INSTITUTO DOCTUM DE EDUCAÇÃO E TECNOLOGIA

FACULDADES INTEGRADAS DE CARATINGA

CIÊNCIA DA COMPUTAÇÃO

FRAMEWORKS DE DESENVOLVIMENTO *MOBILE* MULTIPLATAFORMA – UMA ANÁLISE A PARTIR DA ISO 9126

RONAN COSTA ARAUJO

Caratinga
2015

Ronan Costa Araujo

FRAMEWORKS DE DESENVOLVIMENTO *MOBILE* MULTIPLATAFORMA – UMA ANÁLISE A PARTIR DA ISO 9126

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga como requisito parcial para obtenção do título de bacharel em Ciência da Computação orientada pelo Prof. Msc. Bruno Vieira Becattini.

Caratinga

2015

FACULDADES INTEGRADAS DE CARATINGA

TRABALHO DE CONCLUSÃO DE CURSO

TERMO DE APROVAÇÃO

TÍTULO DO TRABALHO

***FRAMEWORKS* DE DESENVOLVIMENTO *MOBILE* MULTIPLATAFORMA – UMA ANÁLISE A PARTIR DA ISO 9126**

por

Ronan Costa Araujo

Este Trabalho de Conclusão de Curso foi apresentado perante a Banca de Avaliação composta pelos professores Bruno Vieira Becattini, Wanderson Miranda Nascimento e Glauber Luis da Silva Costa, às 20 horas e 20 minutos do dia 14 de dezembro de 2015 como requisito parcial para a obtenção do título de bacharel. Após a avaliação de cada professor e discussão, a Banca Avaliadora considerou o trabalho aprovado, com a qualificação: ÓTIMO.

Trabalho indicado para publicação: (X)SIM ( )NÃO

Caratinga, 14 de DEZEMBRO de 2015

Bruno Vieira Becattini

Professor Orientador e Presidente da Banca

Professor Avaliador 1

Professor Avaliador 2

Ronan Costa Araujo

Aluno(a)

Coordenador (a) do Curso

AGRADECIMENTOS

A Deus pelas oportunidades e bênçãos que me foram dadas na vida, e também por momentos difíceis que foram fundamentais para o aprendizado.

Agradeço aos meus pais e irmãos, meus maiores exemplos, pelo amor, incentivo e apoio durante esta caminhada. Aos meus amigos, que durante essa jornada estiveram sempre comigo transmitindo conhecimento e sempre ajudando uns aos outros. A minha namorada e companheira por compartilhar comigo esse momento, pelo carinho, alegria, compreensão, amor e todo apoio durante o curso e o desenvolvimento deste trabalho.

Ao meu orientador Bruno Vieira, que ao longo deste trabalho sempre se mostrou presente e disponível para ajudar e encontrar a melhor solução para os problemas.

Aos meus professores por todo ensinamento, apoio e dedicação durante o curso e em especial a professora Fabrícia Pires por suas inúmeras contribuições.

RESUMO

Frameworks de desenvolvimento *mobile* multiplataforma são ferramentas que possibilitam a criação de um mesmo aplicativo para diferentes sistemas operacionais de uma maneira descomplicada e com grande economia. Utilizando apenas um código-fonte ou uma mesma linguagem de programação no desenvolvimento de uma aplicação móvel é possível criar uma aplicação para ser utilizada em diferentes plataformas móveis (Android, Windows Phone, Android, iOS, BlackBerry). Existe uma grande variedade de *frameworks* no meio comercial e acadêmico, dos quais, Cordova e Xamarin são exemplos dos que mais se destacam. Cada um desses *frameworks* em destaque possuem suas próprias características no desenvolvimento de seus aplicativos, abordagem, disponibilização de recursos, documentação e qualidade do aplicativo final.

O objetivo deste trabalho foi desenvolver um aplicativo nos *frameworks* Cordova e Xamarin para as plataformas Android e Windows Phone e realizar uma análise comparativa dos aplicativos e desses *frameworks*, utilizando critérios da ISO/IEC 9126.

Para realizar a análise comparativa de cada *framework* foram selecionadas dez métricas de software e posteriormente após realizar a análise e comparação dos dados, concluiu-se que para o aplicativo desenvolvido neste trabalho o *framework* Cordova é o mais eficiente nas duas plataformas abordadas.

Palavras-chave: *Framework*, Multiplataforma, Dispositivos móveis, Cordova, Xamarin, Android, iOS, Windows Phone.

ABSTRACT

Mobile platform development frameworks are tools that enable you to create the same application for different operating systems in an uncomplicated manner and with great savings. Using a single source or a single programming language in the development of a mobile application you can create an application for use in various mobile platforms (Android, Windows Phone, Android, iOS, BlackBerry). There is a wide variety of frameworks in business and academia, of which Cordova and Xamarin are examples of that stand out. Each of these featured frameworks have their own characteristics in the development of your applications, approach, availability of resources, documentation, and quality of the final application.

The objective of this study was to develop an application in Cordova and Xamarin frameworks for Android and Windows Phone platforms and perform a comparative analysis of these applications and frameworks, using criteria of ISO / IEC 9126.

To perform the comparative analysis of each framework have been selected ten software metrics and then after performing the analysis and comparison of data, it was concluded that for the application developed in this work Cordova framework is most efficiently addressed in the two platforms.

Keywords: Framework, Multiplatform, mobile devices, Cordova, Xamarin, Android, iOS, Windows Phone.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

IDC – International Data Corporation

SDK – Software Development Kit

SO – Sistema Operacional

PDA – Personal Digital Assistants

OHA – Open Handset Alliance

VB – Visual Basic

GPS – Global Positioning System

URL – Universal Resource Location

ISO – International Organization for Standardization

IEC – International Electrotechnical Comission

SUMÁRIO

INTRODUÇÃO

Com os avanços na comunicação e na tecnologia da informação, os dispositivos móveis vem ganhando destaque, tornando-se mais poderosos em vários aspectos. Os dispositivos estão cada vez mais potentes, em relação a desempenho e capacidade de armazenamento, e também mais difundidos e acessíveis aos usuários.

O desenvolvimento de aplicativos móveis é um processo trabalhoso, pois cada dispositivo possui uma configuração diferente, como: sistema operacional, tamanho da tela e processamento. Um bom aplicativo deve funcionar em diversas plataformas de diferentes fabricantes e sistemas operacionais.

Os *frameworks* surgiram para auxiliar e maximizar o desenvolvimento de *software* para diversas plataformas. Cada *framework* possui suas próprias metodologias e particularidades durante o desenvolvimento de suas aplicações, e é de suma importância realizar o desenvolvimento de um aplicativo para avaliar e realizar os testes necessários de como cada *framework* se comporta e utiliza os recursos dos dispositivos móveis. Foi desenvolvido um aplicativo resultante para cada *framework* dando ênfase no processo com que cada *framework* acessa as funcionalidades do dispositivo. O aplicativo desenvolvido teve como objetivo utilizar os recursos dos dispositivos móveis: câmera, GPS, *bluetooth* e mensagens de alerta.

Para desenvolver a análise proposta foi necessário realizar uma pesquisa bibliográfica a fim de se obter informações sobre o desenvolvimento de aplicativos móveis multiplataforma. Dando sequência, foi realizado um estudo sobre os *frameworks* de desenvolvimento *mobile* multiplataforma mais utilizados no ano de 2015. Os *frameworks* selecionados foram: Cordova e Xamarin.

No desenvolvimento do trabalho e obtenção dos resultados esperados foi necessário realizar a análise comparativa dos *frameworks* Cordova e Xamarin e dos aplicativos finais gerados por cada um segundo critérios da ISO 9126.

Para desenvolver a análise proposta, foram realizadas atividades de revisão bibliográfica na norma ISO/IEC 9126 que auxilia na avaliação da qualidade de software. Tendo como sequência o estudo e seleção das métricas de qualidade de

software propostos pela ISO 9126 onde foram selecionadas e especificadas quais características e critérios de avaliação mais pertinentes para serem aplicados na análise comparativa.

Para realizar a comparação, foi implementado um aplicativo utilizando os *frameworks* Cordova e Xamarin, com o objetivo de identificar as principais características de cada *framework* e através da análise comparativa determinar qual *framework* é mais vantajoso, em determinada situação e para qual tipo de aplicação.

Em seguida foi feito um estudo do método de desenvolvimento de cada *framework* selecionado, visando evidenciar as características mais relevantes, tipos de desenvolvimento e o padrão de desenvolvimento final de cada aplicativo resultante desenvolvido.

Para realizar a análise dos *frameworks* selecionados e dos aplicativos desenvolvidos em cada um deles foram selecionadas dez métricas de software e após realizar a análise e a comparação dos dados obtidos, concluiu-se que o *framework* Cordova é o mais eficiente nas duas plataformas abordadas.

No capítulo 1 deste trabalho é exposto uma introdução sobre a história dos dispositivos móveis, em seguida é mostrado o mercado de dispositivos móveis e sistemas operacionais móveis. Dando continuidade no capítulo 1 também é exposto os principais conceitos do que é um aplicativo móvel, sistema operacional móvel e falando também sobre os sistemas operacionais Android e Windows Phone que foram selecionados para o desenvolvimento dos aplicativos.

Na seção 1.2 é exposto as estratégias de desenvolvimento que um aplicativo móvel pode ser desenvolvido e também é falado sobre os *frameworks* utilizados neste trabalho: Cordova e Xamarin. Na seção 1.4 é exposto os conceitos de qualidade de software, em seguida na seção 1.5 é apresentado o modelo de qualidade da ISO 9126.

O capítulo 2 apresenta o desenvolvimento deste trabalho, a metodologia desenvolvida, a descrição dos aplicativos desenvolvidos, a descrição do hardware utilizado para desenvolver os aplicativos e realizar os testes, e as métricas de software selecionadas para realizar a análise comparativa dos *frameworks*.

O capítulo 3 apresenta os resultados obtidos após a aplicação das métricas selecionadas e a conclusão final deste estudo onde é exposto a discussão final e os resultados finais obtidos.

1. REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os aspectos teóricos relacionados ao presente trabalho. A seção 1.1 apresenta o estudo inicial feito sobre o desenvolvimento de aplicações móveis. A seção 1.2 apresenta os métodos de desenvolvimento multiplataforma. Em seguida, na seção 1.3 é abordado o modelo de desenvolvimento multiplataforma. Finalizando este capítulo, a seção 1.4 apresenta as métricas de qualidade de software propostas pela ISO/IEC 9126.

1.1 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

Existe uma grande diversidade de dispositivos e sistemas operacionais móveis no mercado. O desenvolvimento de aplicativos para esses dispositivos não é uma atividade simples devido essa grande diversidade.

Os tópicos a seguir apresentam uma pesquisa sobre a história dos dispositivos móveis, aplicativos móveis e sistemas operacionais móveis dando ênfase nos sistemas operacionais Android e Windows Phone abordados neste trabalho.

1.1.1 Dispositivos móveis

Com os avanços ocorridos nas tecnologias de comunicação sem fio, a popularização dos dispositivos móveis (*tablets,* celulares*, smartphones*) possibilitaram novos cenários e atividades de auxílio a atividades cotidianas e profissionais em várias áreas da sociedade. Os usuários passaram a difundir e acessar aplicações móveis e vários serviços oferecidos (jogos, *e-mails*, redes sociais, editores de texto, aplicativos multimídia) mesmo se deslocando, em praticamente todos os lugares, a qualquer hora e local. Desde o lançamento e

sucesso do primeiro modelo do *iPhone* em 2007, tornou-se quase um padrão a produção de dispositivos com tela sensível ao toque, com vários sensores, multifuncionais, recursos multimídia e interfaces mais amigáveis que aperfeiçoaram a experiência do usuário mediante o uso dessas aplicações (CÂMARA, 2015).

Dispositivo móvel é todo equipamento, que pode ser transportado a qualquer lugar. Quanto menos depender de características físicas, maior será seu grau de mobilidade. Por exemplo, a bateria: quanto maior a duração e menor o tempo para recarregar, maior a mobilidade promovida pelo dispositivo. São classificados dispositivos móveis os celulares, *tablets* e outros dispositivos desse gênero que oferecem mobilidade e autonomia (MORIMOTO, 2009).

O desenvolvimento de aplicativos para os dispositivos móveis, no entanto, não é uma atividade simples devido a diversidade de dispositivos móveis existentes. A variedade de configurações (fabricantes, tamanhos de telas, capacidades de processamento, sistemas operacionais) tornam o desenvolvimento, o teste e a portabilidade das aplicações trabalhosas e inconvenientes (WEINBERG, 2011).

A heterogeneidade de várias plataformas e sistemas operacionais ocasionou grandes problemas para empresas de desenvolvimento de aplicativos e também para desenvolvedores autônomos. Para qual plataforma será melhor desenvolver os aplicativos? Quais aparelhos são os mais utilizados? Será melhor desenvolver aplicativos para várias plataformas e modelos de aparelho a fim de atingir um público-alvo maior?

Cada plataforma possui suas próprias metodologias de desenvolvimento de software as quais permitem o desenvolvimento de aplicações exclusivamente destinadas a seu sistema operacional utilizando sua linguagem de programação, dessa forma, Objective-C para iOS (APPLE, 2015), C# para Windows Phone (MICROSOFT, 2015) e Java para o Android (GOOGLE, 2015).

Com o objetivo de facilitar e potencializar o processo de desenvolvimento de um mesmo aplicativo para diversas plataformas surgiram diversos *frameworks* de desenvolvimento de aplicativos móveis multiplataforma. Cada *framework* tem suas próprias características no processo de desenvolvimento e restrições nas aplicações resultantes. Entre os *frameworks* mais utilizados, podemos evidenciar o Xamarin e Cordova, que são responsáveis por 70% da opção primária para a maioria dos desenvolvedores que usam desenvolvimento multiplataforma (VISION MOBILE, 2015).

No entanto, qual desses *frameworks* é o mais adequado para utilização? Qual *framework* os desenvolvedores devem utilizar para seu projeto? Para responder essa questão será necessário conhecermos a documentação e a metodologia de desenvolvimento que cada um deles aborda para assim estarmos preparados a decidir qual é mais vantajoso, em determinada situação e para qual tipo de aplicação.

