícitas do cliente.	Ex.: erros/fase
Métricas técnicas	Concentram-se nas características do software e não no processo por meio do qual o software foi desenvolvido.	Ex.: complexidade lógica e grau de manutenibilidade

Fonte: Chou (2015).

4.1 MÉTRICAS DIRETAS E INDIRETAS

As métricas diretas são aquelas onde os atributos são observados custo, esforço, quantidade de linhas de código produzidas, total de defeitos registrados e as métricas indiretas são aquelas obtidas a partir de outras métricas eficiência, confiabilidade, qualidade, funcionalidade (GUARIZZO, 2008).

Definir custo e o esforço utilizado para construir um software, também como o número de linhas de código produzido, tempo de desenvolvimento e outras medidas diretas são mais utilizadas por serem mais fáceis de serem quantificadas, porem quando falamos de qualidade, funcionalidade, eficiência e capacidade de

manutenção, que são avaliadas indiretamente e apresentam uma maior dificuldade (GOMES, 2015).

4.2 MÉTRICAS ORIENTADAS A TAMANHO E FUNÇÃO

Métricas de software orientadas ao tamanho são medidas diretas do software e do processo por meio do qual ele é desenvolvido, elas são utilizadas para mesurar o custo por *Thousand* Lines of Code (KLOC), a partir de dados brutos obtidos durante o desenvolvimento do Software (GOMES, 2015).

As métricas orientadas a tamanho, consideram o tamanho do software produzido em linhas de código, também podem ser aplicadas em todas as atividades da engenharia análise, projeto, código, teste, e poderão medir produtividade em KLOC/pessoa-mês, qualidade defeitos/KLOC. Já a métrica orientada à função em vez de contar linhas de código são focada em analisar e efetuar medidas na funcionalidade do software (SOMMERVILLE, 2003).

A análise de pontos por função e avaliada inteiramente pelo o usuários e como os mesmos veem o produto final e os resultados obtidos pelo software produzido. Ela se baseia parcialmente em dados subjetivos, implicando a organização estabelecer um plano de implantação da sistemática da medição, definindo padrões para contagem (CHOU, 2015).

4.3 MÉTRICAS DE PRODUTO E PRODUTIVIDADE

As métricas de produto e produtividade se analisam as características do próprio software, elas se dividem em duas classes estáticas e dinâmicas, conforme Claro (2008):

a) Métricas estáticas, que são coletadas por medições feitas das representações do sistema, como projeto, programa ou documentação.
b) Métricas dinâmicas, que são coletadas por medições feitas de um programa em execução.

Métricas de produtividade se concentram na saída do processo de engenharia de software e analise e quantificam o em número de casos de uso, iteração.

4.4 MÉTRICAS DE QUALIDADE E MÉTRICAS TÉCNICAS

As métricas de qualidade, oferecem uma indicação de quanto o software se adequa as exigências efetuadas pelos usuários e clientes, seguindo as medidas estabelecidas. Segundo Pressman (1995) existem vários tipos de medidas de qualidade de software, as principais pode ser representas por:

a) Corretitude é a medida efetuada para verificar se todas as funções do *software* produzido operam corretamente, caso que este número for baixo o *software* oferecera uma utilidade irrelevante aos usuários e cliente.
b) Manutenibilidade é a facilidade com que um programa pode ser corrigido se um erro for encontrado, adaptado se o seu ambiente se modificar ou ampliado se o cliente desejar inclusões e alterações nos requisitos funcionais. Não existe nenhuma forma de se medir a manutenibilidade diretamente, deve-se usar medidas indiretas, a manutenção de software é responsável por mais esforço do que qualquer outra atividade de engenharia de software.
c) Integridade é a segurança que o *software* tem contra invasões e acessos indevidos de usuários sem permissão, essa medida de software vem tornando-se cada vez mais importante na era dos hackers e dos vírus. Esse atributo mede a capacidade que um sistema tem de se suportar ataques a sua integridade, ataques podem ser feitos a todos os três componentes do software: programas, dados e documentos.
d) Usabilidade é o quanto o *software* produzido é amigável ao usuário se ele não for amigável estará destinado ao fracasso, mesmo que as funções que eles executem sejam valiosas.

Para análise de desempenho de software segundo Teles (2005), é necessário definir o fluxo do software, após isso estimar o tempo e recursos consumidos pelo Software, podendo descobrir potenciais pontos de gargalo e predizer o desempenho do sistema.

5 TRABALHOS CORRELATOS

Neste capitulo são apresentados trabalhos que possuam conteúdos semelhantes ou que tenham alguma relação com este projeto de pesquisa, disponibilizando maiores informações sobre a proposta apresentada.

5.1 DESENVOLVIMENTO DE CROSS-PLATFORM MOBILE APPS UTILIZANDO O TITANIUM MOBILE

Com o crescente desenvolvimento da telefonia móvel, desde que foi criada em 1973 por Martin Cooper, trouxe diversos benefícios para todo o mundo, mas com o surgimento de várias marcas de dispositivos moveis e a incompatibilidade entra SO, surgi-o os framework de desenvolvimento cross-plataforma, com estes frameworks é possível desenvolver aplicativos compatíveis com mais de uma plataforma.

Este trabalho tem como objetivo apresentar o Titanium Mobile, um framework utilizado para criação de cross-platform mobile apps. Serão expostos os pontos positivos e negativos deste tipo de aplicativo e também será realizado um estudo de caso para demonstrar a eficiência deste framework.

Para a realização desta pesquisa foi abordado na pratica de como a IDE Titanium Mobile funciona para cada plataforma, o autor efetuou o estudo de cada funcionalidade e de como ela seria desenvolvida cross-plataforma com tecnologia Web do Titanium Mobile, assim foi desenvolvido um aplicativo e executado na plataforma Android e IOs.

5.2 ESTUDO DE FRAMEWORKS MULTIPLATAFORMA PARA DESENVOLVIMENTO DE APLICAÇÕES MOBILE HÍBRIDAS

Com o expressivo crescimento das tecnologias móveis, experimentado ao longo dos últimos anos, observa-se que a diversidade de tais equipamentos aumentou muito. Consequentemente o número de plataformas e ambientes/linguagens de programação para desenvolvimento de aplicações também foi incrementado. Tal situação encarece e complexifica o desenvolvimento de uma aplicação que venha a atender a todos dispositivos, pois é necessário conhecimentos específicos de cada plataforma, e também que seja desenvolvida

uma aplicação diferente para cada plataforma, neste contexto surgem as aplicações híbridas, em que se desenvolve apenas uma aplicação, e esta pode ser utilizada em vários dispositivos com diferentes sistemas operacionais.

Para a realização desta pesquisa e melhor compreender o funcionamento dos frameworks para desenvolvimento e das aplicações híbridas, foi desenvolvido um estudo de caso. Neste estudo, uma aplicação (diário de classe escolar) foi desenvolvida, utilizando-se de frameworks para aplicações híbridas e disponibilizada em diferentes plataformas. Entende-se que além de servir como meio de análise do estudo, a aplicação também resulta em um software útil em meio acadêmico.

5.3 ESTUDO SOBRE O PHONEGAP E SEU DESEMPENHO ANTE A LINGUAGEM NATIVA DO ANDROID

Com a venda de smartphones batendo recordes de trimestre em trimestre, o acesso da população aos aplicativos e sites móveis se torna cada vez maior. Porém, entrar para este mercado não é tão simples, além das conhecidas barreiras na área de desenvolvimento, há diversas particularidades distintas de configuração, tanto do hardware quanto do software por parte dos fabricantes. Os *frameworks* multiplataforma surgem como uma alternativa de desenvolvimento de um único código que funcione em diversos sistemas operacionais. Com aproveitamento de código, custos reduzidos e uma facilidade maior de atualizações futuras, esses *frameworks* se tornam uma opção atraente para o desenvolvimento.

Para a realização desta pesquisa foi analisado o comportamento e o desempenho do PhoneGap, *framework* multiplataforma mais utilizado no sistema operacional com maior fatia de mercado, o Android. Também apresenta um comparativo com a linguagem Java (nativa do Android), mostrando as vantagens, perdas e o perfil de aplicação mais adequado para essa estrutura de *framework*.

5.4 ANÁLISE COMPARATIVA DE FERRAMENTAS DE DESENVOLVIMENTO DE APLICATIVOS MÓVEIS MULTIPLATAFORMA

Este trabalho apresenta uma análise comparativa entre as ferramentas de desenvolvimento móvel multiplataforma Delphi XE5, Xamarin e PhoneGap. Os critérios para avaliação foram determinados com base na norma NBR ISO/IEC

25000. Para permitir a comparação foram implementados dois aplicativos utilizando as ferramentas, sendo o primeiro simples e o segundo utilizando mais recursos do dispositivo, a fim de medir a diferença de benefícios que cada ferramenta oferece.

Nesta pesquisa foram estabelecido critérios da avaliação, estes foram correlacionados com as principais funcionalidades de 3 ferramentas de desenvolvimento, visando qualificá-las e compará-las, para isso foi utilizado um questionário produzido pelo autor e aplicado a ao desenvolvedores que utilizam as 3 ferramentas.

5.5 ANÁLISE DE PERFORMANCE DE FRAMEWORKS DE DESENVOLVIMENTO MOBILE MULTIPLATAFORMA

A intenção deste trabalho é estudar diferentes frameworks para desenvolvimento mobile e realizar um teste de performance com os mesmos. Entende-se que o fato de os *frameworks* utilizarem diferentes linguagens e tecnologias é natural que se comportem de forma diferenciada. (WAHLBRINCK; BONIATI, 2014)

Neste trabalho foram utilizados as tecnologias PhoneGap, AppGyver e Corona SDK que são *frameworks* de desenvolvimento de aplicativos cross-plataforma, foi desenvolvido um aplicativo com um algoritmo especifico e efetuado testes comparativos entras os *frameworks*. (WAHLBRINCK; BONIATI, 2014)

As comparações neste trabalho foram feitas no tempo de exclusão e compatibilidades com os SOs no mercado atual, o algoritmo utilizado calcula números primos em um determinado intervalo de números. (WAHLBRINCK; BONIATI, 2014)

6 ANÁLISE COMPARATIVA DE APLICAÇÕES MOBILE NATIVA E APLICAÇÕES MOBILE CROSS-PLATAFORMA

O desenvolvimento pratico deste trabalho objetivou a análise comparativa de desempenho e custo de desenvolvimento de aplicativos nativos comparando com aplicativos cross-plataforma Mobile utilizando Métricas orientada ao tamanha para poder definir os custos e teste de sobrecarga para os testes de desempenho. Os conhecimentos adquiridos durante o levantamento bibliográfico referente a dispositivos moveis, as plataformas Android e Windows Phone aliados aos estudos das métricas de teste, propiciaram subsídios para a quantificação do desempenho e do custo em KLOC de desenvolvimento.

O aplicativo do sistema operacional Android nativo foi desenvolvido na IDE do Android Studio, já para o sistema operacional Windows Phone nativo foi desenvolvido na IDE Visual Studio 2012, o aplicativo cross-plataforma foi desenvolvido na IDE PhoneGap.

As três aplicações tem a mesma função, elas efetuaram integração em um WebService desenvolvido em Java com banco MySql, tem um processo sobrecarga de processo, para que fosse possível efetuar análise comparativa de desempenho entre as aplicações cross-plataforma e nativa, contudo também foi analisado e tempo gasto no desenvolvimento das telas e do acesso ao Hardware do dispositivo afim de estipular o custo necessário para desenvolver cada funcionalidade, as aplicações acessaram a câmera e o GPS afim de estipular desempenho do software ao acessar o hardware.