1.1.2 Mercado de dispositivos móveis

De acordo com a última versão da pesquisa da *International Data Corporation* (IDC), realizada em 23 de julho de 2015, fornecedores fabricaram um total de 337,2 milhões de smartphones em todo o mundo no segundo trimestre de 2015. Um aumento de 11,6% a partir dos 302,1 milhões de unidades no segundo semestre de 2014. O crescimento global do mercado de smartphones não só foi impulsionado pelo sucesso dos dispositivos das empresas Samsung, Apple e outros, mas pela abundância de aparelhos a preços acessíveis que continuam a conduzir a produção em mercados chave importantes. IDC(2015).

A IDC, é a principal empresa provedora global de inteligência de mercado, consultoria e eventos para os mercados de tecnologia da informação, serviços de telecomunicações e tecnologia de consumo. Com cerca de 1.100 analistas de sistema em todo o mundo, a IDC é especialista regional e local sobre tecnologia e oportunidades e tendências da indústria em mais de 110 países. Seus serviços ajudam os profissionais de TI, investidores e executivos a tomarem decisões baseadas em fatos atuais da tecnologia para atingirem seus objetivos de negócio.

IDC (2015), também destaca em sua pesquisa que o crescimento global do mercado de smartphones não foi só impulsionado pelo sucesso de dispositivos emblemáticos *premium* da Samsung, Apple e outros, mas também pela abundância de aparelhos a preços acessíveis que continuam a conduzir a fabricação em muitos mercados importantes. A medida em que a produção de dispositivos móveis de recurso continuam a diminuir, os fornecedores começaram a investir em ambos os

mercados emergentes e desenvolvidos com smartphones competitivos que são ricos em recursos e de baixo custo.

Embora grande parte da atenção esteja sendo dada para a Apple e Samsung na camada superior, o mercado de smartphones na verdade continua a diversificar a medida que mais fabricantes entram neste mercado o tornando cada vez mais competitivo (IDC, 2015).

A Tabela 1 mostra a lista dos fornecedores de smartphones em destaque.

Tabela 1 - Lista de fornecedores de smartphones em destaque

Fabricante	**2015 – 2º sem. (milhões)**	**2015 – 2º sem. Quota de mercado**	**2014 – 2º sem. (milhões)**	**2014 – 2º sem. Quota de mercado**
Samsung	73.2	21.7%	74.9	24.8%
Apple	47.5	14,1%	35.2	11.7%
Huawei.	29.9	8.9%	20.2	6.7%
Xiaomi	17.9	5.3%	13.8	4.6%
Lenovo	16.2	4.8%	15.8	5.2%
Outros	152.5	45.2%	142.2	47.1%
Total	**337.2**	**100.%**	**302.1**	**100.0%**

Fonte: Tradução do próprio Autor. Fonte: IDC, 2015.

Como mostrado na Tabela 1, a Samsung manteve a liderança no mercado de smartphones em todo o mundo. A Apple, em segundo lugar da lista continua a dominar no mercado chinês graças a rápida expansão das redes 4G, como afirma IDC (2015) em sua pesquisa.

A Tabela 2 apresenta uma pesquisa realizada pela *International Data Corporation* sobre o mercado de smartphones no ano de 2015.

Tabela 2 - Mercado de smartphones em todo o mundo

Período	**Android**	**iOS**	**Windows Phone**	**BlackBerry**	**Outros**
2015 – 2º sem.	82.8%	13.9%	2.6%	0.3%	0.4%
2014 – 2º sem.	84.8%	11.6%	2.5%	0.5%	0.7%
2013 – 2º sem.	79.8%	12.9%	3.4%	2.8%	1.2%
2012 – 2º sem.	69.3%	16.6%	3.1%	4.9%	6.1%

Fonte: Tradução do próprio Autor. Fonte: IDC, 2015.

De acordo com pesquisa realizada pela empresa IDC apresentada na Tabela 2, no ano de 2015, o comércio de smartphones em todo mundo cresceu 13% ao ano, com 341,5 milhões de novas fabricações. Android domina o mercado com uma quota de 82,8% em 2015.

A Apple obteve grande sucesso graças ao apetite insaciável dos seus consumidores por dispositivos com maiores telas. A popularidade do iPhone 6 Plus continuou em muitos mercados-chave, incluindo a China, onde o mercado global de smartphones teve um crescimento de 6,7% (IDC, 2015).

1.1.3 Aplicativo móvel

Um aplicativo móvel é um *software* destinado a ser executado em um dispositivo móvel, como em um *smartphone*, e é obtido normalmente através das lojas de aplicativos existentes em cada plataforma. Este padrão de desenvolvimento e distribuição se deu com o lançamento do iPhone SDK (*Software Development Kit)* em março de 2008 e início da App Store junto ao lançamento do *iPhone* 3G em julho de 2008 como resposta a quantidade de *jailbreak* (desbloqueio de restrições) existente, até então a única forma de instalar aplicações de terceiros (9TO5MAC STAFF, 2011).

1.1.4 Sistemas Operacionais Móveis

Segundo Janssen (2012), um sistema operacional móvel é um sistema construído exclusivamente para um dispositivo móvel, como um smartphone, assistente digital pessoal (PDA), *tablet*, ou outro sistema operacional móvel integrado. Os sistemas operacionais móveis mais populares são: Android, Symbian, iOS, BlackBerry OS e Windows Mobile.

Um sistema operacional móvel é responsável por identificar e definir recursos do dispositivo móvel e funções, incluindo teclados, sincronização de aplicativos, e-

mail e mensagens de texto. Um sistema operacional móvel é semelhante a um sistema operacional padrão (como Windows, Linux e Mac), mas é relativamente simples e leve. Gerencia principalmente as ligações locais, conexões de rede sem fio e de banda larga, serviços multimídia, entre outros métodos de entrada. (JANSSEN, 2012).

1.1.4.1 Android

O Android é uma plataforma desenvolvida pela Google para dispositivos móveis. Em 5 de novembro de 2007, a empresa publicou a primeira plataforma Open Source de desenvolvimento para dispositivos móveis baseada na plataforma Java com sistema operacional Linux, que foi chamada de Android. Mantida pela OHA (Open Handset Alliance), um grupo de mais de 40 empresas que se uniram para inovar e acelerar o desenvolvimento de aplicativos e serviços, trazendo para os consumidores uma experiência melhor. A plataforma Android é a primeira plataforma móvel completa, aberta e livre (ALVES, 2010).

O Android possui um kit de desenvolvimento de software (SDK), um pacote de código aberto disponível para download para Windows, Mac OS X e Linux, mostrando que a linguagem Java é de fato a principal linguagem de programação em execução nos telefones baseados em Linux. O software Android possui a máquina virtual Dalvik, criada pelo Google para realizar a execução de programas Java, um navegador baseado no *WebKit* motor, e suporte para várias mídias e formatos de imagens.

WebKit é um motor de navegador de código aberto que foi desenvolvido pela Apple Inc. e é utilizado atualmente em navegadores de internet como: Google Chrome, Apple Safari e no navegador padrão do Android. Ele faz a renderização de páginas web em navegadores como Google Chrome e Safari. (HAWKES, 2013).

O SDK inclui um emulador para que os programadores possam desenvolver softwares, mesmo sem possuir o hardware do telefone. Também possui API’s (interfaces de desenvolvimento de aplicativos) que permitem que os programadores aproveitem vários serviços disponíveis nos dispositivos, tais como: vídeos, *streaming*

de áudio, reprodução de gráficos 3D, rede sem fio e *bluetooth* (SHENKLAND, 2009).

A Figura 1 mostra o funcionamento interno do sistema Android para telefones celulares:

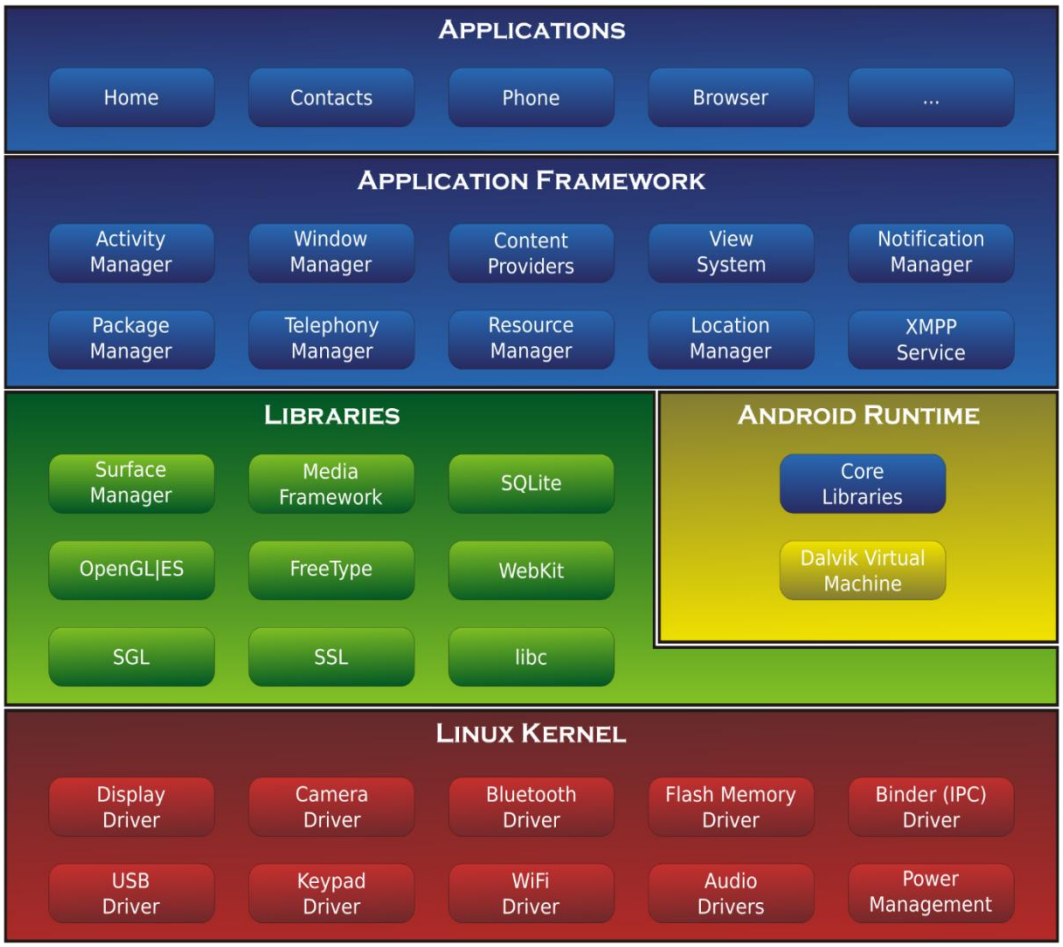

Figura 1 - Arquitetura de funcionamento interno do Sistema Android.

Fonte: GOOGLE, 2015.

1.1.4.2 Windows Phone

Windows Phone é o sistema operacional *mobile* desenvolvido pela Microsoft. Lançado em 21 de outubro de 2010 e substituiu a antiga plataforma *Windows Mobile*. Sua interface é simples, segue o modelo do estilo Metro, adotado no Windows 8 e é considerada a que oferece uma experiência desktop aos seus usuários melhor do que qualquer outro smartphone.

Para desenvolver aplicações para Windows Phone, os desenvolvedores podem utilizar as linguagens VB, C# ou C++. O sistema operacional está na versão 8.1 e é sucessor do *Windows Mobile*, entretanto não é compatível com as aplicações do antigo sistema operacional (ALLEN et. al. 2015).

A Figura 2 mostra a arquitetura de funcionamento do Windows Phone.

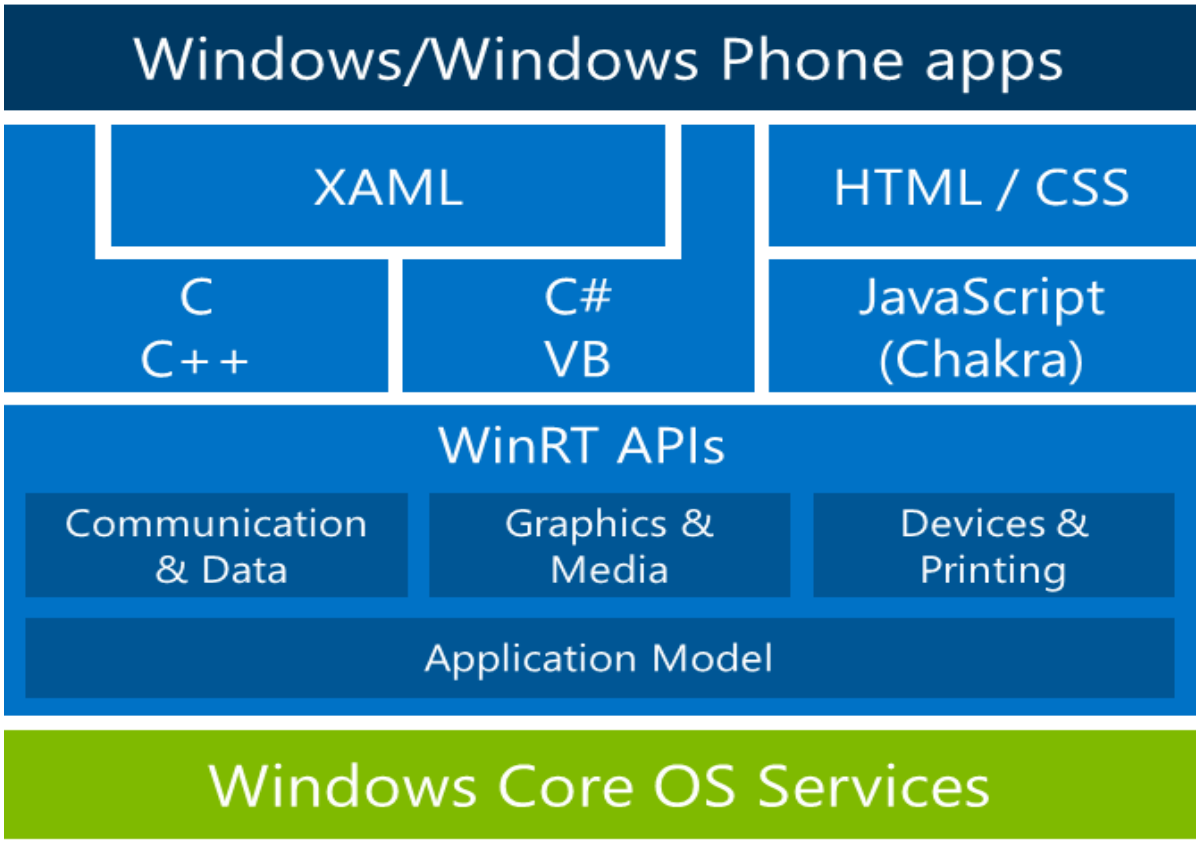

Figura 2 - Arquitetura do sistema Windows Phone.