6.1 METODOLOGIA

Para a realização do presente trabalho de pesquisa foram aplicadas as seguintes metodologias: levantamento bibliográfico; estudo sobre o sistema operacional Android, estudo sobre o sistema operacional Windows Phone, estudo sobre ambiente de desenvolvimento cross-plataforma, as liguagem Java, C#, HTML5, CSS3, JavaScript.

Com tudo, além dos itens citados, para o desenvolvimento do protótipo de aplicativo foram efetuadas pesquisas para atender os requisitos necessários; levantamento das metodologias de análise comparativa de desempenho e

levantamento do custo de desenvolvimento;

6.1.1 Levantamento bibliográfico

Os objetos de estudo abordados no levantamento bibliográfico podem ser organizados da seguinte forma: Dispositivos moveis, SO Android, SO Windows Phone, PhoneGap cross-plataforma, Métrica de teste. Os temas foram pesquisados em teses de doutorado, dissertações de mestrado, trabalhas de conclusão de curso, artigos científicos, livros e sites oficiais da tecnologia. Relacionar referencias que envolvessem ao máximo dispositivos moveis e análise de desempenho e custo de desenvolvimento.

6.1.2 Estudo sobre o sistema operacional Android

Foram efetuadas pesquisas sobre o sistema operacional Android, o qual será utilizado para o desenvolvimento do aplicativo nativo, que se mostrou ser uma plataforma com alta reputação. Deste modo foram obtidos conhecimento do funcionamento da plataforma para que fosse feita uma implementação adequada e que respeitassem os pré-requisitos da metodologia.

6.1.3 Estudo sobre o sistema operacional Windows Phone

Foram efetuadas pesquisas sobre o sistema operacional Windows Phone, o qual será utilizado para desenvolvimento do aplicativo nativo, que se mostrou ser uma plataforma dinâmica com alta qualidade gráfica. Deste modo foram obtidos conhecimento do funcionamento da plataforma para que fosse feita uma implementação adequada e que respeitassem os pré-requisitos da metodologia.

6.1.4 Estudo sobre o ambiente de desenvolvimento cross-plataforma PhoneGap.

Foram efetuadas pesquisas sobre o ambiente de desenvolvimento cross-plataforma PhoneGap, o qual foi utilizado para desenvolvimento do aplicativo cross-plataforma, que se mostrou ser uma plataforma dinâmica com alta qualidade de

desempenho e fácil desenvolvimento. Deste modo foram obtidos conhecimentos do funcionamento da tecnologia para que fosse feita uma implementação adequada e que respeitassem os pré-requisitos da metodologia.

6.1.5 Estudo sobre as linguagens Java, C#, HTML5, CSS3, JavaScript, SQL

De acordo com os conhecimentos obtidos na UNESC sobre as linguagens Java, C#, HTML5, CSS3, JavaScript e a pesquisa sobre elas foi realizado o desenvolvimento dos aplicativos nativos e cross-plataforma conforme as necessidade encontradas. Assim foi possível ter um controle completo dos testes comparativos de desempenho e do custo de desenvolvimento.

6.1.6 Desenvolvimentos das aplicações mobile nativa e aplicações mobile cross-plataforma de um aplicativo de Achados e Perdidos

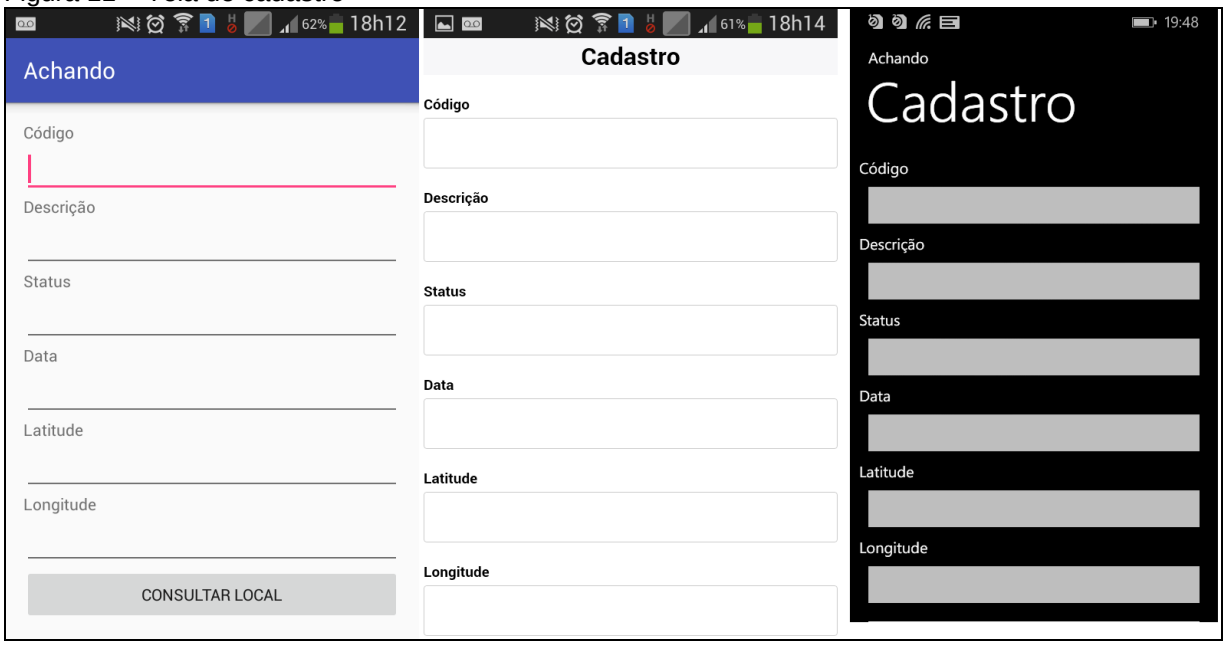
Para a análise comparativa de custo de desenvolvimento por KLOC foi desenvolvido um aplicativo de achados e perdidos, conforme os Correios (2016) são perdidos em 10 dias mais de 15 mil documentos isto somente documentos, estes dados mostram a relevância de se desenvolver um aplicativo para podermos cadastrar e consultar os itens achados e perdidos , assim podendo os encontrar mais facilmente, contudo ainda avaliarmos as tecnologias nativas e cross-plataforma, a análise dos aplicativos foi efetuada utilizado a Métrica orientada ao tamanho, explicado no capitulo 6, este desenvolvimento e quantificação de tempo foi dividida na seguinte lista de etapas.

6.1.6.1 Desenvolvimento de Tela de cadastro

Nesta etapa foi desenvolvida uma tela de cadastro que contém os campos: Código, Descrição, Data, Latitude, Longitude e Foto do Item perdido. Esta tela foi desenvolvida Nativa em Windows Phone e Android e cross-plataforma utilizando o PhoneGap, nas três aplicações foram utilizados os componentes de tela nativos de cada linguagem, para o Windows Phone e Android foram utilizado o XML e para o PhoneGap o HTML, esta tela foi desenvolvida para cadastra itens de achados e perdidos com intuito de ter um fácil uso, assim conforme a figura 11

abaixo as telas tem uma aparência diferente uma da outra, pois para cada tecnologia há um padrão de componentes diferente.

Figura 11 – Tela de cadastro



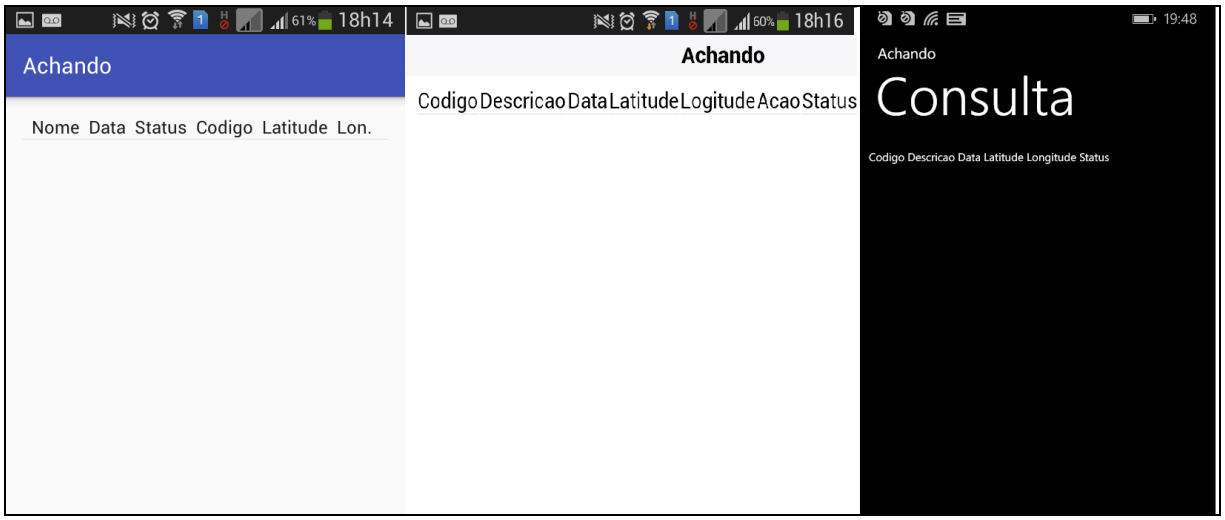
Fonte: Do autor (2016)

6.1.6.2 Desenvolvimento das consultas no Web Service

Nesta etapa foi desenvolvido uma tela que efetua uma consulta em um Web Service Java, após isso os dados retornados são mostrados na tela em uma lista que contém: Código, Descrição, Data, Latitude, Longitude, esta informação e mostrada em uma grid padrão do próprio aplicativo.

Foi utilizado o *HTTPClient* para efetuar as consultas pelo aplicativo Android e *WebClient* no Windows Phone já no PhoneGap foi utilizado *jQuery*, esse bibliotecas foram utilizada para construir uma consulta via GET no *WebService,* esta consulta retorna um *Json* que os aplicativos percorrem e mostram na tela, na figura 12 podemos ver a tela nas versões Windows Phone nativo e cross-plataforma e Android Nativo e cross-plataforma.

Figura 12 – Tela de consulta



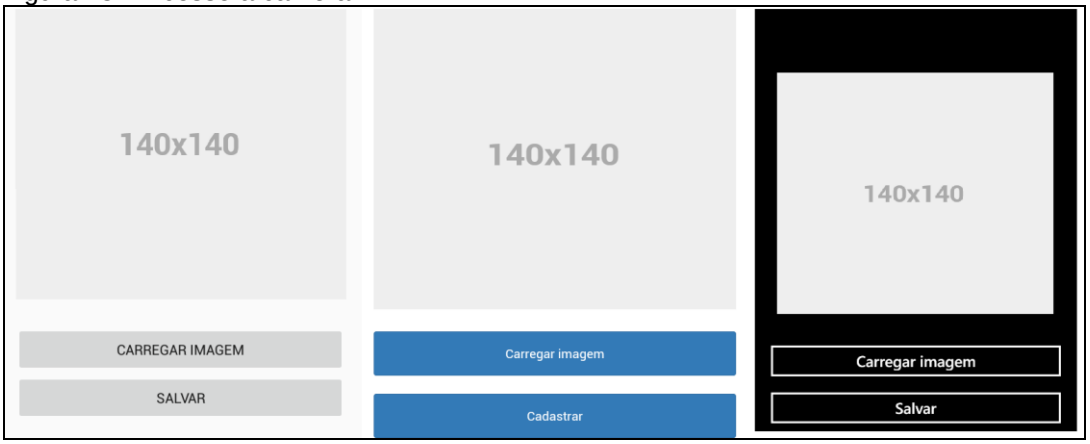
Fonte: Do autor (2016)

6.1.6.3 Desenvolvimento do acesso a câmera do dispositivo

Nesta etapa foi desenvolvido o acesso a câmera do dispositivo, esta funcionalidade ficou localizada na tela de cadastro de item perdido do aplicativo. Esta funcionalidade consiste em acessar a câmera física do dispositivo capturar uma foto e mostrar na tela do aplicativo.

Para os três tecnologias foi construído um método que efetua a chamada da câmera, e outro que fica esperando o retorno da mesma, no Android foi inicializado uma "*activity*" com um "*intente*" passando por parâmetro a propriedade "*MediaStore.ACTION_IMAGE_CAPTURE*", já para o WindowsPhone foi efetuado uma chamada no "*CameraCaptureTask*", para o PhoneGap foi efetuado uma chamada no método "*navigator.camera.getPicture*", após isso no método de retorno a imagem retornada e convertida em um *bitmap* e apresentada na tela, conforme figura 13 esta funcionalidade está relacionada diretamente ao botão "Carregar imagem" do aplicativo.

Figura 13 – Acesso a câmera



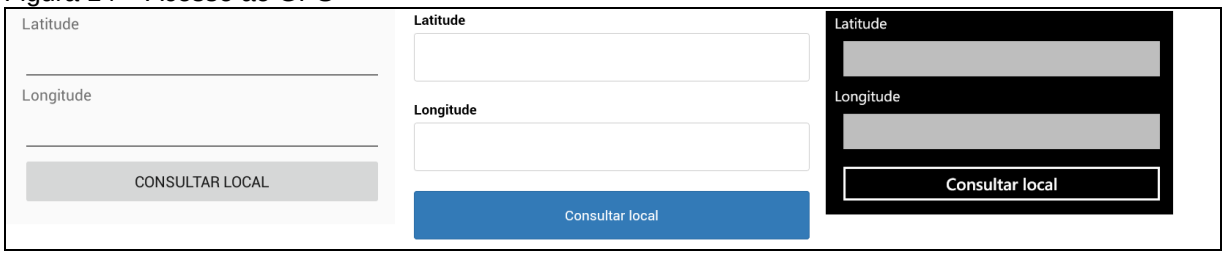
Fonte: Do autor (2016)

6.1.6.4 Desenvolvimento ao acesso GPS

Nesta etapa foi desenvolvido o acesso ao GPS do dispositivo, esta funcionalidade está localizada na tela de “Cadastro de item perdido”, ela consiste de acessão o sistema de localização e retornar a latitude e longitude para a tela do aplicativo.

Para o desenvolvimento desta função em Android foi utilizado a biblioteca “*LocationManeger”*, para o Windows Phone foi utilizado a biblioteca “*Geoposition”* e para o PhoneGap o método “*navigator.geolocatio.getCurrentPositon*”, estes métodos tem como retorno a latitude e longitude e outras informações sobre a localização do dispositivo, conforme figura 15, esta funcionalidade está relacionada diretamente ao botão “Consultar local” do aplicativo, e o retorno dela e mostrado nos campos de “Latitude” e “Longitude”.

Figura 14 – Acesso ao GPS

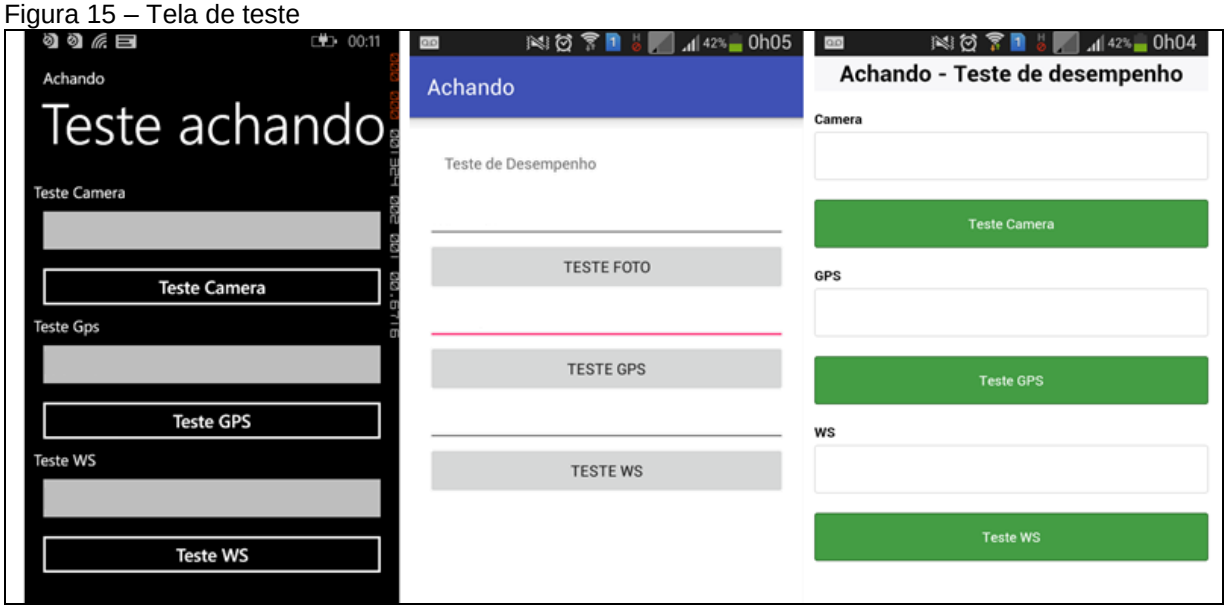


Fonte: Do autor (2016)

6.1.7 Desenvolvimento das aplicações mobile nativa e mobile cross-plataforma para testes de desempenho

Para a análise comparativa desempenho foi desenvolvido um tela junto ao aplicativo de achados e perdidos, que efetuara os teste de desempenho por carga, explicado no capitulo 6, está tela efetua três testes, caga no acesso ao GPS e Web Service, e acesso a câmera, Conforme figura 16 para cada teste foi criado um botão e executara a função de teste desenvolvida, a função de teste do GPS e Web Service foram construídas para executar a consulta 100 vezes e retornar a média de tempo de exclusão em tela, o teste de acesso a câmera será executado uma única vez pois este processo necessita de interação do usuário para confirmar a imagem.

Figura 15 – Tela de teste



Fonte: Do autor (2016)

Para os testes de desempenho do aplicativo e comparações entre os aplicativos Android nativo e cross-plataforma foi utilizado o dispositivo Samsung Galaxy S3 Slim G3812, com processador Quad core de 1.2gz e 1 GB de memória RAM, com Android 4.2.2.

Figura 16 - Samsung Galay S3 Slim

Fonte: Google(2016)

Para os testes de desempenho do aplicativo e comparações entre os aplicativos Windows Phone nativo e cross-plataforma foi utilizado o dispositivo Nokia

Lumia 530, com processador Quad core de 1.2gz e 512 GB de memória RAM, com Windows Phone 8.1.

Figura 17 – Nokie Lumia 530

Fonte: Microsoft (2016)

6.1.8 Aplicação das métricas de análise comparativa entre as aplicações desenvolvidas

A aplicação da análise comparativa foi feita em duas etapas comparados somente aplicativos testados no mesmo dispositivo, ou seja, entre aplicativo nativo Windows phone comparado ao aplicativo Cross-plataforma compilado para o Windows Phone, mesma comparação foi utilizada para os aplicativos da plataforma Android.

Na primeira etapa foi feita a aplicação na métrica orientada ao tamanho, conforme capitulo 6, a métrica orientada ao tamanho consistem de mensurar o tempo gasto para desenvolver e a quantidade de linhas de código gerada, o desenvolvimento foi feito em cinco semanas, utilizando somente três dias por semana para o desenvolvimento, antes de cada desenvolvimento foi efetuado a leitura da documentação do desenvolvedor e estudado a estrutura de cada funcionalidade desenvolvida, após a conclusão do desenvolvimento foi quantificado as linhas de código foi necessário para desenvolver cada funcionalidade, esta etapa gerou a tabela 2 para os dados obtidos a partir do desenvolvimento do aplicativo nativo em WindowsPhone.

Tabela 2 – Dados Windows Phone

Windows Phone			
Funcionalidade	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Tela de cadastro	51	46	1,108695652
Acesso a Câmera	17	32	0,53125
Acesso ao GPS	23	44	0,522727273
Comunicação WS	33	59	0,559322034
TOTAL	**124**	**181**	**2,721994959**

Fonte: Do autor (2016)

A tabela 3 apresenta os dados obtidos no desenvolvimento do aplicativo nativo em Android e já efetuado os cálculos de KLOC para cada funcionalidade utilizando a Metrica Orientada ao Tamanho.

Tabela 3 – Dados Android

Funcionalidade	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Tela de cadastro	91	55	1,654545455
Acesso a Câmera	38	68	0,558823529
Acesso ao GPS	19	37	0,513513514
Comunicação WS	31	58	0,534482759
TOTAL	**179**	**218**	**3,261365256**

Fonte: Do autor (2016)

A tabela 4 apresenta os dados obtido no desenvolvimento do aplicativo cross-plataforma em PhoneGap e já efetuado os cálculos de KLOC para cada funcionalidade utilizando a Métrica Orientada ao Tamanho.

Tabela 4 – Dados PhoneGap

Funcionalidade	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Tela de cadastro	69	59	1,169491525
Acesso a Câmera	12	23	0,52173913
Acesso ao GPS	13	25	0,52
Comunicação WS	17	16	1,0625
TOTAL	**111**	**123**	**3,273730656**

Fonte: Do autor (2016)

Na segunda etapa foi efetuado a sobrecarga dos métodos pré-definidos para poder estipular o desempenho de cada funcionalidade, os testes foram

efetuados duas vezes cada dispositivo, uma vez com a versão nativa e uma com a cross-plataforma PhoneGap, estes testes foram efetuados em dispositivos físicos de propriedade do próprio acadêmico, também foram efetuados em ambiente controlado utilizando uma rede de internet ADSL de 50 *gigabyte* (50GB somente nominal) , estes testes obtiveram os dados das figuras 12,12.

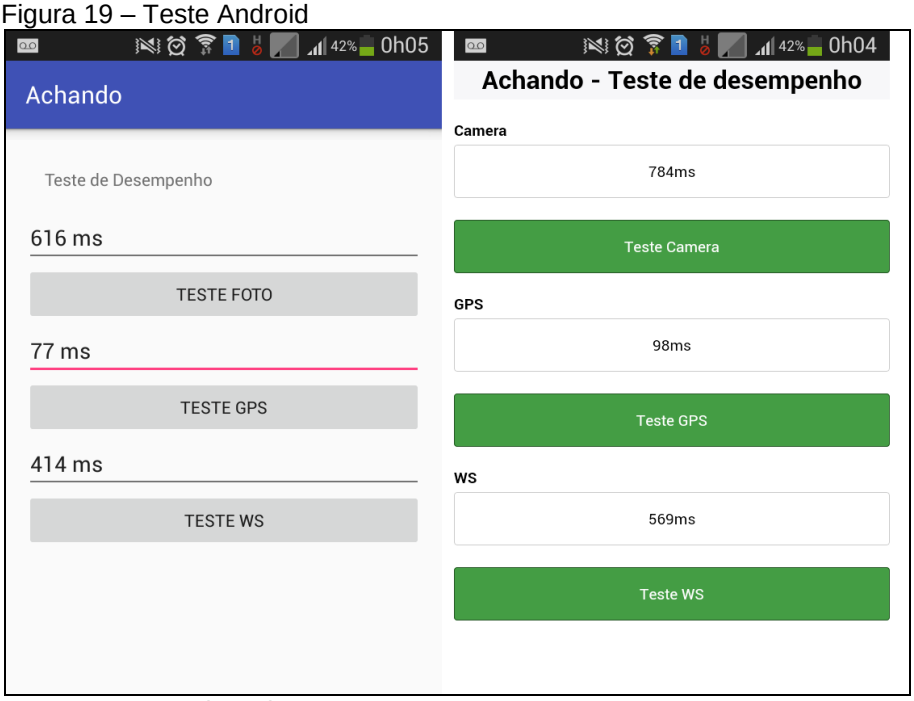
Na figura 18 apresenta os dados obtidos dos testes de sobrecarga efetuados nos aplicativos nativos e cross-plataforma PhoneGap desenvolvidos e instalados no dispositivo Nokia Lumia 530 com o Windows Phone 8.1.