Fonte: Microsoft, 2015.

1.2 MÉTODOS DE DESENVOLVIMENTO

Um aplicativo para um dispositivo móvel pode ser implementado utilizando três estratégias de desenvolvimento: Desenvolvimento nativo, no qual utiliza linguagem de programação e ferramentas específicas que são fornecidas pela empresa responsável por cada plataforma; Desenvolvimento Web, onde é utilizado

linguagem HTML5, CSS e JavaScript; Desenvolvimento Híbrido, que utiliza linguagem de programação chamada de script que vai ser interpretada através de uma camada do sistema ou montada no aplicativo, que tem o papel de interagir com o sistema (CHARL, 2011).

1.2.1 Aplicativo nativo

Os aplicativos nativos residem no dispositivo smartphone/tablete e podem ser acessados através de ícones na tela principal. Eles são instalados através de um aplicativo de Loja (como Google Play do Android e App Store da Apple). São desenvolvidos especificamente para tal plataforma, podem aproveitar todas as funcionalidades do sistema operacional do dispositivo, como: bússola, câmera, GPS, acelerômetro, lista de contatos, etc. Também é possível utilizar sistemas de notificação nativos do sistema operacional e funcionar sem conexão com a internet caso o conteúdo esteja embarcado. (AMBROS, 2013).

1.2.2 Aplicativo web

Aplicativos *web,* ou *web apps*, não são aplicativos raiz. São sites que aparecem como um aplicativo nativo. Eles são executados através de um navegador e tipicamente escritos em HTML5. Os usuários o acessam como fariam como um site: acessam determinada URL e tem a opção de “instalá-lo” na tela principal do seu dispositivo criando um atalho para aquela página. (AMBROS, 2013).

1.2.3 Aplicativo híbrido

Os aplicativos híbridos são parcialmente nativos e parcialmente *web apps*. Como os nativos, eles devem ser baixados através de um aplicativo de loja (como

Google Play do Android e App Store da Apple), ficam armazenados na tela principal do dispositivo e podem aproveitar todas as funcionalidades do dispositivo (câmera, GPS, acelerômetro, gestos, etc.). Como *web apps*, eles podem ser baseados em HTML5, CSS3 e JavaScript, exibidos através de um navegador embutido no aplicativo, tendo parte ou conteúdo total carregado da web. Os aplicativos híbridos são populares porque permitem o desenvolvimento multiplataforma, utilizando o mesmo código-fonte para diferentes sistemas operacionais através de ferramentas como Cordova, PhoneGap, Sencha Touch e Xamarin que permitem, compilar para o formato nativo de tal plataforma, reduzindo custos de produção. (AMBROS, 2013).

1.3 DESENVOLVIMENTO MULTIPLATAFORMA

As aplicações estão ficando diversificadas e as bases de usuários estão se expandindo, não há uma plataforma clara de escolha. Diante desse empasse, os desenvolvedores de aplicativos são confrontados com o debate sobre se eles deveriam estar projetando aplicações multiplataforma ou vários aplicativos para diferentes plataformas.

Os desenvolvedores hoje em dia estão buscando um público-alvo de duas principais plataformas: Android e iOS. Há dois objetivos principais de um aplicativo multiplataforma: adquirir muitos clientes ou oferecer melhor qualidade dentro de um mercado alvo. Quando a maioria de seu público-alvo está usando a mesma plataforma, a escolha para qual será desenvolvida se torna fácil. Mas quando estamos atendendo diversas plataformas, há uma necessidade de desenvolver para múltiplas plataformas. CYGNIS (2013).

Cygnis (2013) também cita as principais vantagens do desenvolvimento multiplataforma:

- Maior alcance de consumidores;
- Mesmo código para várias plataformas;
- Fácil marketing;
- Aparência unificada;
- Uso de tecnologias conhecidas;

- Redução de custo no desenvolvimento.

Cygnis Media (2013) também cita algumas desvantagens, como:

- Possíveis particularidades em relação à interface de usuário para cada plataforma;
- Dificuldades no uso de recursos nativos de cada plataforma;
- Dificuldade para atingir desempenho semelhante entre plataformas.

1.3.1 Frameworks de desenvolvimento

Para MATTSON (2000), um *framework* é uma arquitetura feita com o objetivo de obter a máxima reutilização, representada com um conjunto de classes abstratas e concretas com o objetivo de resolução de uma família de problemas.

Segundo RENE (2014), o objetivo de um *framework* é aumentar a produtividade e minimizar os esforços gerados no desenvolvimento. Isso gera tempo para que os desenvolvedores *front-end* por exemplo, se preocupem com outras questões mais importantes. Hoje em dia, todas as linguagens, pelo menos as mais utilizadas, possuem frameworks para auxiliar o desenvolvedor. Com o constante crescimento do mercado *mobile* e do HTML5, CSS3 e JavaScript, cresceram o número de frameworks para dispositivos móveis, ou *mobile*, sejam eles: Android, iOS, Windows Phone, que são as marcas dominantes no mercado atualmente.

O uso de *frameworks* de desenvolvimento *mobile* multiplataforma, permite que os desenvolvedores criem um único aplicativo para diversos sistemas operacionais móveis de uma maneira simples e utilizando grande reúso de código-fonte. Utilizando apenas um código fonte base é possível através dos *frameworks*, criar uma aplicação pronta para utilização em vários sistemas operacionais móveis (*Android*, *Windows Phone*, *iOS*, *BlackBerry OS*).

Desenvolver um aplicativo para diversas plataformas com o objetivo de atender a maioria dos dispositivos, requer retrabalho, pois é grande a diversidade de

plataformas e linguagens utilizadas no desenvolvimento de aplicações nativas (NUNES, 2013).

De acordo com a *Vision Mobile* (2015), o surgimento de *frameworks* multiplataforma está em andamento. Melhorias de hardware em dispositivos móveis estão compensando os problemas de desempenho, assim os desenvolvedores e fornecedores estão aprendendo como produzir uma boa experiência com o usuário sem desenvolver um aplicativo totalmente nativo.

Vision Mobile (2015) também cita em um relatório realizado em 2015, onde é analisado em profundidade o mercado de aplicativos multiplataforma, com base em dados de pesquisa de 8 mil desenvolvedores de aplicativos e análise de 185 mil aplicativos no Google Play Store. O relatório analisa os prós e contras dos *frameworks* multiplataforma, o comportamento dos desenvolvedores em relação a eles, e prevê o futuro do desenvolvimento multiplataforma. Este relatório mostra que Cordova e Xamarin são os *frameworks* líderes estabelecidos no mercado de desenvolvimento multiplataforma. O relatório informa quais ferramentas se destacam, quais os desenvolvedores estão utilizando, e porquê escolhê-las.

A Figura 3 mostra os três melhores *frameworks* multiplataforma de acordo com a Vision Mobile.

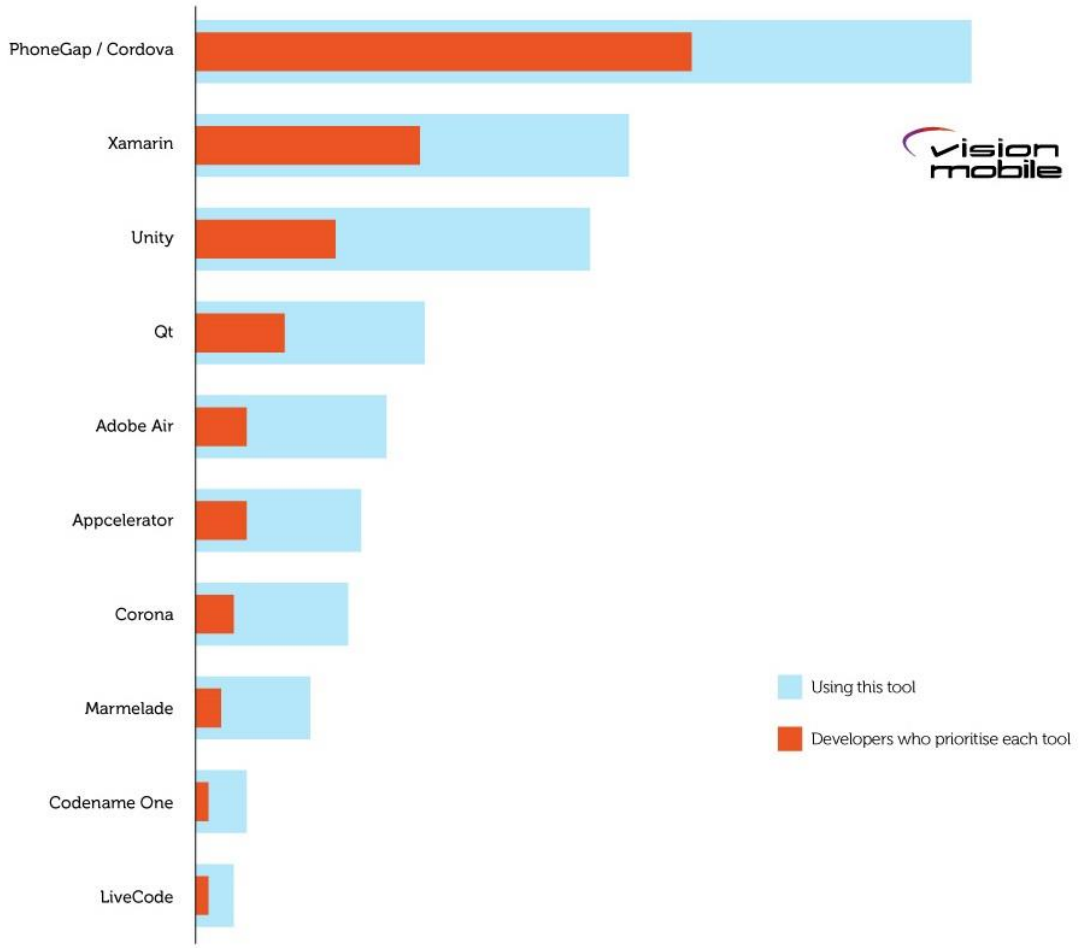

Figura 3 - Os três melhores *frameworks* multiplataforma.

Fonte: IDC, 2015

A Figura 3 mostra os melhores *frameworks* multiplataforma, os itens da cor azul apresentam os desenvolvedores que usam o *framework*, já os itens na cor vermelha mostram os desenvolvedores que priorizam determinado *framework* na hora de desenvolver um aplicativo móvel.

De acordo com a pesquisa realizada pela *Vision Mobile* os *frameworks* Cordova, Xamarin e Unity são responsáveis por 70% da opção primária para a maioria dos desenvolvedores que usam desenvolvimento multiplataforma.

No decorrer do desenvolvimento deste trabalho, são utilizados os dois principais *frameworks* de desenvolvimento plataforma: Cordova e Xamarin. O *framework* Unity que ocupa a terceira colocação não foi analisado neste trabalho por ser específico para o desenvolvimento de jogos.

1.3.1.1 Apache Cordova

Apache Cordova, ou somente Cordova é um *framework* de desenvolvimento móvel de código aberto fundado em 2012 pela *Apache Software Foundation* (ASF). Com ele é possível utilizar tecnologias padrões da web, como HTML5, CSS3 e JavaScript para o desenvolvimento multiplataforma, evitando o uso da linguagem de desenvolvimento nativa das plataformas móveis existentes. Os aplicativos são desenvolvidos e direcionados para cada plataforma, e contam com ligações compatíveis com os padrões de API para acessar sensores, dados e status da rede de cada dispositivo (APACHE CORDOVA, 2015).

1.3.1.2 Xamarin

Xamarin é um *framework* de desenvolvimento móvel multiplataforma criado pela empresa *Xamarin Inc.* em 2011. Possibilita a criação de aplicações nativas iOS, Android, Mac e Windows Phone. Utilizando a linguagem de programação C# ele possibilita a criação de aplicações iOS nativas, também aplicativos Android e Windows Phone compartilhando seu código com múltiplas plataformas (XAMARIN, 2015).

1.3.1.3 Plug-ins

No contexto de desenvolvimento móvel multiplataforma, os plug-ins são trechos de código-fonte que possibilitam fazer o acesso as funções nativas que os dispositivos possuem. Um plug-in é parte de um código fonte que concede uma interface para acessar determinados componentes nativos de um dispositivo móvel.

Ao desenvolver um aplicativo híbrido, o desenvolvedor vê a necessidade de acessar os recursos do dispositivo tais como câmera, *bluetooth*, GPS, dentre outras

informações do dispositivo então ele irá fazer o uso do plug-in para isso (APACHE CORDOVA, 2015).

1.4 MÉTRICA DE SOFTWARE

Uma métrica de software é um método para determinar quantitativamente determinada medida para determinado atributo de processo, projeto, ou recurso. No contexto de desenvolvimento de software as métricas possibilitam aos gestores de projeto a contínua monitoração das medidas estimadas desde o início do projeto. Sua utilização foi proposta, principalmente para diminuir erros cometidos pelos gerentes ao estimar custos e tempo durante todo o desenvolvimento de um sistema (FONTOURA, 2006).

As métricas tem como objetivo aplicar uma abordagem de engenharia durante o desenvolvimento e a manutenção de um software. Ao realizar a medição das características de um software e o processo de seu desenvolvimento, é obtido um feedback que pode ser utilizado para compreender, controlar e melhorar constantemente os produtos e processos (MENDES, 2010).

O processo de qualidade de um software pode ser aplicado e entendido de diversas formas e utilizar diversas abordagens, mas todas se incluem as fases de avaliação e medição, o que demonstra a importância de estudos referentes a métricas de software. A norma ISO/IEC 9126 aborda esse assunto determinando um modelo de qualidade com seus componentes trazendo melhoria de processos para que, informações sejam coletadas e auxiliem na tomada de decisões para que o software se torne cada vez mais eficiente (SILVA, 2010).

A norma ISO/IEC 9126 descrita na próxima seção, possui um acervo de métricas que ajudam os softwares a se adequarem a um padrão de qualidade internacional.

1.5 NBR ISO/IEC 9126 – QUALIDADE DE SOFTWARE

Intitula-se ISO (*International Organization for Standardization*) a organização responsável internacionalmente para a padronização. Ela foi fundada em 1947 como uma organização não governamental e atualmente conta com mais de 100 organizações nacionais de padronização, representando mais de 120 países. Sua sede fica em Genebra, Suíça e tem como atividade principal a produção de padrões para especificações para várias áreas. O objetivo principal da ISO é desenvolver padrões mundiais que facilitem o intercâmbio de produtos e serviços, assim criando uma cooperação intelectual, econômica e científica de técnicas (GUERRA; COLOMBO, 2009).

A IEC (*International Electrotechnical Commission*), fundada em 1906 é uma organização mundial que divulga normas internacionais relacionadas com eletrônica, eletricidade e outras áreas afins. A ISO, com participação da IEC, desenvolveu um conjunto de normas que tratam exclusivamente de padronização para a qualidade de produtos de software. As duas normas indicam modelos de qualidade e rotinas para a avaliação da qualidade de determinado software, trazendo um grande auxílio para se obter um software mais seguro e confiável e também trazendo como diferencial a qualidade do seu produto ou serviço. Empresas tem a possibilidade de se firmar e ganhar mais confiança no mercado fornecendo produtos com maior nível de qualidade e melhorando seus processos e seu produto final.

1.5.1 ISO 9126-3 – Modelo de qualidade

A ISO/IEC 9126-3 descreve um modelo de qualidade de software. Este modelo é constituído por duas partes, qualidades externas, qualidades internas e qualidade em uso, permitindo que o software seja avaliado em várias situações como requisitos, desenvolvimento, apoio, manutenção, aquisição, auditoria de software e garantia de qualidade.

As qualidades externas e qualidades internas possuem seis características, que são por sua vez divididas em subcaracterísticas. Métricas de qualidade

internas fornecem aos usuários, avaliadores, testadores e desenvolvedores o benefício que eles são capazes de avaliar as questões de qualidade de produtos de software desde cedo, antes que o produto de software seja executado. Métricas de qualidade externas usam medidas derivadas do comportamento do sistema, por meio de testes, operações ou observações do software executável e fornecem aos usuários, avaliadores, testadores e desenvolvedores o benefício que eles são capazes de avaliar a qualidade do produto de software durante o teste ou operação (ISO 9126-3,2004).

A Figura 4 mostra o modelo de qualidade para qualidade interna e externa de software. Ela classifica a qualidade do software em seis características: funcionalidade, confiabilidade, usabilidade, eficiência, facilidade de manutenção e portabilidade.

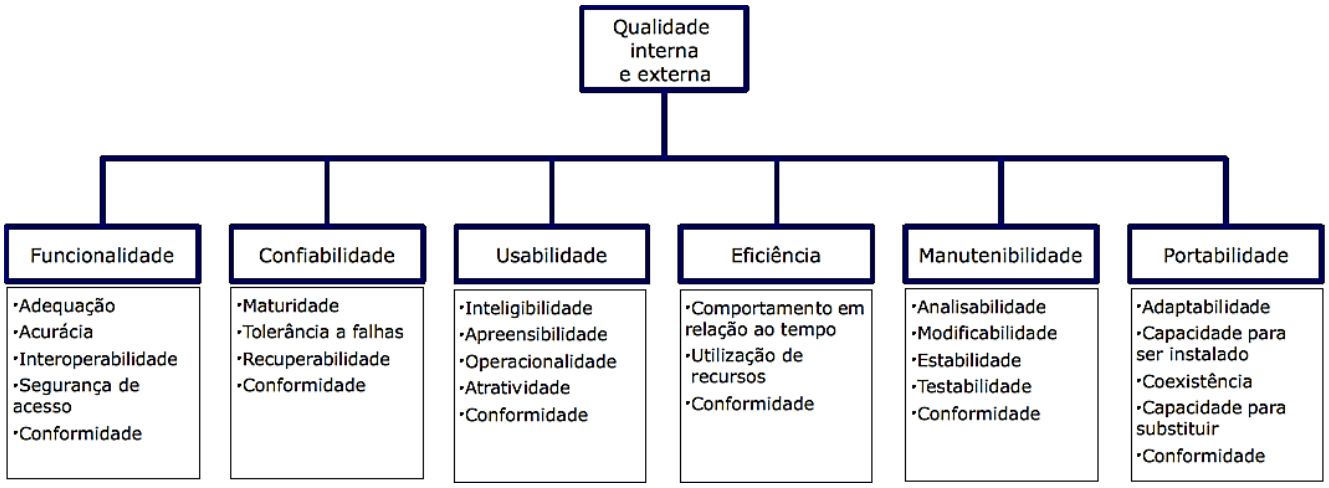

Figura 4 - Modelo de qualidade interna e externa.

Fonte: ISO/IEC 9126-1, 2004

As definições são dadas para cada característica de qualidade e suas subcaracterísticas. A ISO/IEC 9126-3 descreve que as características de qualidade do software são determinadas por um conjunto de seis características:

- **Funcionalidade:** A capacidade do produto de software fornecer funções que satisfaçam as necessidades explícitas e implícitas quando o software é usado em condições especificadas;

- **Confiabilidade:** A capacidade do produto de software para manter um nível específico de desempenho quando utilizado sob condições especificadas;

- **Usabilidade:** A capacidade do produto de software de ser compreendido, aprendido, usado e atraente para o usuário, quando usado sob condições especificadas;

- **Eficiência:** A capacidade do produto de software para proporcionar um desempenho adequado, em relação à quantidade de recursos usados, sob condições estabelecidas;

- **Manutenibilidade:** A capacidade do produto de software ser modificado. As modificações podem incluir correções, melhorias ou adaptações do software;

- **Portabilidade:** A capacidade do produto de software ser transferido de um ambiente para o outro.

2. METODOLOGIA

No primeiro estágio deste trabalho, foi feito uma pesquisa bibliográfica a fim de se obter o maior número de informações sobre pesquisas e trabalhos já existentes sobre o desenvolvimento de aplicativos móveis multiplataforma com o objetivo de encontrar metodologias que já foram ou não abordadas que possam ser incluídas no presente estudo.

No segundo estágio foi realizado um estudo sobre os principais *frameworks* de desenvolvimento *mobile* multiplataforma mais utilizados no ano de 2015. Os *frameworks* selecionados foram: Cordova (APACHE CORDOVA, 2015) e Xamarin (XAMARIN INC., 2015). Esses *frameworks* foram selecionados com base em uma pesquisa realizada pela Vision Mobile (2015), onde foram analisados os prós e contras dos *frameworks* de desenvolvimento multiplataforma e a atitude dos desenvolvedores em relação a eles. Os *frameworks* Cordova e Xamarin são os líderes de mercado e opção primária para a maioria dos desenvolvedores segundo a pesquisa realizada pela Vision Mobile (2015).

Posteriormente foi realizado um estudo sobre o método de desenvolvimento de cada *framework* selecionado, visando levantar características mais relevantes, tipos de desenvolvimento e o padrão de desenvolvimento final de cada aplicação resultante.

Dando seguimento ao estudo, foi desenvolvido um aplicativo simples em cada *framework*, que utilize as funcionalidades do dispositivo móvel: câmera, *bluetooth*, GPS, lanterna e mensagens de alerta.

Por fim, serão apresentadas as métricas de software selecionadas para a análise comparativa do presente estudo.

2.1 DESENVOLVIMENTO DO APLICATIVO

O aplicativo foi desenvolvido para os sistemas operacionais Android e Windows Phone. O sistema operacional iOS não foi utilizado devido à dificuldade em se obter uma licença de desenvolvedor.

Para cada *framework* abordado neste estudo, foram desenvolvidas duas versões do aplicativo, uma versão para o Android e outra para o Windows Phone, gerando quatro aplicações no total.

2.1.1 Escopo do aplicativo

O desenvolvimento do aplicativo proposto neste estudo visa explorar e analisar a utilização dos recursos dos dispositivos móveis: sensores, comunicação externa, câmera, mensagens de alerta, dentre outros recursos que são características gerais dos aplicativos móveis.

Para atingir o este objetivo, o aplicativo proposto possui uma interface simples com botões de acesso as funcionalidades desenvolvidas: câmera, mensagem de alerta, GPS e *bluetooth*. A utilização da câmera é feita através de um botão que abrirá uma interface de acesso à câmera onde a foto pode ser capturada e salva no dispositivo. A utilização do GPS, feita através de um segundo botão onde o aplicativo exibirá na parte inferior da tela a localização atual do dispositivo. Para utilização da funcionalidade de mensagens de alerta o aplicativo irá exibir uma notificação com uma mensagem de teste na parte superior do dispositivo. Por fim, a funcionalidade de acesso ao *bluetooth* permite ativar ou desativar a comunicação *bluetooth* do dispositivo.

2.1.2 Materiais utilizados

Para o desenvolvimento dos aplicativos Android e Windows Phone utilizando os *frameworks* abordados foi utilizado um notebook da marca Samsung com processador core i5 2450M 2.50 GHz com 4GB de memória RAM com sistema operacional Windows 8.1 de 64 bits.

Para realizar a implementação dos aplicativos propostos foi utilizada a ferramenta Visual Studio 2015 que possibilita criar aplicativos multiplataforma

utilizando sua extensão para aplicativos híbridos. Os *frameworks* Cordova e Xamarin possuem extensões que foram instaladas no Visual Studio para o desenvolvimento dos aplicativos.

Para realizar os testes das aplicações Android, foi utilizado um smartphone Motorola Moto G 2ª Geração, o qual possui um processador Qualcomm MSM8226 Snapdragon 400 Quad-core de 1.2 GHz com 1GB de memória RAM, sistema operacional Android Lollipop versão 5.0.2 e tela de 5 polegadas. Para realizar os testes das aplicações Windows Phone foi utilizado um smartphone Nokia Lumia 920, o qual possui um processador Qualcomm MSM8960 Snapdragon S4 Dual-core com 1.5 GHz com 1GB de memória RAM, com o sistema operacional Windows Phone versão 8.1 e tela de 5 polegadas.

2.2 MÉTRICAS DE SOFTWARE

A NBR ISO/IEC 9126 é a referência básica para se avaliar a qualidade de software. O modelo de qualidade da ISO/IEC 9126-3 define um conjunto de seis características, que são divididas em subcaracterísticas: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

Para o presente estudo, se tornou necessário especificar as subcaracterísticas mais relevantes para a comparação dos *frameworks*, onde algumas foram alteradas para se enquadrar na análise do objeto de estudo deste trabalho, definindo-se assim os critérios de avaliação utilizados na análise comparativa descritas na próxima seção.

2.2.1 MÉTRICAS DE FUNCIONALIDADE

Mede a capacidade do software de fazer o que foi proposto de forma apropriada. Verifica se o software fornece funções que satisfaçam as necessidades explícitas e implícitas para o qual foi proposto.

Tabela 3- Descrição das métricas de funcionalidade

Nome da métrica	Objetivo	Fórmula	Interpretação do resultado
Adequação funcional	O quão adequadas são as funções implementadas?	X = 1- A/B A = Número de funções nas quais são detectados problemas em avaliação. B = Número de funções verificadas.	0 <= X <= 1 Quanto mais próximo de 1, mais adequada.

Fonte: Tradução do próprio Autor. Fonte: ISO/IEC 9126-3, 2004.

Para obter os resultados de análise da métrica de adequação funcional foram analisados os plug-ins utilizados de cada plataforma, onde foram implementadas funções utilizando os recursos do dispositivo.

2.2.2 MÉTRICAS DE CONFIABILIDADE

Indica um conjunto de atributos para avaliar a capacidade de produtos de software se manter em um nível de desempenho aceitável e desejado no caso de acontecerem falhas.

Tabela 4 - Descrição das métricas de confiabilidade

Nome da métrica	Objetivo	Fórmula	Interpretação do resultado
Evitar operação incorreta	Quantas funções são implementadas com capacidade de evitar operações incorretas?	X = A / B A = Número de funções implementadas para evitar padrões de operações incorretas.	0 <= X Quanto mais próximo de 1 melhor é o resultado. (mais funções foram implementadas para corrigir falhas).

		B = Número de dados incorretos introduzidos.	
Eficácia de restauração	O quão eficaz é a capacidade de restauração?	X = A / B A = Número de requisitos de restauração alvo implementadas. B = Número de requisitos de restauração.	0 <= X <= 1 Onde X é maior, melhor eficácia
Remoção de falhas	Qual é a proporção de falhas corrigidas?	X = A A = Número de falhas corrigidas. Y = A / B A = Número de falhas ocorridas. B = Número de falhas detectadas em avaliação.	X = contagem A = contagem Y = contagem / contagem B = contagem

Fonte: Tradução do próprio Autor. Fonte: ISO/IEC 9126-3, 2004.

Para obtenção dos resultados da métrica de evitar operações incorretas foram forçados erros nas aplicações desenvolvidas pelos *frameworks* através de tipos de dados incorretos, parâmetros incorretos e sequência de dados incorreta. Para a análise de eficácia e restauração foi verificado no desenvolvimento dos aplicativos os erros de execução ocorridos em cada *framework* e também verificado se cada um deles consegue recuperar os dados que estavam sendo trabalhados. Para realizar a medição da métrica de remoção de falhas foram utilizadas as falhas detectadas durante o uso de cada *framework*.

2.2.3 MÉTRICAS DE EFICIÊNCIA

Mede a capacidade do produto de software proporcionar um desempenho adequado, em relação à quantidade de recursos utilizados sob condições estabelecidas.