Figura 18 – Teste Windows Phone

Fonte: Do autor (2016)

Na figura 19 apresenta os dados obtidos dos testes de sobrecargas efetuados no aplicativo nativo e cross-plataforma PhoneGap desenvolvido e instalados no dispositivo Samsung Galaxy S3 Slim com o Android 4.2.2.

Figura 19 – Teste Android



Fonte: Do autor (2016)

6.2 RESULTADOS OBTIDOS

Ao terminar a análise comparativa dos aplicativos de achados e perdidos desenvolvidos obtivemos os seguintes resultados da comparação dos resultados da Métrica Orientada ao Tamanho e dos testes de Sobrecarga.

6.2.1 Analise comparativa utilizando Métrica Orientada ao Tamanho

Considerando os dados obtidos com a métrica orientada ao tamanho identificou-se que:

a) para a construção da tela de cadastro conforme a tabela 5, foi necessário 51 linha de código fonte e 46 minutos de desenvolvimento para o Windows Phone , 91 linhas de código fonte e 55 minutos para o Android e 69 linhas de código fonte e 59 minutos para o PhoneGap, o código fonte do Android mostrou-se mais longo do que os demais e um tempo de desenvolvimento semelhante, a construção utilizando o PhoneGap cross-plataforma obteve um KLOC de 1,1694915 linha/min, comparando este resultado com o KLOC da construção nativa o PhoneGap apresentou-se superior somente com o Windows Phone nativo com 1,108695652 linha/min, já com Android nativo foi obtido um KLOC de 1,654545

linha/min sendo superior ao PhoneGap cross-plataforma e o Windows Phone;

Tabela 5 – Comparação Tela Cadastro

Aplicativo	Quantidade de linhas	Tempo de desenvolvimento em minutos	KLOC
Windows Phone Nativo	51	46	1,108695652
Android Nativo	91	55	1,654545455
PhoneGap	69	59	1,169491525

Fonte: Do autor (2016)

b) para a construção do Acesso a câmera física do dispositivo conforme a tabela 6, foi necessário 17 linhas de código fonte e 32 minutos para o Windows Phone Nativo, 38 linhas de código fonte e 68 minutos para o Android nativo e para o PhoneGap cross-plataforma foi necessário 12 linhas de código fonte e 23 minutos de desenvolvimento, o acesso a câmera em Android mostrou-se maior que os demais mas mais simples de desenvolver, a construção desta funcionalidade utilizando o PhoneGap cross-plataforma com o KLOC de 0,5217391 linha/min teve um custo maior comparado ao Android com o KLOC de 0,558824 linha/min e Windows Phone com o KLOC de 0,53125 linha/min, neste comparação os aplicativos nativos mostraram-se melhor que o cross-plataforma.

Tabela 6 – Comparação Acesso a Câmera

Aplicativo	Quantidade de linhas	Tempo de desenvolvimento em minutos	KLOC
Windows Phone Nativo	17	32	0,53125
Android Nativo	38	68	0,558823529
PhoneGap	12	23	0,52173913

Fonte: Do autor (2016)

c) para a construção do acesso ao GPS do dispositivo conforme tabela 7,foi necessário 13 linhas de código fonte e 25 minutos de desenvolvimento para o PhoneGap já para o desenvolvimento nativo em Android foi necessário 19 linhas e 37 minutos de desenvolvimento e para o Windows Phone foi necessário 23 linhas de código fonte e 44 minutos, para acesso ao GPS o PhoneGap teve o desenvolvimento mais rápido que desenvolvimentos nativos, e para a construção desta funcionalidade

utilizando o PhoneGap cross-plataforma com o KLOC de 0,52 linha/min teve um custo menor comparado ao Android nativo com o KLOC de 0,513514 linha/min e maior comparado ao Windows Phone nativo com o KLOC de 0,522727273 linha/min.

Tabela 7 – Comparação Acesso ao GPS

Aplicativo	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Windows Phone Nativo	23	44	0,522727273
Android Nativo	19	37	0,513513514
PhoneGap	13	25	0,52

Fonte: Do autor (2016)

d) para a construção da comunicação com o WebService utilizando a internet do dispositivo conforme a tabela 8, foi necessário 17 linhas de código fonte e 16 minutos de desenvolvimento do método para o aplicativo em PhoneGap esse resultado foi possível pela a facilidade na utilização na bibliotecas do jQuery, para o Android foi necessário 31 linhas de código fonte e 58 minutos de desenvolvimento e para o Windows Phone doi necessário 33 linhas de código fonte e 59 minutos de desenvolvimento, a construção desta funcionalidade utilizando o PhoneGap cross-plataforma com o KLOC de 1,0625 linha/min mostrou-se com o custo menor ao Android nativo com o KLOC de 0,534483 linha/min e o Windows Phone Nativo com o KLOC de 0,559322034 linha/min.

Tabela 8 – Comparação Comunicação Ws

Aplicativo	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Windows Phone Nativo	33	59	0,559322034
Android Nativo	31	58	0,534482759
PhoneGap	17	16	1,0625

Fonte: Do autor (2016)

e) em uma visão geral para a construção de todos os métodos conforme a tabela 12 foi necessário 111 linhas de código fonte e 123 minutos de desenvolvimento, os métodos em PhoneGap mostraram-se menores e de simples de desenvolver, comparado com o desenvolvimento dos métodos nativos em Android com 179 linhas de código fonte e 218 minutos de desenvolvimento e Windows Phone com 124 linhas de código fonte e 181

minutos de desenvolvimento, assim com um custo de desenvolvimento em 0.902439024 linhas por minuto o desenvolvimento em PhoneGap mostrou-se com um custo menor de desenvolvimento comparado com o Android nativo com 0,821100917 linhas por minuto e com o Windows Phone com 0,685082873 linhas por minuto.

Tabela 9 – Comparação TOTAL Desenvolvimento

Aplicativo	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Windows Phone Nativo	124	181	0,685082873
Android Nativo	179	218	0,821100917
PhoneGap	111	123	0,902439024

Fonte: Do autor (2016)

6.2.3 Analise comparativa de desempenho

Considerando os dados obtidos após a aplicação de carga de exclusão nas funções de testes desenvolvidas, conforme as tabelas 10 e 11, identificou-se que para os testes de comparativos de desempenho entre os aplicativos cross-plataforma PhoneGap com o Android e Windows Phone nativo as aplicativos nativas mostraram-se com o desempenho superior aos aplicativos cross-plataforma, isto ocorre pois os aplicativos em PhoneGap rodam na camada de WebView conforme capitulo 4 assim todos os processos tem que passar por essa camada antes de chegar ao SO nativo, já as aplicações nativas rodam junto com o SO nativo diminuindo o tempo de resposta, e melhorando o desempenho do aplicativo.

Tabela 10 – Desempenho Android

	Nativo	**Cross-Plataforma**
Acesso a câmera	616ms	784ms
Acesso ao GPS	77ms	98ms
Acesso ao WS	414ms	569ms

Fonte: Do autor (2016)

Tabela 11 – Desempenho Windows Phone

	Nativo	**Cross-Plataforma**
Acesso a câmera	376ms	483ms
Acesso ao GPS	66ms	76ms
Acesso ao WS	234ms	490ms

Fonte: Do autor (2016)

Após a análise comparativa entre os aplicativos nativos e os cross-plataforma PhoneGap identificou-se que o desenvolvimento em PhoneGap tem um

custo menor e tem um tempo de desenvolvimento reduzido mas quando estamos falando de estabilidade e desempenho o mesmo se mostra inferior aos aplicativos nativos, os aplicativos nativos apesar de terem um custo maior no desenvolvimento mostraram-se estáveis e com um melhor desempenho, nos quesitos de manutenção do aplicativo as nativas se mostraram-se melhor, pois as mesmas tem IDEs especificas para o seu desenvolvimento e várias ferramentas para auxiliar o desenvolvimento, já o PhoneGap não tem uma IDE especifica assim e desenvolvido normalmente em editores de texto simples como NotPad++ e Sublime Text.

Por fim, o desenvolvimento de um aplicativo cros-plataforma pode ser útil quando temos que atender várias plataformas e temos pouco tempo para desenvolver o aplicativo, o PhoneGap mostrou-se com um custo e uma facilidade no desenvolvimento que pode pesar bastante na decisão de como começar o desenvolvimento de um novo aplicativo, este trabalho pode ser mais um dos instrumentos utilizado por desenvolvedores e analistas para justificar a decisão de qual tecnologia iniciar o desenvolvimento de um aplicativo.

7 CONCLUSÃO

A aplicação da análise comparativa entre aplicativos mobile nativo e cross-plataforma contribui para o início de desenvolvimentos com tecnologia mobile. Assim como um produto necessita de análise de regra de negócio antes de passar pelo processo de desenvolvimento, e necessário decidir a melhor forma de se desenvolver o mesmo, para que no futura não sejam identificado pontos falhos no desenvolvimento.

Com a realização deste trabalho, notou-se que a várias formas de análise comparativa de custo e desempenho que podem se adequar a ambientes distintos e condições diferentes, as métricas orientadas ao tamanha se adequaram a todos os requisitos que foi avaliado para as definições de custo de desenvolvimento, contudo também foi utilizado a metodologia de sobrecarga para avaliar o desempenho das funcionalidades desenvolvidas.

Para este trabalho o foco foi a comparação entre aplicativos nativos e cross-plataformas em Android e Windows Phone, visando apresentas os custos de desenvolvimento em KLOC e desempenho. Foi possível perceber a evolução da computação e quanto o desenvolvimento de aplicações mobile nativa e cross-plataforma estão evolução, além de conhecer os recursos de desenvolvimentos em PhoneGap, Android e Windows Phone.

O estudo acerca de dispositivos moveis auxiliou no entendimento do funcionamento das plataformas Android e Windows Phone, e com esse entendimento, foi possível identificar e escolher quais as funcionalidades e serem comparadas, a comparação do desenvolvimento nativo com o cross-plataforma em PhoneGap foi possível através do desenvolvimento de um aplicativo de Achado e Perdidos que contempla as funcionalidades de acesso ao Hardware do dispositivo e acesso externo utilizando a internet, além de possibilitar a comparação das telas desenvolvidas.

A mescla de duas formas de análise, usando as metodologias de métricas orientada ao tamanho e análise de sobrecarga de execução foi o tema deste trabalho e mostrou eficiência para a aplicação no desenvolvimento de aplicativos nativo e cross-plataforma. A comparação dos resultados obtidos na métrica orientada ao tamanho mostrou que o custo de desenvolvimento para um aplicativo cross-plataforma em PhoneGap e menor comparado ao desenvolvimento dos aplicativos nativos, mas em relação ao desempenho ele se mostrou inferior aos

aplicativos nativos. Além disso, com a análise comparativa de custo de desenvolvimento e desempenho, foi possível identificar que para o desenvolvimento de aplicativos voltado ao desempenho e uso de hardware do dispositivo a melhor pratica seria utilizar a tecnologia nativa, pois apresentou resultados melhores e instáveis.

Os trabalhos correlatos agregaram formidavelmente para a definição do trabalho desenvolvido, e alguns forneceram subsídios do desenvolvimento e análise comparativa de aplicativos cross-plataforma.

A metodologia resultante da mescla das análise comparativa de custo e de desempenho da comparação de aplicativos cross-plataforma e nativo, podem ser utilizados como base para desenvolvimento de novos aplicativos considerando o desempenho e o custo de desenvolvimento e fornecer aplicativos instáveis e com melhor resultado ao usuário.

De modo geral, todos os objetivos específicos foram cumpridos e a apresentação deste trabalho pode ser mais um dos exemplos para o alerta de como a falta de definição de tecnologia que será utilizada para o início dos aplicativos podem afetar diversas áreas e impedir que um produto evolua.

Como sugestão de trabalhos futuros, podem-se: aplicar ou mesclar outras métricas na análise comparativa, inclusive os descritos neste trabalho; aplicar a metodologia dessa pesquisa em outras funcionalidades ou outras plataformas; avaliar outros tipos de desenvolvimento cross-plataforma comparado com o desenvolvimento nativo.

REFERÊNCIAS

ABLESON, W. Frank et al. **Android em ação.** 3.ed. Rio de Janeiro: Elsevier, 2012. 656p.

ALLEN, Sarah; GRAUPERA, Vidal; LUNDRIGAN, Lee. **Pro Smartphone Cross-Platform Development:** iPhone, Blackberry, Windows Mobile and Android Development and Distribution. New York: Apress, 2010. 288 p.

BUDIU, Raluca. **Mobile: Native Apps, Web Apps, and Hybrid Apps**. 2013 Disponível em: <http://www.nngroup.com/articles/mobile-native-apps/>. Acesso em: 17 jun. 2015.

CARDOSO, Gabriel Schade. **Criando aplicação para seu Windows Phone.** São Paulo: Casa do Código, 2014. 149 p.

CARVALHO, Alan. **Criando na Internet com HTML 4.** Rio de Janeiro: Book Express, 2001. 183 p.

CAFÉ, Adriel Almeida. **Desenvolvimento de Cross-Platform Mobile Apps Utilizando o Titanium Mobile.** 2012. 19 f. Tese (Doutorado) - Curso de Ciências da Computação, Faculdade Zacarias de Góes, Valença, 2012.

CHOU, Tim. **Os custos ocultos de Software.** Tradução nossa. Disponível em: <http://www.datamation.com/entdev/article.php/11070_2214031_2/The-Hidden-Cost-of-Software.htm>. Acesso em: 15 nov. 2015.

CLARO, Daniela Barreiro; SOBRAL, João Bosco Mangueira. **Programação Java.** Florianopolis: Pearson Education, 2000.

CLARO, **Daniela B. Métricas de Software**. 2008. Disponível em: <http://www.inf.ufsc.br/~danclaro/download/disciplinas/M%E9tricas%20de%20Software.doc>. Acesso em: 20 de mai. de 2016.

CORREIOS. **Achados e perdidos**. Disponível em: <https://www.correios.com.br />. Acesso em: 01 Mai. 2016.

DARIVA, Roberto. **Gerenciamento de dispositivos móveis e serviços de telecom**. Rio de Janeiro: Elsevier, 2011. 160 p.

DEITEL, Harvey M.. **Java como programar.** 4. ed. São Paulo: Bookman, 2003.

DURÃES, Ramon. **Introdução ao Microsoft Silverlight.** Disponível em: <http://www.linhadecodigo.com.br/artigo/1290/introducao-ao-microsoft-silverlight.aspx>. Acesso em: 16 nov. 2015.

ESPADA, Jordán. **Using extended web technologies to develop bluetooth multiplataforma mobile application fron interact with smart thigs.** Madrid: Elsevier, 2013. 12 p.

FIGUEIREDO, Carlos Maurício Seródio; NAKAMURA, Eduardo. **Computação Móvel:** Novas Oportunidades e Novos Desafios. **T&C Amazônia**, Manaus, n. 2, p.16-28, 01 jun. 2003. Semestral. Disponível em: <https://portal.fucapi.br/tec/imagens/revistas/ed02_completo.pdf> Acesso em: 02 Out. 2015.

GOMES, Alvaro Eduardo. **Métricas e Estimativas de Software:** O início de um rally de regularidade. Disponível em: <http://www.linhadecodigo.com.br/artigo/102/metricas-e-estimativas-de-software-o-inicio-de-um-rally-de-regularidade.aspx>. Acesso em: 15 nov. 2015.

GOOGLE. **Android Developers**. Disponível em: <http://developer.android.com/>. Acesso em: 01 Nov. 2015.

HICKSON, Rosângela. **Aprenda a programar em C, C++ e C#.** Rio de Janeiro: Editora Campus, 2002. 358 p.

IBGE. **PNDA:** De 2005 para 2011, número de internautas cresce 143,8% e o de pessoas com celular, 107,2%. Disponível em: <http://saladeimprensa.ibge.gov.br/noticias?view=noticia&id=1&idnoticia=2382&busca=1&t=pnad-2005-2011-numero-internautas-cresce-143-8-pessoas-celular-107-2 >. Acesso em: 08 out. 2015.

JIMÉNEZ, Dayron Agüero; MENESES, Yadira Calimano. Estudio de viabilidad de una herramienta software para monitorización de tráfico IP en Windows Phone. **Revista Cubana de Ciencias Informáticas,** La Habana, p.23-35, 01 mar. 2013. Mensal. Disponível em: <http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992013000100004&lang=pt>. Acesso em: 02 nov. 2015.

JOBSTRAIBIZER, Flávia. **Criação de Aplicativos para Celulares com Google Android.** São Paulo: Universo dos livros, 2009. 128 p.

KESSIN, Zachary. **Programming HTML5 Applications.** New York: O'reilly Media, 2011. 144 p.

LECHETA, Ricardo. R. **Googloe Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK. 3. ed. São Paulo: Novatec, 2013. 824 p.

LEE, Wei-meng. **Beginning Android™ Application Development**. Indianápolis: Wiley, 2011.

LIMA, Edwin. **C# E .NET – GUIA DO DESENVOLVEDOR.** Rio de Janeiro: Editoração Eletrônica Riotexto, 2002. 358 p.

MAZZA, Lucas. **HTML5 E CSS3 - DOMINE A WEB DO FUTURO.** São Paulo: Casa do Código, 2013. 2005 p.

MEIER, Reto. **Professional Android™ 2 Application Development.** Indianapolis: Wrox, 2010. 576 p.

MICROSOFT. **Windows Phone Emulator for Windows Phone 8.** https://msdn.microsoft.com/pt-br/library/windows/apps/ff402563(v=vs.105).aspx Acesso em: 09 Nov. 2015.

MICROSOFT. **Windows Phone Developers**. Disponível em: https://msdn.microsoft.com/library/windows/apps/ff402563(v=vs.105).aspx#feedback Acesso em: 02 Nov. 2015.

MICROSOFT. **SILVERLIGTH.** Disponível em: <https://msdn.microsoft.com/pt-br/library/bb404700(v=vs.95)>. Acesso em: 16 nov. 2015.

MOBILE, Vision. **Cross-Platform Developer Tools**. 2012 Disponível em: <http://developer.verizon.com/content/dam/vdc/documents/VisionMobile-Cross-Platform-Tools-2012-blog-post.pdf>. Acesso em: 17 jun. 2015.

MONACO, Thiago José. **Introdução ao Windows Phone 7**. https://msdn.microsoft.com/pt-br/library/gg699671.aspx Acesso em: 10 Nov. 2015.

MONACO, Thiago; CARMO, Rodolpho Marques do. **Desenvolvendo Aplicações para Windows Phone.** Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2012. 453 p.

MUNRO, Jamie. **20 Recipes for Programming PhoneGap.** Sebastopol,: O'reilly Media, 2012.

MYER, Thomas. **BEGINNING PhoneGap.** Indianapolis: John Wiley & Sons, Inc., 2012. 362 p.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android para desenvolvedores.** Rio de Janeiro: Brasport, 2009. 240 p.

PETZOLD, Charles. **Programming Windows Phone 7.** Washington: Microsoft Corporation, 2010. 997 p. Tradução nossa.

PHONEGAP. **PhoneGap Developer App Reference Guide.** Disponível em: <http://docs.phonegap.com/references/developer-app

POULSEN, T.; WHINNERY, K.; LUKASAVAGE, T.; DOWSETT, **P. Building Mobile Applications with Titanium**. Mountain View: Appcelerator, 2012.

PRESSMAN, Roger S**. Engenharia de Software**, São Paulo: Makron Books, 1995.

PREZOTTO, Ezequiel Douglas; BONIATI, Bruno Batista. **Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas.** 2014. 8 f. TCC (Graduação) - Curso de Tecnologia em Sistemas Para Internet, Universidade Federal de Santa Maria, Rio Grande do Sul, 2014.

SCHWARZ, Ronan et al. **The Android developer's cookbook:** Building application with the Android SDK. 2. ed. 2013. 464 p.

SILVA, Maurício Samy. **HTML 5 a linguagem de marcação de revolucionou a Web.** São Paulo: Novatec Editora Ltda., 2011.

SILVA, Maurício Samy. **CSS3:** Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das css3. São Paulo: Novatec Editora Ltda., 2012. 150 p.

SILVEIRA, Paulo et al. **Introdução a arquitetura e design de software:** Uma visão sobre a plataforma Java. Rio de Janeiro: Elsevier, 2012. 284 p.
SHACKLES, Greg. **Mobile Development with C#:** Building Native iOS, Android, and Windows Phone Applications.New York: O'reilly, 2012. 174 p.

SOMMERVILLE, I. **Engenharia de Software**. 6ª Ed. São Paulo: Addison Wesley, 2003.

STARK, Jonathan. **Building Android Apps with HTML, CSS, and JavaScript:** Making Native Apps with Standards-Based Web Tools. New York: O'reilly Media, 2010. 184 p.

TELES, Fabrício de Siqueira. **Um Processo para Análise de Desempenho de Produtos de Software.** 2005. Disponível em: <http://scielo.sld.cu/scielo.php?.script=sci_arttext&pid=S2227-18992013000100004&lang=pt>. Acesso em: 15 nov. 2015.

VASCONCELOS, Alexandre**. Métricas de Software**, 2005. Disponível em: <http://www.cin.ufpe.br/~if720/slides/introducao-a-metricas-de-software.ppt>. Acesso em: 01 de Jun. de 2016.

WAHLBRINCK, Kamile A.; BONIATI, Bruno B.. **Análise de Performance de Frameworks de Desenvolvimento Mobile Multiplataforma.** Frederico Westphalen - Rs: 1, 2014. Disponível em: <http://www.eati.info/eati/2014/assets/anais/artigo52.pdf>. Acesso em: 01 jun. 2016.

APÊNDICE(S)

APÊNDICE A - ARTIGO CIENTÍFICO

Análise Comparativa de Aplicações Mobile Nativa e Aplicações Mobile Cross-Plataforma

Luciano Antunes[1], Samuel S. Matos[2]

[1]Professor do Curso de Ciências da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – Brasil

[2]Acadêmico do Curso de Ciências da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – Brasil

luc@unesc.net, samuel.matos.mtx@hotmail.com

Abstract.*. The mobile devices are increasingly evolving and popularizing and bringing the application development demand for such devices, so with these new applications arise the need to carry out a comparative analysis of performance and cost of development of native applications and cross-platform applications .*

Resumo. *Os dispositivos moveis estão cada vez mais evoluindo e se popularizando e trazendo consigo a demanda de desenvolvimento de aplicativos para tais dispositivos, assim com esses novos aplicativos surgem a necessidade de efetuar análise comparativa de desempenho e custo de desenvolvimento entre aplicativos nativos e aplicativos cross-plataforma.*

1. Introdução

Atualmente, com o crescimento de tecnologias para dispositivo moveis, observou-se que a diversidade de plataformas e aplicativos para os mesmos aumentou consideravelmente, consequentemente as plataformas e *Integrated Development Environment* (IDE) evoluíram junto. Esta situação impacta diretamente nos custos de desenvolvimento, pois desta forma a complexidade de atender todas as plataformas disponíveis aumenta, pois é necessário conhecimentos específicos da linguagem de cada plataforma, e também é necessário desenvolver uma aplicação diferente para cada uma delas (PREZOTTO; BONIATI, 2014).