Tabela 5 - Descrição das métricas de eficiência

Nome da métrica	Objetivo	Fórmula	Interpretação do resultado
Tempo de processamento	Qual é a unidade de tempo estimada para se realizar determinado número de tarefas?	X = Tempo estimado	X = contagem Quanto menor, melhor.
Tempo de retorno	Qual é o tempo estimado para completar um grupo de tarefas relacionadas?	X = Tempo em segundos (calculado ou simulado)	X = tempo Quanto menor, melhor.
Utilização de entrada/saída	Comparar os erros de entrada/saída e compará-los com o número de linhas de código responsável em chamadas do sistema.	X = A/B A = Número de mensagens de erro. B = Número de linhas de código relacionadas com as chamadas do sistema.	X = tamanho Quanto menor, melhor.
Utilização de memória	Qual é o tamanho da memória que o produto vai ocupar para completar uma tarefa específica?	X = tamanho em Megabytes.	X = tamanho Quando menor, melhor
Tamanho da aplicação gerada	Analisar as propriedades do arquivo final gerado.	X = tamanho em bytes.	X = tamanho Quando menor, melhor.
Consumo de memória	Após a aplicação ser inicializada, verifica-se o valor.	X = MBs Tamanho em Megabytes.	X = tamanho Quanto menor, melhor.

Fonte: Tradução do próprio Autor. Fonte: ISO/IEC 9126-3, 2004.

Os resultados da métrica de tempo de resposta foram obtidos calculando o tempo de resposta ao acessar as funcionalidades desenvolvidas de cada aplicativo resultante. Para aplicar a métrica de tempo de processamento foi executado todas as funcionalidades desenvolvidas nos aplicativos resultantes e analisado o tempo necessário para execução. Para obter os resultados da métrica de utilização de memória foi calculado o quanto de memória o aplicativo utilizou para realizar a tarefa de acesso câmera dos dispositivos. Para obter o tamanho da aplicação gerada foi analisado o arquivo final gerado por cada *framework* utilizando o gerenciador de arquivos do Windows. Para obter os resultados da métrica de consumo de memória foi realizado a verificação no gerenciador de aplicação de cada dispositivo. Nas métricas de tempo de processamento e consumo de memória citadas na Tabela 5 foram realizadas dez execuções e em seguida foi feito o cálculo da média dos valores a fim de obter um resultado mais preciso.

3. RESULTADOS

Neste capítulo são apresentados os resultados da análise comparativa obtidos após o desenvolvimento do aplicativo e da aplicação das métricas.

3.1 MÉTRICAS DE FUNCIONALIDADE

Nas próximas sessões são abordados os resultados e discussões sobre as métricas aplicadas.

3.1.1 Adequação funcional

A métrica de adequação funcional foi aplicada analisando os quatro plug-ins de acesso aos recursos do dispositivo de cada *framework*: *plug-ins* de acesso a câmera, GPS, *bluetooth* e mensagens de alerta. Ambos os *frameworks* apresentaram falhas durante a utilização de alguns de seus plug-ins. A forma de avaliação que a métrica apresenta é utilizando o número de funções nas quais foram encontrados problemas em avaliação e o número de funções verificadas. De acordo com a descrição métrica, quanto mais próximo do valor 1 melhor e mais satisfatório será o resultado obtido.

Tabela 6 - Dados sobre a métrica de adequação funcional

	Android	**Windows Phone**
Cordova	0,75	0,50
Xamarin	0,75	0,75

Fonte: Próprio autor.

Observando na Tabela 6, o *framework* Xamarin apresentou os melhores resultados mais próximos do valor 1 iguais a 0,75 nos sistemas operacionais Android e Windows Phone obtendo melhor resultado final em comparação com o *framework* Cordova que obteve desempenho inferior, obtendo o valor 0,50 no sistema operacional Windows Phone e 0,75 no sistema operacional Android.

O aplicativo gerado pelo *framework* Cordova apresentou um erro na execução do plug-in de acesso a câmera no sistema operacional Android e dois erros nos plug-ins de acesso a câmera e GPS no sistema operacional Windows Phone o que acabou comprometendo o seu resultado final que foi igual a 0,50. O aplicativo gerado pelo *framework* Xamarin apresentou erro de acesso ao GPS do dispositivo em ambos os sistemas operacionais.

3.2 MÉTRICAS DE CONFIABILIDADE

3.2.1 Evitar operação incorreta

Para obter os resultados da métrica de evitar operação incorreta foram forçados erros nos aplicativos através da utilização dos plug-ins existentes em cada *framework*. Os erros forçados foram durante a utilização dos plug-ins, onde foram realizadas passagens de parâmetros incorretos, tipos de variáveis incorretas e sequência de dados incorreta. A forma para realizar aplicação desta métrica é analisando o número de funções implementadas a fim de se evitar padrões de operações incorretas e também analisando o número de dados incorretos que são introduzidos. De acordo com a descrição da métrica, quanto mais próximo do valor 1 melhor e mais satisfatório. Para o caso a análise apresentada na Tabela 7, o resultado mais próximo do valor 1 implica melhor resultado e indica que em determinado *framework* avaliado mais funções foram implementadas para corrigir falhas.

Tabela 7 - Dados sobre a métrica de evitar operação incorreta

	Android	**Windows Phone**
Cordova	0,4	0,8
Xamarin	0,8	0,8

Fonte: Próprio autor.

O *framework* Cordova apresentou desempenho igual a 0,8 como observado na Tabela 7, em ambos os sistemas operacionais. Ambos os *frameworks* apresentaram resultados iguais, obtendo o valor 0,8 no Windows Phone porém o *framework* Cordova obteve o melhor desempenho no Android onde foi obtido valor superior em comparação com outro *framework*.

No resultado final, conclui-se que o *framework* Cordova utiliza mais funções para corrigir falhas no sistema operacional Android em comparação com o *framework* Xamarin.

3.2.2 Eficácia de restauração

Para obtenção dos resultados da métrica de eficácia de restauração foi analisado a capacidade de restauração de cada aplicativo resultante desenvolvido em cada *framework*. A capacidade de restauração foi analisada mediante os erros ocorridos e verificado se cada aplicativo resultante conseguia recuperar as informações sem comprometer o funcionamento do aplicativo. A forma de aplicação da métrica de eficácia de restauração é analisando o quão eficaz é a capacidade de restauração dos aplicativos. A obtenção dos resultados de acordo com a métrica é feita analisando o número de requisitos de restauração que foram implementadas e o número de requisitos de restauração existentes. Resultados mais próximos do valor 1 apresentam melhor eficácia.

Tabela 8 - Dados sobre a métrica de eficácia de restauração

	Android	**Windows Phone**
Cordova	0,8	1
Xamarin	1	1

Fonte: Próprio autor.

Ambos os *frameworks* apresentaram resultados iguais nos sistemas operacionais Android e Windows Phone pois através dos testes realizados conseguiram restaurar e voltar ao funcionamento normal do aplicativo sem comprometer o seu funcionamento. Conclui-se que ambos apresentaram boa satisfação e não comprometem o funcionamento da aplicação caso aconteça erros durante a sua execução.

3.2.3 Remoção de falhas

A métrica de remoção de falhas foi aplicada durante o desenvolvimento dos aplicativos nos dois *frameworks* em avaliação. Foi obtido o número de falhas detectadas durante a utilização e o desenvolvimento dos aplicativos propostos neste trabalho. O objetivo da métrica de remoção de falhas é analisar as falhas ocorridas e verificar qual foi a proporção de correção das mesmas. De acordo com a descrição e da métrica os resultados da avaliação são obtidos analisando o número de falhas detectadas em avaliação e o número de falhas corrigidas.

Tabela 9 - Dados obtidos da métrica de remoção de falhas

	Android	**Windows Phone**
Cordova	5	3
Xamarin	4	8

Fonte: Próprio autor

Como mostrado na tabela 9, o *framework* Xamarin apresentou maior proporção de erros apresentando 7 erros no resultado obtido. Por se tratar de utilizar linguagem orientada a objetos o Xamarin apresentou maiores erros na utilização de classes e heranças. O *framework* Cordova apresentou resultado igual a 4 durante a avaliação. Por utilizar o método de programação web: HTML5/CSS/Javascript o que torna sua programação mais simples, o *framework* Cordova se tornou mais eficaz durante a implementação dos aplicativos e apresentou menos erros durante a avaliação.

3.3 MÉTRICAS DE EFICIÊNCIA

A presente seção apresenta o resultado das métricas de eficiência.

3.3.1 Tempo de processamento

A métrica de tempo de processamento foi utilizada para calcular o tempo de processamento para acessar determinada quantidade de funcionalidades. Foram selecionadas para avaliação do tempo de processamento todas as funcionalidades desenvolvidas no aplicativo onde foi calculado o tempo necessário para realizar o acesso a todas elas. A métrica de tempo de processamento utiliza como objetivo a obtenção da unidade de tempo estimada para se realizar determinado número de tarefas. O resultado final obtido é o tempo de execução, onde quanto menor for, melhor.

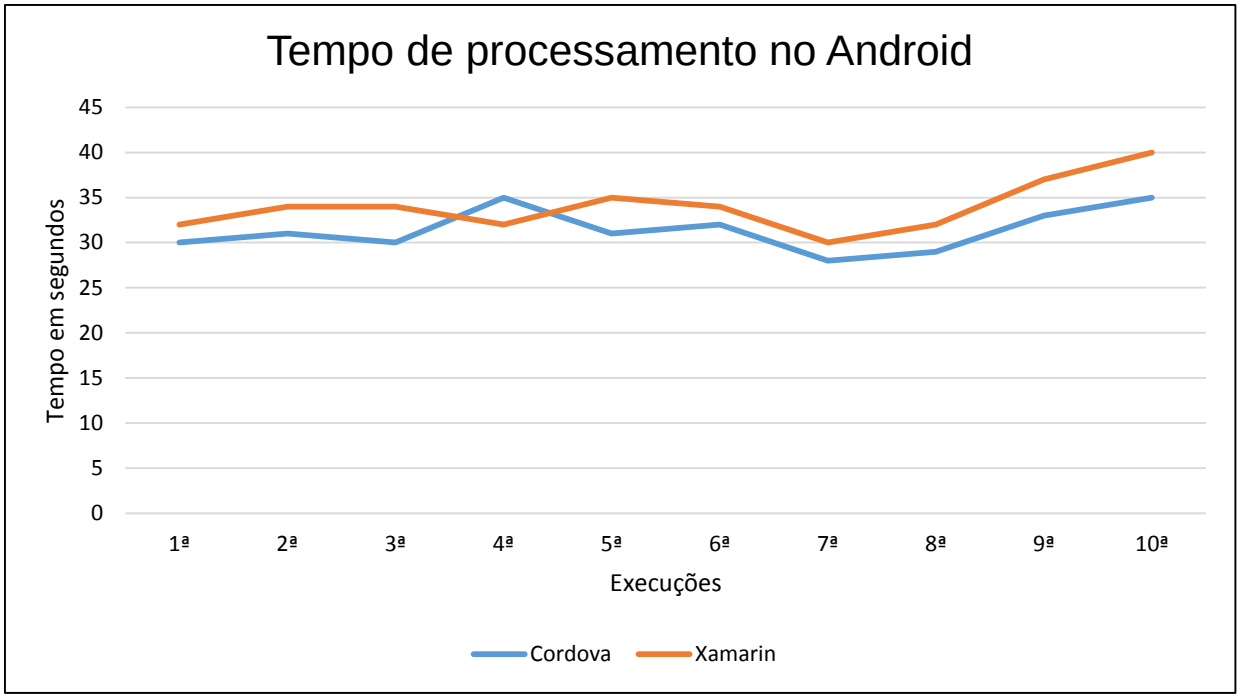

Gráfico 1 - Dados sobre o tempo de processamento no Android

Fonte: Próprio autor

Como apresentado no Gráfico 1, o Cordova apresentou melhor eficiência no tempo de processamento em todas as dez execuções exceto na quarta execução. Obteve melhor desempenho geral apresentando melhor tratamento no tempo de processamento no Android.

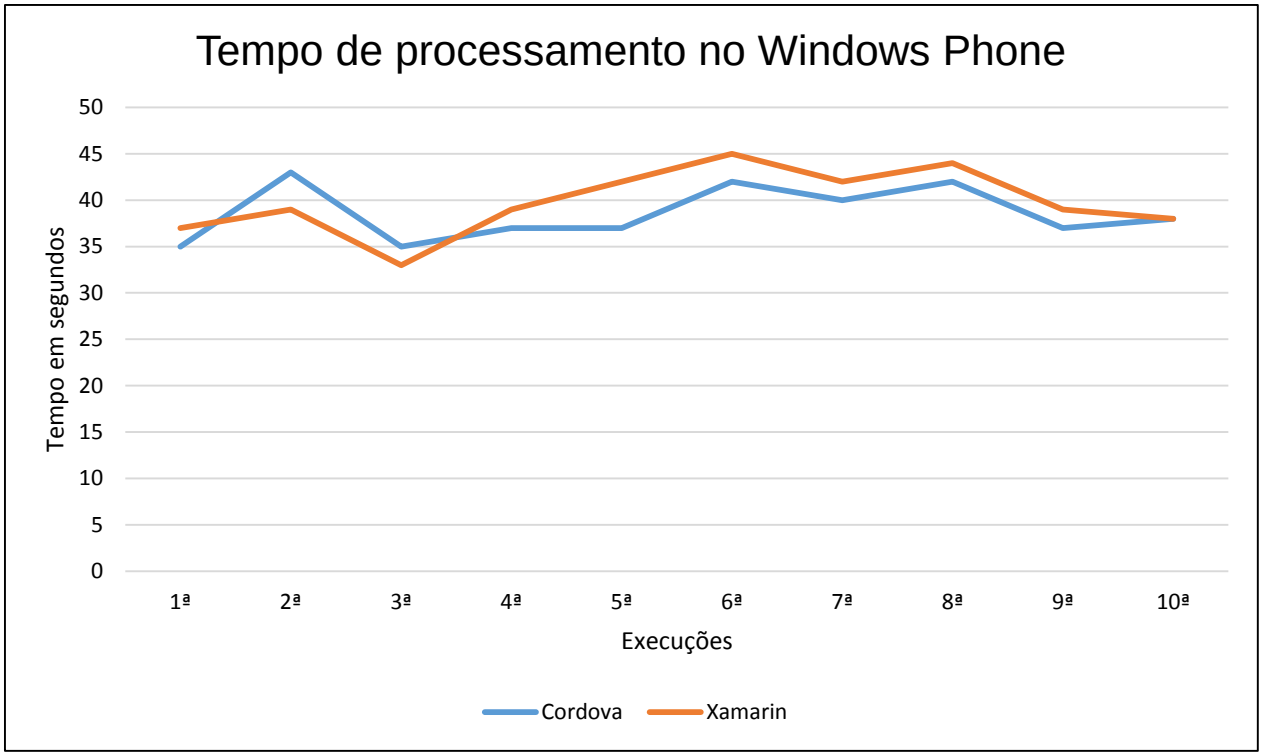

Gráfico 2 - Dados sobre o tempo de processamento no Windows Phone
Fonte: Próprio autor

No Windows Phone, como mostrado no Gráfico 2, o Cordova iniciou as duas primeiras execuções demandando mais tempo, porém no desempenho final analisando todas as dez execuções obteve o melhor resultado apresentando menos tempo de processamento para realizar a tarefa.