Segundo Kessin (2011), visando sanar estes problemas surgiu as *webapps* e as *hybrid apps*, na web as aplicações são Cross-plataforma assim como é possível escrever páginas de internet que irão funcionar no Windows, Mac OS X, Linux, iPhone/iPad, Android entre outras plataformas. Como toda plataforma mobile possui browsers que são capazes de interpretar scripts do JavaScript e reproduzir documentos HyperText Markup Language (HTML) customizados com Cascading Style Sheets (CSS), é possível desenvolver *Apps* que irão funcional em qualquer plataforma existente para dispositivos moveis.

Hoje as plataformas possuem características distintas que tornam elas mais atrativas para os usuários e desenvolvedores, normalmente os usuários procuram os dispositivos mais fáceis de serem manipulados, que tenham mais funcionalidades e um bom custo-benefício. Mas os desenvolvedores procuram plataformas com uma grande quantidade de dispositivos ativos, que tenha uma boa documentação e suporte técnico, de fácil aprendizado, baixo custo de desenvolvimento e tecnologias familiares (CAFÉ, 2012).

Desta forma, a pesquisa aqui proposta, ressalta a necessidade de desenvolver um aplicativo, usando tecnologia nativa e cross-plataforma, e efetuar

testes e análise comparativa de desempenho e custo entre elas, visando identificar qual aplicação terá o maior desempenho, também evidenciar quais as dificuldade que os desenvolvedores terão ao iniciar o desenvolvimento usando tecnologia cross-plataforma.

2.Plataforma Android

O Android é um SO móvel baseado em uma versão modificada do Linux. Foi originalmente desenvolvido por uma equipe com mesmo nome, Android, Inc. Em 2005 a Google comprou o Android e assumiu o seu trabalho de desenvolvimento (bem como sua equipe de desenvolvimento), isso fazia parte de uma estratégia para entrar no espaço de dispositivos móveis (LEE, 2011, tradução nossa).

A Google queria que o Android fosse *open-source* e *software* livre, portanto, a maioria do código do Android foi liberado sob a licença Apache *open-source*, deste modo qualquer um que tivesse interesse em utiliza-lo teria fácil acesso ao download do código fonte completo, além disso as empresas podem adicionar suas próprias extensões e customiza-lo, diferenciando o seu produto dos demais (LEE, 2011, tradução nossa).

Por ser a primeira plataforma de software que possui aplicativos para telefones celulares *open-source*, o Android vem revolucionando o mercado mundial de aplicativos para smartphones e sendo demasiadamente notado pelos maiores mercados de telefonia celular do globo (ABLESON et al, 2012).

3.Plataforma WindowsPhone

O sistema operacional Windows Mobile da família Windows, desenvolvido pela Microsoft, apesar de ter o nome do Windows, não é um sistema derivado ou uma versão colhida do mesmo, mas um novo sistema projetado especificamente para dispositivos móveis, isso não significa que ele não apresenta semelhanças aos sistemas operacionais para PC desenvolvido pela Microsoft, pois desta forma o SO permite que o usuário tenha uma familiaridade e uma experiência agradável com o dispositivo (CARDOSO ,2014).

A Microsoft desenvolveu o Windows Phone com um intuito diferente, a solução foi desenvolvida para integrar todas as “telas” da Microsoft, de forma a ajudar a visão da empresa a se tornar realidade. Em qualquer apresentação da Microsoft é sempre falado das três telas, o Windows no seu PC, o Xbox na sua TV, e o Windows Phone em suas mãos, o Windows Phone em suas versões futuras será cada vez mais o hub de conexão entre todas as soluções Microsoft (MÔNACO; CARMO, 2012).

Windows Phone é totalmente integrado com o .NET Compact Framework fornecendo assim uma série de bibliotecas para desenvolvimento das aplicações. Estas aplicações poderão ser desenvolvidas a partir do próprio Visual Studio, ou pelo Visual Studio Express for Windows Phone, sendo que em ambos os produtos as aplicações poderão ser criadas utilizando Silverlight ou XNA (PETZOLD, 2010, tradução nossa).

4.PhoneGap Cross-Plataforma

Basicamente o PhoneGap é um conjunto bibliotecas que permite aos desenvolvedores interagir diretamente com um dispositivo móvel através da utilização de JavaScript. Com o grande número de plataformas móveis é muito difícil e caro para criar múltiplas aplicações em Java, Objective-C, ou outras línguas nativas. Através da biblioteca PhoneGap, a maioria dos desenvolvedores web

podem utilizar seu conhecimento existente de HTML, CSS e JavaScript para desenvolver aplicativos moveis cross-plataforma (MUNRO, 2012).

Esta bibliotecas específicas fornece recursos de comunicação que as tecnologias da web não têm, tais como acesso ao módulo Bluetooth e Wi-Fi para enviar e receber informações. Mas em qualquer caso, as aplicações construídas com esta tecnologia podem ser considerados como aplicações *webapps* híbrida (ESPADA, 2013).

O PhoneGap fornece aos desenvolvedores acesso ao recursos nativos do dispositivo móvel do usuário, ele também fornece as tecnologias necessárias para interagir com os sistemas externos localizado em outras redes. Isso significa que os usuários podem ser altamente móvel, sem perder o acesso a altamente valiosa e oportuna dados da empresa (SHOTTS, 2014).

Hoje o PhoneGap e compatível com os maiores SOs disponíveis no mercado, dentre eles Windows Phone, Android e IOS, pois seus aplicativos são construídas por tecnologia web aberto (PHONEGAP, 2015).

5.Metrica Orientada ao tamanho e Teste de Desempenho por Carga

As métricas de software têm como objetivo especificar as funções de coleta de dados de avaliação e desempenho, custo, prevenindo problemas na exclusão do projeto ou na implantação do mesmo. Assim quando vemos aplicação de métricas temos que ter em mente que se trata de avaliação de dados números quantitativos que irão auxiliar na decisão e mostrar indicadores do estado atual do projeto ou software (PRESSMAN, 1995).

Métricas de software orientadas ao tamanho são medidas diretas do software e do processo por meio do qual ele é desenvolvido, elas são utilizadas para mesurar o custo por *Thousand* Lines of Code (KLOC), a partir de dados brutos obtidos durante o desenvolvimento do Software (GOMES, 2015).

As métricas orientadas a tamanho, consideram o tamanho do software produzido em linhas de código, também podem ser aplicadas em todas as atividades da engenharia análise, projeto, código, teste, e poderão medir produtividade em KLOC/pessoa-mês, qualidade defeitos/KLOC. Já a métrica orientada à função em vez de contar linhas de código são focada em analisar e efetuar medidas na funcionalidade do software (SOMMERVILLE, 2003).

A análise de pontos por função e avaliada inteiramente pelo o usuários e como os mesmos veem o produto final e os resultados obtidos pelo software produzido. Ela se baseia parcialmente em dados subjetivos, implicando a organização estabelecer um plano de implantação da sistemática da medição, definindo padrões para contagem (CHOU, 2015).

Para análise de desempenho de software segundo Teles (2005), é necessário definir o fluxo do software, após isso estimar o tempo e recursos consumidos pelo Software, podendo descobrir potenciais pontos de gargalo e predizer o desempenho do sistema.

6 Análise Comparativa de Aplicações Mobile Nativa e Aplicações Mobile Cross-Plataforma

O desenvolvimento pratico deste trabalho objetivou a análise comparativa de desempenho e custo de desenvolvimento de aplicativos nativos comparando com aplicativos cross-plataforma Mobile utilizando Métricas orientada ao tamanha para poder definir os custos e teste de sobrecarga para os testes de desempenho. Os conhecimentos adquiridos durante o levantamento bibliográfico referente a dispositivos moveis, as plataformas Android e Windows Phone aliados aos estudos

das métricas de teste, propiciaram subsídios para a quantificação do desempenho e do custo em KLOC de desenvolvimento.

O aplicativo do sistema operacional Android nativo foi desenvolvido na IDE do Android Studio, já para o sistema operacional Windows Phone nativo foi desenvolvido na IDE Visual Studio 2012, o aplicativo cross-plataforma foi desenvolvido na IDE PhoneGap.

As três aplicações tem a mesma função, elas efetuaram integração em um WebService desenvolvido em Java com banco MySql, tem um processo sobrecarga de processo, para que fosse possível efetuar análise comparativa de desempenho entre as aplicações cross-plataforma e nativa, contudo também foi analisado e tempo gasto no desenvolvimento das telas e do acesso ao Hardware do dispositivo afim de estipular o custo necessário para desenvolver cada funcionalidade, as aplicações acessaram a câmera e o GPS afim de estipular desempenho do software ao acessar o hardware.

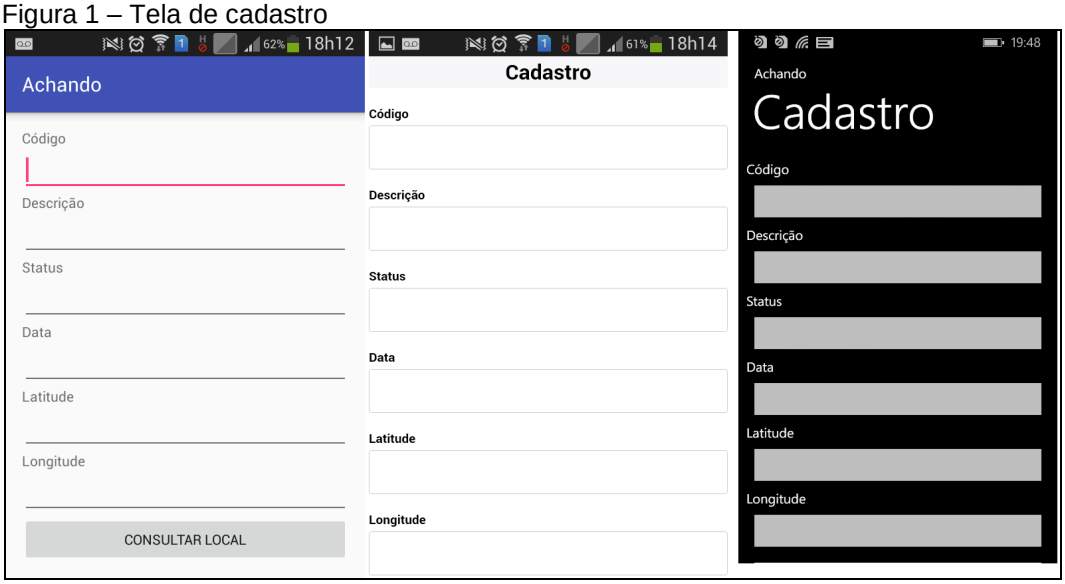
6.1.6 Desenvolvimentos das aplicações mobile nativa e aplicações mobile cross-plataforma de um aplicativo de Achados e Perdidos

Para a análise comparativa de custo de desenvolvimento por KLOC foi desenvolvido um aplicativo de achados e perdidos, conforme os Correios (2016) são perdidos em 10 dias mais de 15 mil documentos isto somente documentos, estes dados mostram a relevância de se desenvolver um aplicativo para podermos cadastrar e consultar os itens achados e perdidos , assim podendo os encontrar mais facilmente, contudo ainda avaliarmos as tecnologias nativas e cross-plataforma, a análise dos aplicativos foi efetuada utilizado a Métrica orientada ao tamanho, explicado no capitulo 6, este desenvolvimento e quantificação de tempo foi dividida na seguinte lista de etapas.