A análise final do tempo de processamento foi um pouco demorada visto a grande demanda de tempo que a funcionalidade de acesso ao GPS necessita como analisado anteriormente em conjunto com as demais funcionalidades do aplicativo na métrica de tempo de resposta. No final da execução o *framework* Cordova obteve melhor desempenho em ambos sistemas operacionais. O *framework* Xamarin obteve desempenho inferior, tendo maior demanda de tempo de execução em ambos sistemas operacionais.

3.3.2 Tempo de retorno

A métrica de tempo de retorno foi aplicada durante a verificação de tarefas relacionadas, ou seja, que possuem objetivos em comum. A tarefa selecionada foi a de ativação e desativação da conexão *bluetooth* do dispositivo desenvolvida no

aplicativo. A métrica de tempo de retorno tem como objetivo analisar o tempo estimado para completar um grupo de tarefas correlatas. O resultado obtido é através da medida de tempo em segundos, onde quanto menor o tempo, melhor será o resultado.

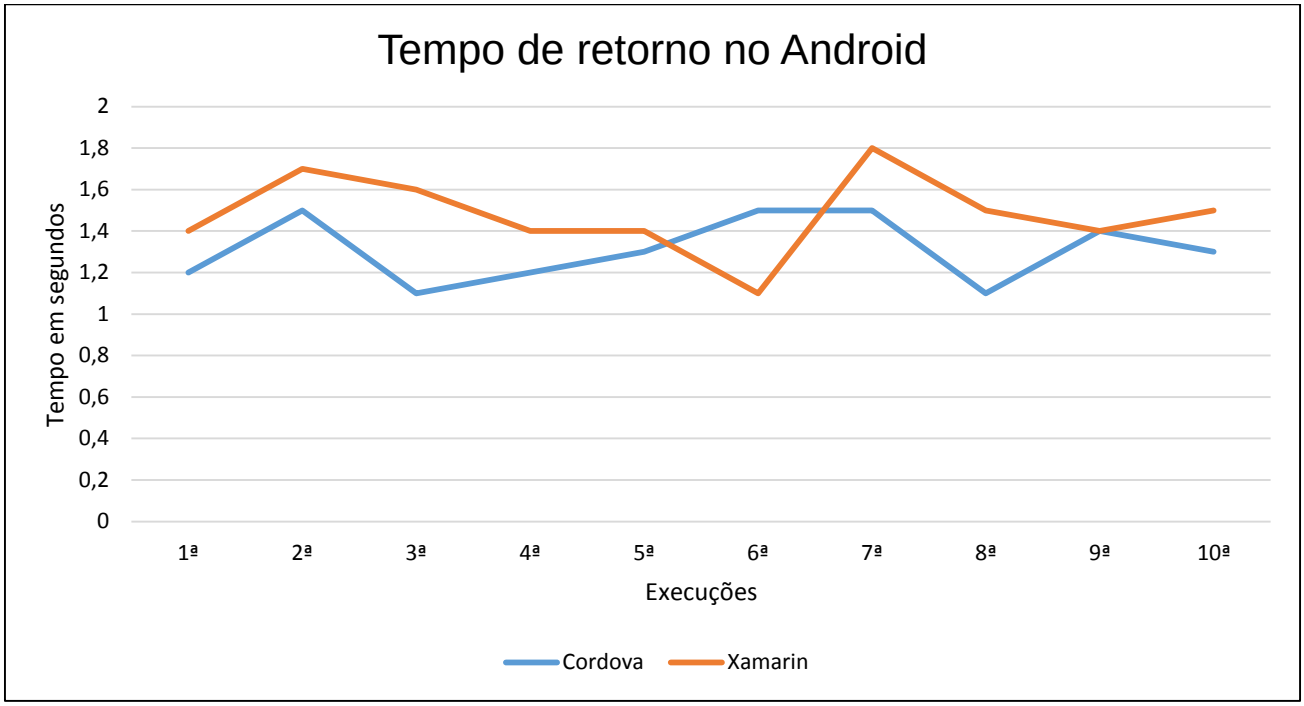

Gráfico 3 - Dados sobre a métrica de tempo de retorno no Android
Fonte: Próprio autor

No desempenho geral, como mostrado no Gráfico 3 após as dez execuções o Cordova apresentou melhor desempenho, realizando menos tempo para realizar a tarefa de tempo de retorno no Android.

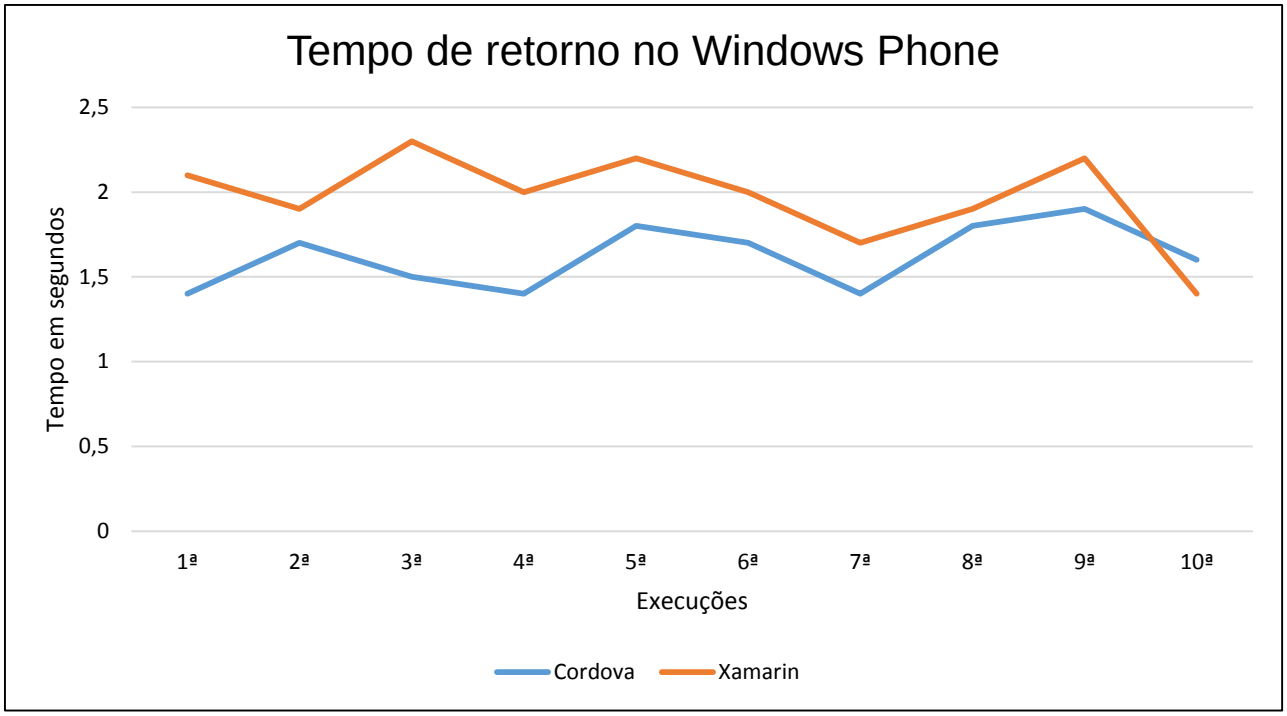

Gráfico 4 - Dados sobre a métrica de tempo de retorno no Windows Phone
Fonte: Próprio autor

Como mostrado no Gráfico 4, o Cordova obteve melhor desempenho para realizar a tarefa de tempo de retorno no Windows Phone.

O *framework* Cordova apresentou menor tempo de execução em ambos os sistemas operacionais.

3.3.3 Utilização de entrada/saída

A obtenção de resultados da métrica de utilização de entrada e saída foram realizadas através das funcionalidades desenvolvidas nos aplicativos. Foram verificados os erros de entrada e saída ocorridos durante a utilização das funcionalidades desenvolvidas no aplicativo. A métrica de utilização de entrada e saída tem como propósito realizar o levantamento dos erros de entrada e saída encontrados e compará-los com o número de linhas de código responsáveis pelas suas respectivas chamadas. Para obtenção dos resultados, segundo a métrica deverá ser analisado o número de mensagens de erro e o número de linhas de

código relacionados com as chamadas do sistema. De acordo com a métrica o resultado alcançado será em tamanho, onde quanto menor, melhor.

Tabela 10 - Dados sobre a métrica de utilização de entrada/saída

	Android	**Windows Phone**
Cordova	0,01	0,03
Xamarin	0,04	0,06

Fonte: Próprio autor

Esta métrica é relevante para realizar a análise mais robusta do comportamento de cada aplicativo mediante a possível entrada ou saída de vários tipos de dados. Analisando os dados como visto na Tabela 12 o *framework* Cordova obteve os menores resultados, ou seja, o melhor desempenho nesta métrica onde obteve tamanho igual a 0,01 no Android e 0,03 no Windows Phone comprovando que o *framework* obteve menor indício de mensagem de erros nos aplicativos. O *framework* Xamarin obteve resultados maiores em comparação com o Cordova tendo tamanho 0,04 no Android e 0,06 no Windows Phone apresentando maior indício de mensagens de erros em seus aplicativos resultantes.

3.3.4 Utilização de memória

A métrica de utilização de memória visa avaliar a quantidade de memória utilizada para se executar uma tarefa específica. Para realizar essa análise foi selecionada a tarefa de acesso a localização do dispositivo utilizando o GPS. Tal tarefa foi selecionada visto o seu resultado mais demorado em relação as outras tarefas como visto anteriormente na métrica de tempo de processamento.

De acordo como dito na métrica de utilização de memória seu propósito é analisar qual é o tamanho da memória que o produto de software de cada *framework* vai ocupar para completar uma tarefa específica. Para obtenção dos resultados foi apresentado o levantamento de memória consumida em *megabytes*.

Para realizar o levantamento do consumo de memória das tarefas nos sistemas operacionais Android e Windows Phone foi feita a análise do consumo de

memória no gerenciador de aplicativos de cada sistema operacional após a execução da tarefa de localização através do GPS.

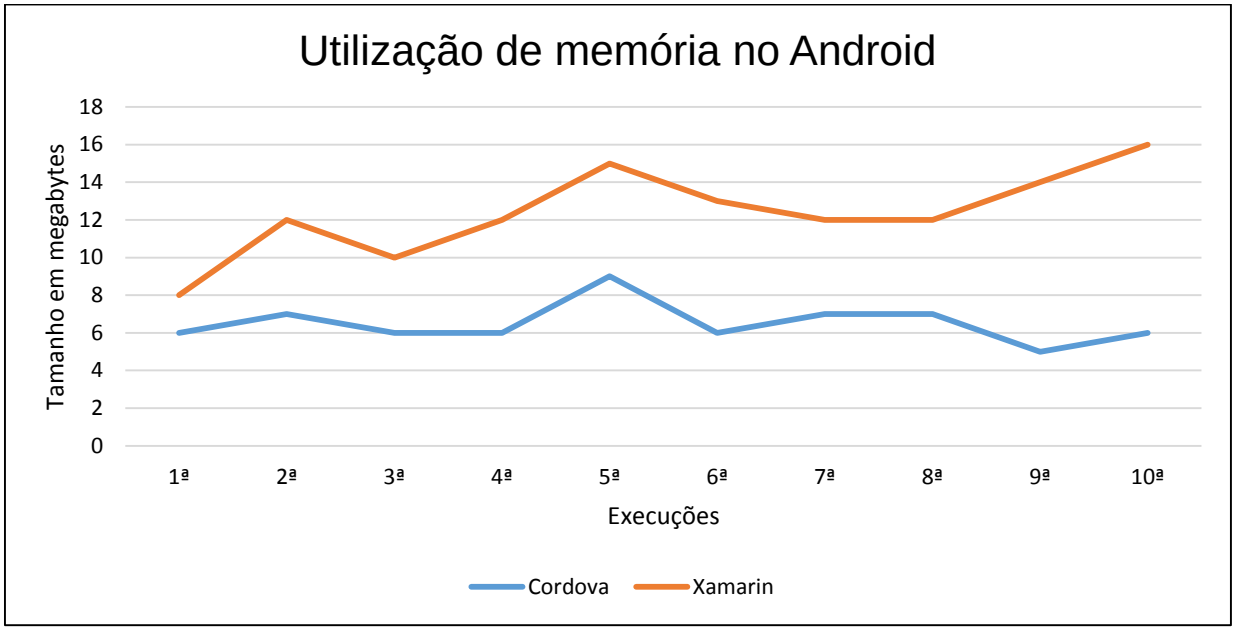

Gráfico 5 - Dados sobre a utilização de memória no Windows Phone
Fonte: Próprio autor

O Cordova apresentou melhor desempenho na utilização de memória como mostrado no Gráfico 5, obtendo menor consumo em todas as execuções realizadas no Android.

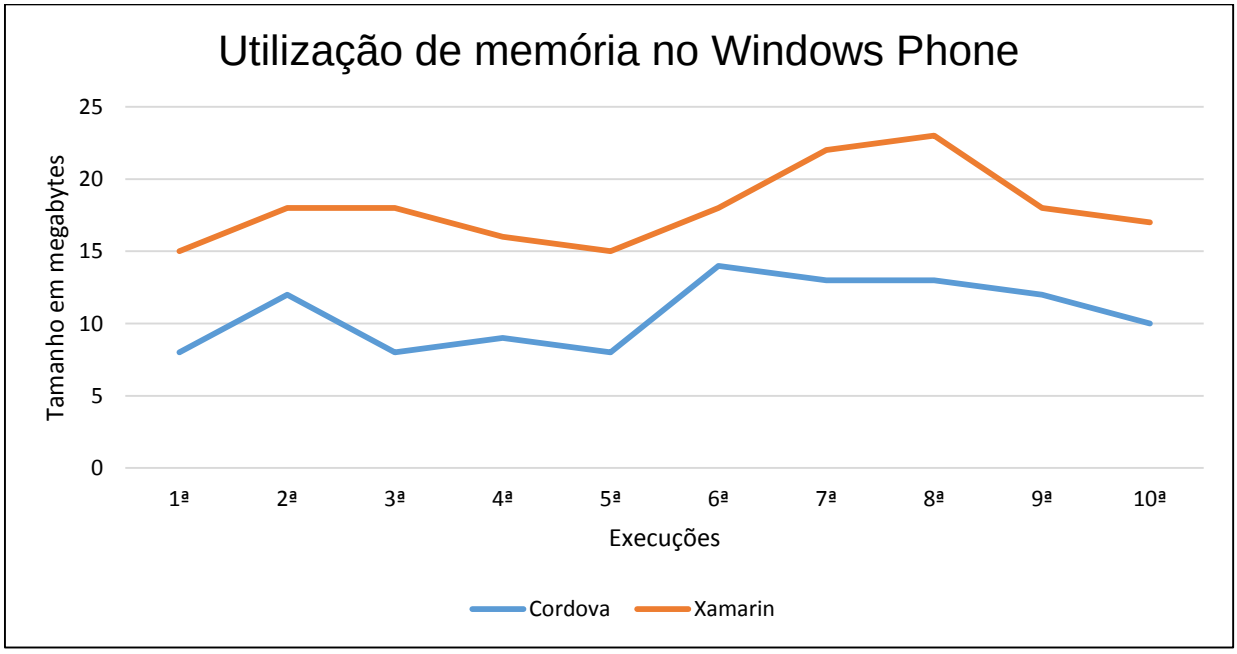

Gráfico 6 - Dados sobre a utilização de memória no Windows Phone
Fonte: Próprio autor

No Windows Phone o Cordova também apresentou melhor desempenho obtendo menor consumo de memória em todas as dez execuções como apresentado no Gráfico 6.

De acordo com os resultados obtidos, o *framework* Cordova obteve o menor consumo de memória em ambos os sistemas operacionais. O Xamarin obteve maior consumo de memória em comparação com o Cordova em ambos os sistemas operacionais.

3.3.5 Tamanho da aplicação gerada

Para obter os resultados da métrica de tamanho da aplicação gerada foi analisado o tamanho dos aplicativos gerados de cada *framework* após o seu desenvolvimento. A métrica de tamanho da aplicação gerada tem o intuito de analisar as propriedades do arquivo final gerado. O resultado é obtido em megabytes onde quanto menor o tamanho, melhor.

Tabela 11 - Dados do tamanho da aplicação gerada em Megabytes

	Android .apk (MB)	**Windows Phone .xap (MB)**
Cordova	1.861 MB	2.325 MB
Xamarin	5.756 MB	6.543 MB

Fonte: Próprio autor

A análise do tamanho da aplicação gerada foi feita através do gerenciador de arquivos do Windows analisando a propriedade do arquivo executável gerado por cada *framework* de cada sistema operacional. Esta análise foi relevante pois para efetuar o download do aplicativo executável gerado de cada *framework* quanto menor o seu tamanho menor será o consumo de dados da rede móvel e mais rápido será a transferência e possivelmente a instalação do mesmo. O *framework* Cordova apresentou o menor tamanho da aplicação em ambos sistemas operacionais.

A aplicação gerada pelo Cordova no Android e no Windows Phone apresentaram desempenhos satisfatórios, onde o tamanho do aplicativo gerado no Android foi de 1.861 megabytes e no Windows Phone 2,325 megabytes. O aplicativo gerado pelo Xamarin no Android foi de 5,756 megabytes apresentando quase o triplo do tamanho se comparado com o aplicativo do Cordova. No Windows Phone o Xamarin apresentou o tamanho igual a 6,543 megabytes, pouco mais do dobro em relação ao *framework* Cordova. Como dito anteriormente o *framework* Cordova apresentou o melhor desempenho nesta métrica, possibilitando facilidade e menor tempo para os usuários durante a aquisição, transferência e download das aplicações nas lojas.

3.3.6 Consumo de memória

A métrica de consumo de memória visa analisar o consumo de memória RAM gasto pelo aplicativo logo após ele ser inicializado. A forma de avaliação que a métrica utiliza é analisando o valor consumido de memória após a inicialização do aplicativo. O resultado é obtido em megabytes e quanto menor o tamanho, melhor.

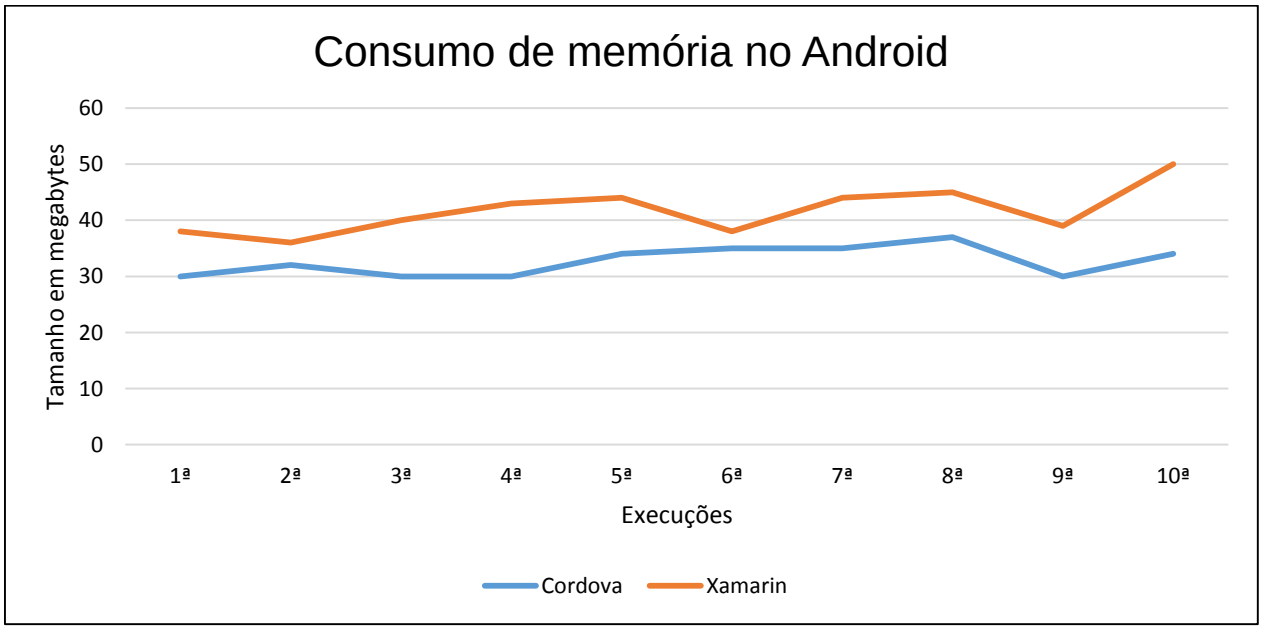

Gráfico 7 - Dados do consumo de memória no Android
Fonte: Próprio autor

Como mostrado no Gráfico 7, o Cordova se mostrou mais eficiente ao realizar baixo consumo de memória no Android em todas as dez execuções realizadas.

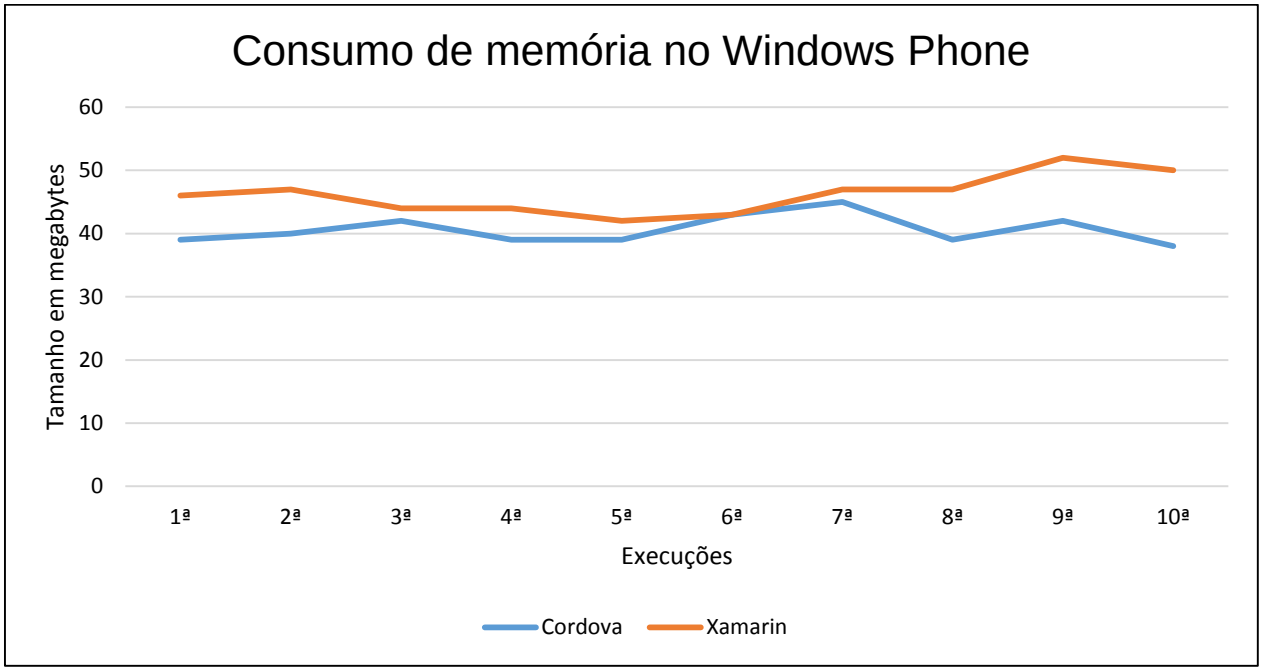

Gráfico 8 - Dados do consumo de memória no Windows Phone
Fonte: Próprio autor

No Windows Phone, o Cordova também mostrou melhor eficiência, garantindo baixo consumo de memória em todas as execuções realizadas, como apresentado no Gráfico 8.

O consumo de memória se tornou mais eficiente no aplicativo gerado pelo *framework* Cordova nos sistemas operacionais Android e Windows Phone. Esta análise é importante para usuários que fazem a utilização de vários aplicativos simultaneamente ou que possuem um dispositivo com menos memória RAM.

3.4 Discussões finais

Neste capítulo é apresentado o resultado final obtido após a análise dos aplicativos e dos *frameworks* de desenvolvimento *mobile* multiplataforma.

3.4.1 Resultados no Android

O **Erro! Autoreferência de indicador não válida.** apresenta os resultados da análise realizada nos dois aplicativos desenvolvidos utilizando os dois *frameworks* para o sistema operacional Windows Phone.

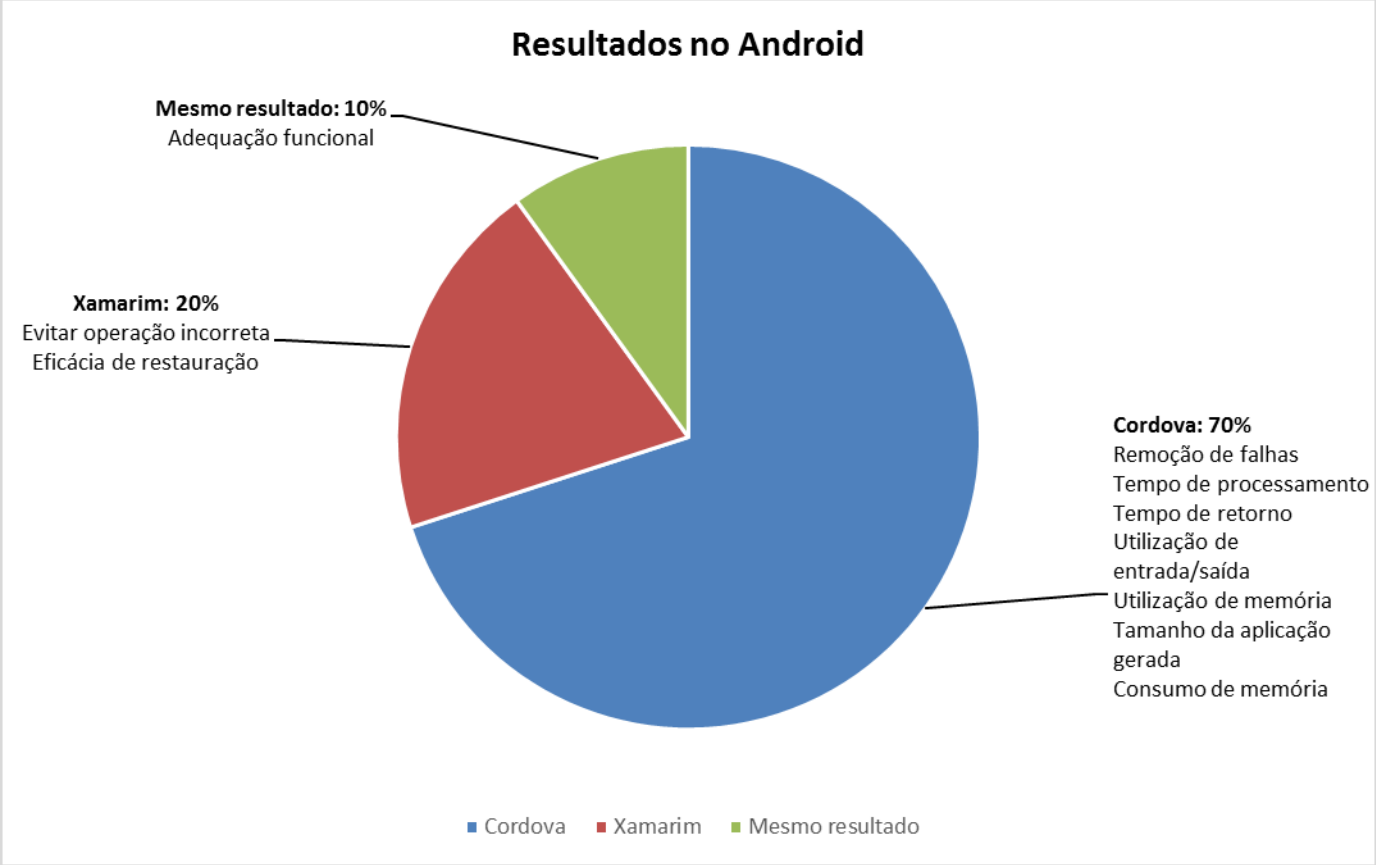

Gráfico 9 - Resultados da análise no sistema operacional Android
Fonte: Próprio autor

O *framework* Cordova na avaliação geral apresentou o melhor resultado (70%), obtendo melhor eficácia no sistema operacional Android em sete das dez métricas aplicadas neste trabalho. Ele se destacou e garantiu melhor resultado nas métricas de remoção de falhas, tempo de processamento, tempo de retorno, utilização de entrada/saída, utilização de memória, tamanho da aplicação gerada e consumo de memória. Durante as pesquisas realizadas neste trabalho o *framework* Cordova apresentou maior popularidade entre a comunidade de desenvolvedores, o que favorece muito a sua qualidade e também na qualidade do aplicativo desenvolvido pelo *framework*, pois ao se defrontar-se com um problema o desenvolvedor encontra maiores opções para solucioná-lo.