6.2 Desenvolvimento de Tela de cadastro

Nesta etapa foi desenvolvida uma tela de cadastro que contém os campos: Código, Descrição, Data, Latitude, Longitude e Foto do Item perdido. Esta tela foi desenvolvida Nativa em Windows Phone e Android e cross-plataforma utilizando o PhoneGap, nas três aplicações foram utilizados os componentes de tela nativos de cada linguagem, para o Windows Phone e Android foram utilizado o XML e para o PhoneGap o HTML, esta tela foi desenvolvida para cadastra itens de achados e perdidos com intuito de ter um fácil uso, assim conforme a figura 1 abaixo as telas tem uma aparência diferente uma da outra, pois para cada tecnologia há um padrão de componentes diferente.

Figura 1 – Tela de cadastro



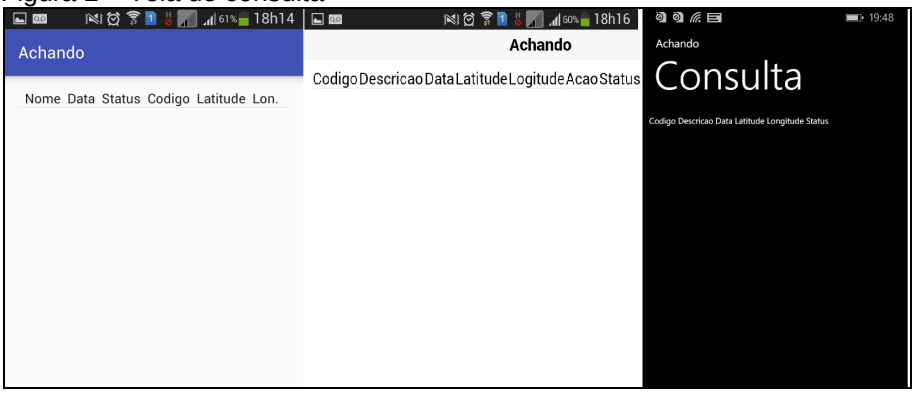
Fonte: Do autor (2016)

6.3 Desenvolvimento das consultas no Web Service

Nesta etapa foi desenvolvido uma tela que efetua uma consulta em um Web Service Java, após isso os dados retornados são mostrados na tela em uma lista que contém: Código, Descrição, Data, Latitude, Longitude, esta informação e mostrada em uma grid padrão do próprio aplicativo.

Foi utilizado o *HTTPClient* para efetuar as consultas pelo aplicativo Android e *WebClient* no Windows Phone já no PhoneGap foi utilizado *jQuery*, esse bibliotecas foram utilizada para construir uma consulta via GET no *WebService,* esta consulta retorna um *Json* que os aplicativos percorrem e mostram na tela, na figura 12 podemos ver a tela nas versões Windows Phone nativo e cross-plataforma e Android Nativo e cross-plataforma.

Figura 2 – Tela de consulta



Fonte: Do autor (2016)

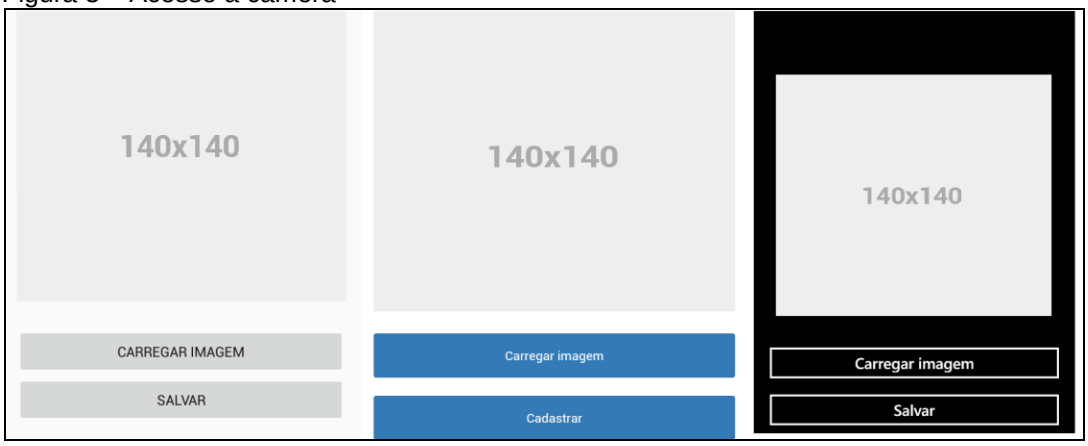
6.4 Desenvolvimento do acesso a câmera do dispositivo

Nesta etapa foi desenvolvido o acesso a câmera do dispositivo, esta funcionalidade ficou localizada na tela de cadastro de item perdido do aplicativo. Esta funcionalidade consiste em acessar a câmera física do dispositivo capturar uma foto e mostrar na tela do aplicativo.

Para os três tecnologias foi construído um método que efetua a chamada da câmera, e outro que fica esperando o retorno da mesma, no Android foi

inicializado uma “*activity”* com um “*intente”* passando por parâmetro a propriedade “*MediaStore.ACTION_IMAGE_CAPTURE*”, já para o WindowsPhone foi efetuado uma chamada no “*CameraCaptureTask”*, para o PhoneGap foi efetuado uma chamada no método “*navigator.camera.getPicture”*, após isso no método de retorno a imagem retornada e convertida em um *bitmap* e apresentada na tela, conforme figura 3 esta funcionalidade está relacionada diretamente ao botão “Carregar imagem” do aplicativo.

Figura 3 – Acesso a câmera



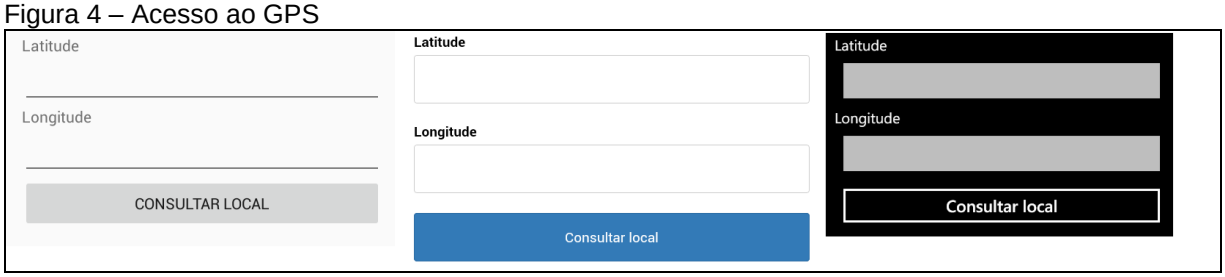
Fonte: Do autor (2016)

6.5 Desenvolvimento ao acesso GPS

Nesta etapa foi desenvolvido o acesso ao GPS do dispositivo, esta funcionalidade está localizada na tela de “Cadastro de item perdido”, ela consiste de acessão o sistema de localização e retornar a latitude e longitude para a tela do aplicativo.

Para o desenvolvimento desta função em Android foi utilizado a biblioteca “*LocationManeger”*, para o Windows Phone foi utilizado a biblioteca “*Geoposition”* e para o PhoneGap o método “*navigator.geolocatio.getCurrentPositon*”, estes métodos tem como retorno a latitude e longitude e outras informações sobre a localização do dispositivo, conforme figura 4, esta funcionalidade está relacionada diretamente ao botão “Consultar local” do aplicativo, e o retorno dela e mostrado nos campos de “Latitude” e “Longitude”.

Figura 4 – Acesso ao GPS



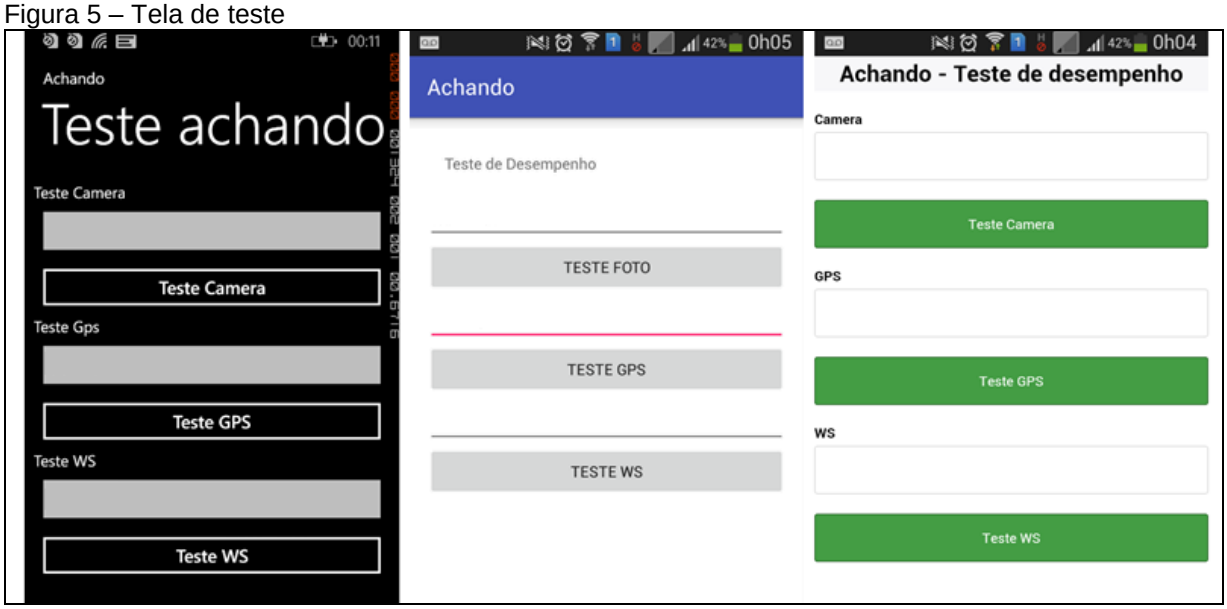
Fonte: Do autor (2016)

6.6 Desenvolvimento das aplicações mobile nativa e mobile cross-plataforma para testes de desempenho

Para a análise comparativa desempenho foi desenvolvido um tela junto ao aplicativo de achados e perdidos, que efetuara os teste de desempenho por carga, está tela efetua três testes, caga no acesso ao GPS e Web Service, e acesso a câmera, Conforme figura 5 para cada teste foi criado um botão e executara a função de teste desenvolvida, a função de teste do GPS e Web Service foram construídas para executar a consulta 100 vezes e retornar a média de tempo de exclusão em tela, o

teste de acesso a câmera será executado uma única vez pois este processo necessita de interação do usuário para confirmar a imagem.

Figura 5 – Tela de teste



Fonte: Do autor (2016)

Para os testes de desempenho do aplicativo e comparações entre os aplicativos Android nativo e cross-plataforma foi utilizado o dispositivo Samsung Galaxy S3 Slim G3812, com processador Quad core de 1.2gz e 1 GB de memória RAM, com Android 4.2.2.

Para os testes de desempenho do aplicativo e comparações entre os aplicativos Windows Phone nativo e cross-plataforma foi utilizado o dispositivo Nokia Lumia 530, com processador Quad core de 1.2gz e 512 GB de memória RAM, com Windows Phone 8.1.

6.7 Resultados Obtidos

Ao terminar a análise comparativa dos aplicativos de achados e perdidos desenvolvidos obtivemos os seguintes resultados da comparação dos resultados da Métrica Orientada ao Tamanho e dos testes de Sobrecarga.

6.7.1 Analise comparativa utilizando Métrica Orientada ao Tamanho

Considerando os dados obtidos com a métrica orientada ao tamanho identificou-se que:

f) para a construção da tela de cadastro conforme a tabela 1, foi necessário 51 linha de código fonte e 46 minutos de desenvolvimento para o Windows Phone , 91 linhas de código fonte e 55 minutos para o Android e 69 linhas de código fonte e 59 minutos para o PhoneGap, o código fonte do Android mostrou-se mais longo do que os demais e um tempo de desenvolvimento semelhante, a construção utilizando o PhoneGap cross-plataforma obteve um KLOC de 1,1694915 linha/min, comparando este resultado com o KLOC da construção nativa o PhoneGap apresentou-se superior somente com o Windows Phone nativo com 1,108695652 linha/min, já com Android nativo foi obtido um KLOC de 1,654545 linha/min sendo superior ao PhoneGap cross-plataforma e o Windows Phone;

Tabela 1 – Comparação Tela Cadastro

Aplicativo	Quantidade de linhas	Tempo de desenvolvimento em minutos	KLOC
Windows Phone Nativo	51	46	1,108695652
Android Nativo	91	55	1,654545455
PhoneGap	69	59	1,169491525

Fonte: Do autor (2016)

g) para a construção do Acesso a câmera física do dispositivo conforme a tabela 2, foi necessário 17 linhas de código fonte e 32 minutos para o Windows Phone Nativo, 38 linhas de código fonte e 68 minutos para o Android nativo e para o PhoneGap cross-plataforma foi necessário 12 linhas de código fonte e 23 minutos de desenvolvimento, o acesso a câmera em Android mostrou-se maior que os demais mas mais simples de desenvolver, a construção desta funcionalidade utilizando o PhoneGap cross-plataforma com o KLOC de 0,5217391 linha/min teve um custo maior comparado ao Android com o KLOC de 0,558824 linha/min e Windows Phone com o KLOC de 0,53125 linha/min, neste comparação os aplicativos nativos mostraram-se melhor que o cross-plataforma.

Tabela 2 – Comparação Acesso a Câmera

Aplicativo	Quantidade de linhas	Tempo de desenvolvimento em minutos	KLOC
Windows Phone Nativo	17	32	0,53125
Android Nativo	38	68	0,558823529
PhoneGap	12	23	0,52173913

Fonte: Do autor (2016)

h) para a construção do acesso ao GPS do dispositivo conforme tabela 3,foi necessário 13 linhas de código fonte e 25 minutos de desenvolvimento para o PhoneGap já para o desenvolvimento nativo em Android foi necessário 19 linhas e 37 minutos de desenvolvimento e para o Windows Phone foi necessário 23 linhas de código fonte e 44 minutos, para acesso ao GPS o PhoneGap teve o desenvolvimento mais rápido que desenvolvimentos nativos, e para a construção desta funcionalidade utilizando o PhoneGap cross-plataforma com o KLOC de 0,52 linha/min teve um custo menor comparado ao Android nativo com o KLOC de 0,513514 linha/min e maior comparado ao Windows Phone nativo com o KLOC de 0,522727273 linha/min.

Tabela 3 – Comparação Acesso ao GPS

Aplicativo	Quantidade de linhas	Tempo de desenvolvimento em minutos	KLOC
Windows Phone Nativo	23	44	0,522727273
Android Nativo	19	37	0,513513514
PhoneGap	13	25	0,52

Fonte: Do autor (2016)

i) para a construção da comunicação com o WebService utilizando a internet do dispositivo conforme a tabela 4, foi necessário 17 linhas de código fonte e 16 minutos de desenvolvimento do método para o aplicativo em PhoneGap esse resultado foi possível pela a facilidade na utilização na bibliotecas do jQuery, para o Android foi necessário 31 linhas de código fonte e 58 minutos de desenvolvimento e para o Windows Phone doi necessário 33 linhas de código fonte e 59 minutos de desenvolvimento, a construção desta funcionalidade utilizando o PhoneGap cross-plataforma com o KLOC de 1,0625 linha/min mostrou-se com o custo menor ao Android nativo com o KLOC de 0,534483 linha/min e o Windows Phone Nativo com o KLOC de 0,559322034 linha/min.

Tabela 4 – Comparação Comunicação Ws

Aplicativo	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Windows Phone Nativo	33	59	0,559322034
Android Nativo	31	58	0,534482759
PhoneGap	17	16	1,0625

Fonte: Do autor (2016)

j) em uma visão geral para a construção de todos os métodos conforme a tabela 5 foi necessário 111 linhas de código fonte e 123 minutos de desenvolvimento, os métodos em PhoneGap mostraram-se menores e de simples de desenvolver, comparado com o desenvolvimento dos métodos nativos em Android com 179 linhas de código fonte e 218 minutos de desenvolvimento e Windows Phone com 124 linhas de código fonte e 181 minutos de desenvolvimento, assim com um custo de desenvolvimento em 0.902439024 linhas por minuto o desenvolvimento em PhoneGap mostrou-se com um custo menor de desenvolvimento comparado com o Android nativo com 0,821100917 linhas por minuto e com o Windows Phone com 0,685082873 linhas por minuto.

Tabela 5 – Comparação TOTAL Desenvolvimento

Aplicativo	**Quantidade de linhas**	**Tempo de desenvolvimento em minutos**	**KLOC**
Windows Phone Nativo	124	181	0,685082873
Android Nativo	179	218	0,821100917
PhoneGap	111	123	0,902439024

Fonte: Do autor (2016)

6.7.3 Analise comparativa de desempenho

Considerando os dados obtidos após a aplicação de carga de exclusão nas funções de testes desenvolvidas, conforme as tabelas 6 e 7, identificou-se que para os testes de comparativos de desempenho entre os aplicativos cross-plataforma PhoneGap com o Android e Windows Phone nativo as aplicativos nativas mostraram-se com o desempenho superior aos aplicativos cross-plataforma, isto ocorre pois os aplicativos em PhoneGap rodam na camada de WebView, assim todos os processos tem que passar por essa camada antes de chegar ao SO nativo,

já as aplicações nativas rodam junto com o SO nativo diminuindo o tempo de resposta, e melhorando o desempenho do aplicativo.

Tabela 6 – Desempenho Android

	Nativo	**Cross-Plataforma**
Acesso a câmera	616ms	784ms
Acesso ao GPS	77ms	98ms
Acesso ao WS	414ms	569ms

Fonte: Do autor (2016)

Tabela 7 – Desempenho Windows Phone

	Nativo	**Cross-Plataforma**
Acesso a câmera	376ms	483ms
Acesso ao GPS	66ms	76ms
Acesso ao WS	234ms	490ms

Fonte: Do autor (2016)

Após a análise comparativa entre os aplicativos nativos e os cross-plataforma PhoneGap identificou-se que o desenvolvimento em PhoneGap tem um custo menor e tem um tempo de desenvolvimento reduzido mas quando estamos falando de estabilidade e desempenho o mesmo se mostra inferior aos aplicativos nativos, os aplicativos nativos apesar de terem um custo maior no desenvolvimento mostraram-se estáveis e com um melhor desempenho, nos quesitos de manutenção do aplicativo as nativas se mostraram-se melhor, pois as mesmas tem IDEs especificas para o seu desenvolvimento e várias ferramentas para auxiliar o desenvolvimento, já o PhoneGap não tem uma IDE especifica assim e desenvolvido normalmente em editores de texto simples como NotPad++ e Sublime Text.

7 Conclusão

A aplicação da análise comparativa entre aplicativos mobile nativo e cross-plataforma contribui para o início de desenvolvimentos com tecnologia mobile. Assim como um produto necessita de análise de regra de negócio antes de passar pelo processo de desenvolvimento, e necessário decidir a melhor forma de se desenvolver o mesmo, para que no futura não sejam identificado pontos falhos no desenvolvimento.

Para este trabalho o foco foi a comparação entre aplicativos nativos e cross-plataformas em Android e Windows Phone, visando apresentas os custos de desenvolvimento em KLOC e desempenho. Foi possível perceber a evolução da computação e quanto o desenvolvimento de aplicações mobile nativa e cross-plataforma estão evolução, além de conhecer os recursos de desenvolvimentos em PhoneGap, Android e Windows Phone.

A mescla de duas formas de análise, usando as metodologias de métricas orientada ao tamanho e análise de sobrecarga de execução foi o tema deste trabalho e mostrou eficiência para a aplicação no desenvolvimento de aplicativos nativo e cross-plataforma. A comparação dos resultados obtidos na métrica orientada ao tamanho mostrou que o custo de desenvolvimento para um aplicativo cross-plataforma em PhoneGap e menor comparado ao desenvolvimento dos aplicativos nativos, mas em relação ao desempenho ele se mostrou inferior aos aplicativos nativos. Além disso, com a análise comparativa de custo de desenvolvimento e desempenho, foi possível identificar que para o desenvolvimento de aplicativos voltado ao desempenho e uso de hardware do dispositivo a melhor

pratica seria utilizar a tecnologia nativa, pois apresentou resultados melhores e instáveis.

A metodologia resultante da mescla das análise comparativa de custo e de desempenho da comparação de aplicativos cross-plataforma e nativo, podem ser utilizados como base para desenvolvimento de novos aplicativos considerando o desempenho e o custo de desenvolvimento e fornecer aplicativos instáveis e com melhor resultado ao usuário.

Como sugestão de trabalhos futuros, podem-se: aplicar ou mesclar outras métricas na análise comparativa, inclusive os descritos neste trabalho; aplicar a metodologia dessa pesquisa em outras funcionalidades ou outras plataformas; avaliar outros tipos de desenvolvimento cross-plataforma comparado com o desenvolvimento nativo.

REFERENCIAS

ABLESON, W. Frank et al. **Android em ação.** 3.ed. Rio de Janeiro: Elsevier, 2012. 656p.

CARDOSO, Gabriel Schade. **Criando aplicação para seu Windows Phone.** São Paulo: Casa do Código, 2014. 149 p.

CAFÉ, Adriel Almeida. **Desenvolvimento de Cross-Platform Mobile Apps Utilizando o Titanium Mobile.** 2012. 19 f. Tese (Doutorado) - Curso de Ciências da Computação, Faculdade Zacarias de Góes, Valença, 2012.

CHOU, Tim. **Os custos ocultos de Software.** Tradução nossa. Disponível em: <http://www.datamation.com/entdev/article.php/11070_2214031_2/The-Hidden-Cost-of-Software.htm>. Acesso em: 15 nov. 2015.

ESPADA, Jordán. **Using extended web technologies to develop bluetooth multiplataforma mobile application fron interact with smart thigs.** Madrid: Elsevier, 2013. 12 p.

GOMES, Alvaro Eduardo. **Métricas e Estimativas de Software:** O início de um rally de regularidade. Disponível em: <http://www.linhadecodigo.com.br/artigo/102/metricas-e-estimativas-de-software-o-inicio-de-um-rally-de-regularidade.aspx>. Acesso em: 15 nov. 2015.

KESSIN, Zachary. **Programming HTML5 Applications.** New York: O'reilly Media, 2011. 144 p.

LEE, Wei-meng. **Beginning Android™ Application Development**. Indianápolis: Wiley, 2011.

MONACO, Thiago; CARMO, Rodolpho Marques do. **Desenvolvendo Aplicações para Windows Phone.** Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2012. 453 p.

MUNRO, Jamie. **20 Recipes for Programming PhoneGap.** Sebastopol,: O'reilly Media, 2012.

PETZOLD, Charles. **Programming Windows Phone 7.** Washington: Microsoft Corporation, 2010. 997 p. Tradução nossa.

PHONEGAP. **PhoneGap Developer App Reference Guide.** Disponível em: <http://docs.phonegap.com/references/developer-app

PRESSMAN, Roger S**. Engenharia de Software**, São Paulo: Makron Books, 1995.

PREZOTTO, Ezequiel Douglas; BONIATI, Bruno Batista. **Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas.** 2014. 8 f. TCC (Graduação) - Curso de Tecnologia em Sistemas Para Internet, Universidade Federal de Santa Maria, Rio Grande do Sul, 2014.

SOMMERVILLE, I. **Engenharia de Software**. 6ª Ed. São Paulo: Addison Wesley, 2003.