Os dois *frameworks* apresentaram desempenho igual (10%) na métrica de adequação funcional onde foram avaliados a utilização dos seus plug-ins de acesso aos recursos do dispositivo. O *framework* Xamarin obteve resultado superior nas métricas de evitar operação incorreta e eficácia de restauração onde apresentou 20% do resultado nas dez métricas aplicadas. O *framework* Xamarin não apresentou grande popularidade entre a comunidade de desenvolvedores, e apresentou escassez de exemplos em sua documentação oficial fazendo com que o desenvolvimento de seu aplicativo fosse complexo, com grande curva de aprendizado.

3.4.2 Resultados no Windows Phone

O Gráfico 10 apresenta os resultados da análise realizada nos dois aplicativos desenvolvidos utilizando os dois *frameworks* para o sistema operacional Windows Phone.

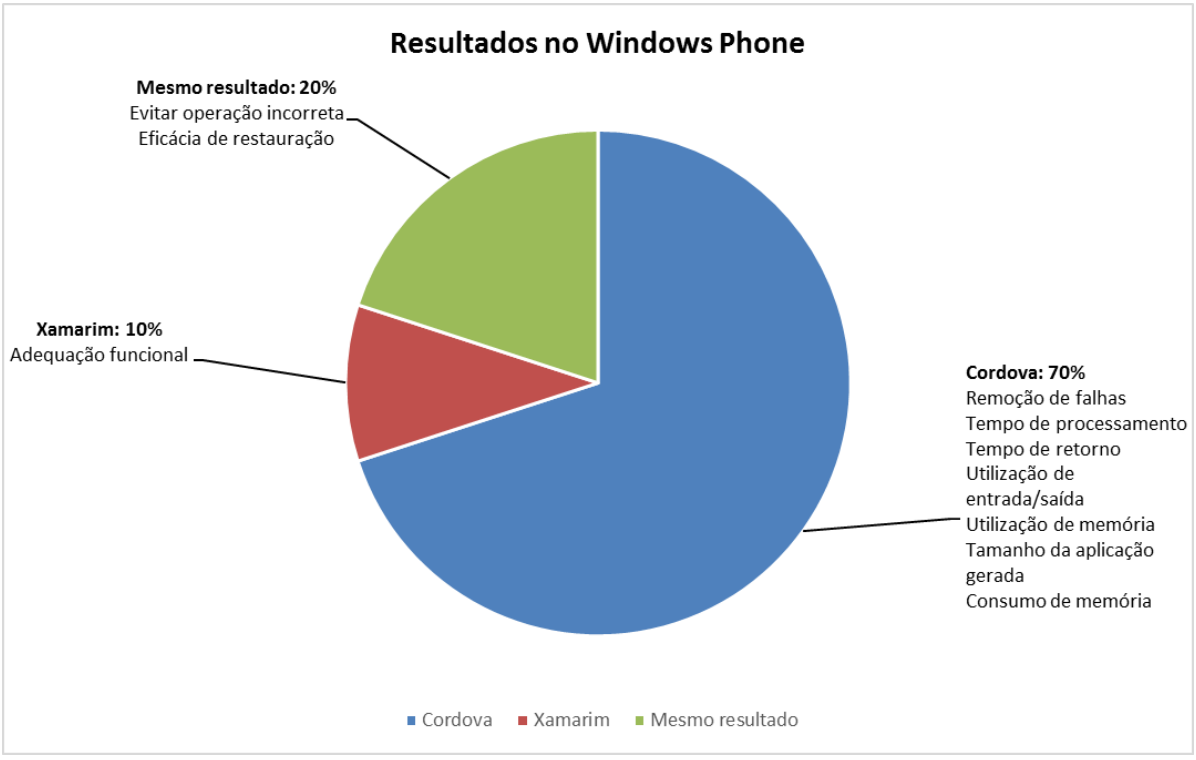

Gráfico 10 - Resultados da análise no sistema operacional Windows Phone
Fonte: Próprio autor

O *framework* Cordova também apresentou resultado superior no sistema operacional Windows Phone atingindo 70% nos resultados de avaliação e obtendo melhor desempenho nas métricas de remoção de falhas, tempo de processamento, tempo de retorno, utilização de entrada/saída, utilização de memória, tamanho da aplicação gerada e consumo de memória. O *framework* Xamarin obteve 10% dos resultados onde obteve melhor desempenho apenas na métrica de adequação funcional. Ambos os *frameworks* apresentaram o mesmo resultado (20%) nas métricas de evitar operação incorreta e de eficácia de restauração.

3.4.3 Discussão final

O Gráfico 11 apresenta o resultado final do desempenho dos dois *frameworks* avaliados nos sistemas operacionais Android e Windows Phone.

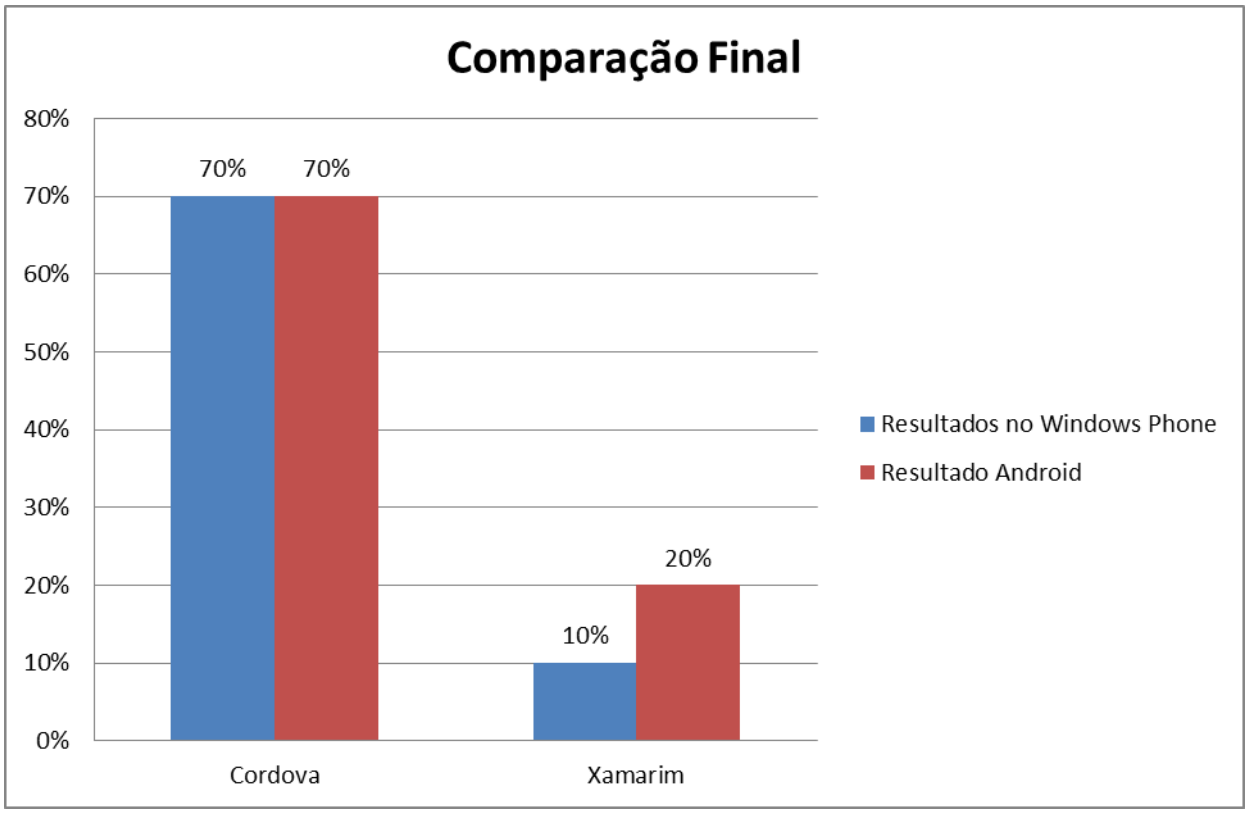

Gráfico 11 - Comparação final entre os frameworks Cordova e Xamarin
Fonte: Próprio autor

No Gráfico 11 podemos observar o resultado final obtido através da análise comparativa. É possível observar que o Cordova se tornou eficaz em 70% das dez métricas utilizadas na análise comparativa em ambos os sistemas operacionais, o que demonstra que o *framework* tem melhor qualificação para o desenvolvimento de aplicativos multiplataforma. O *framework* Xamarin apresentou desempenho inferior na análise comparativa, apresentando 30% de eficácia no sistema operacional Windows Phone e 10% no Android.

A partir do processo de desenvolvimento dos aplicativos propostos neste trabalho, a utilização do *framework* Xamarin se tornou complexa pela escassez de exemplos, por falta de elementos em sua documentação e também por utilizar linguagem de programação orientada a objetos fazendo com que o uso de várias classes e instâncias tornem o desenvolvimento dos aplicativos para os sistemas operacionais Android e Windows Phone fossem trabalhosos e demorados.

O *framework* Cordova proporcionou maior facilidade no desenvolvimento de seus aplicativos para ambos sistemas operacionais visto a utilização de padrões web HTML5/CSS/JavaScript fazendo que o processo de desenvolvimento fosse rápido e efetivo. O Cordova também apresentou maior facilidade para desenvolver a interface dos aplicativos visto a variedade de elementos encontrados em sua documentação. O Cordova também dispõe de uma numerosa comunidade de desenvolvedores, facilitando o trabalho do desenvolvedor e também fazendo com que os desenvolvedores que o utiliza possam contribuir constantemente através da correção de problemas e na criação de novas funcionalidades e plug-ins.

4. CONCLUSÃO

Com base nos resultados desse trabalho que teve como objetivo, realizar a análise comparativa entre os *frameworks* de desenvolvimento *mobile* multiplataforma Xamarin e Cordova com base nas normas de qualidade da ISO/IEC 9126. Os *frameworks* estudados foram Cordova e Xamarin através da leitura de estudos já existentes, artigos, documentação de cada *framework* e também pelo desenvolvimento de um aplicativo que utilize as funcionalidades de cada *framework*, cada um deles com uma interface simples que possibilita acesso aos recursos do dispositivo móvel como *bluetooth*, câmera, GPS e mensagens de alerta.

O aplicativo foi desenvolvido utilizando os dois *frameworks* gerando o mesmo aplicativo para duas plataformas: Android e Windows Phone gerando um total de quatro aplicativos, dois para cada *framework*. Com os aplicativos desenvolvidos foi então realizado a análise comparativa dos mesmos utilizando dez métricas de qualidade de software que foram selecionadas da NBR ISO/IEC 9126. As métricas foram selecionadas e organizadas para então serem aplicadas de acordo com sua descrição. Algumas métricas foram aplicadas analisando os próprios *frameworks* durante o desenvolvimento do aplicativo e outras foram analisando os aplicativos resultantes que cada *framework* gerou analisando a forma com que eles se comportam, acessam os recursos do dispositivo, tempo de processamento que levam para realizar determinada tarefa, consumo de memória, entre outros.

Com base nos estudos realizados e com o resultados alcançados após a realização da análise dos *framework*s sobre as características de cada *framework* e seus aplicativos desenvolvidos, o *framework* Cordova foi classificado como o mais efetivo com relação a curva de aprendizagem, ao conteúdo de qualidade na sua documentação, popularidade entre os desenvolvedores, custo de desenvolvimento e no desempenho final dos aplicativos desenvolvidos.

O *framework* Xamarin obteve alguns benefícios no aplicativo gerado para o sistema operacional Windows Phone, onde apresentou bom desempenho na utilização de recursos e na predição de operações incorretas e também na restauração de erros ocorridos em seu aplicativo. Porém mesmo assim no resultado geral não apresentou grande eficácia em outros casos analisados.

O *framework* Cordova obteve o melhor resultado no aplicativo gerado em ambos os sistemas operacionais e com base nas análises realizadas é considerado a melhor opção para desenvolvedores que procuram maior facilidade de acesso aos recursos, rapidez na hora de acessar os recursos do dispositivo, menor ocorrência de erros, capacidade de restauração, menor número de falhas, menor tempo para realizar tarefas melhor utilização de chamadas de funções do sistema, menor consumo de memória e menor tamanho do aplicativo final.

5. TRABALHOS FUTUROS

Uma possível continuação deste estudo poderia ser a realização de um estudo utilizando métricas de usabilidade também propostas pela ISO/IEC 9126 a fim de se analisar a interface e usabilidade dos aplicativos finais. Além disso também poderá ser proposto o desenvolvimento de um questionário destinado a desenvolvedores de aplicativos *mobile* a fim de identificar o perfil dos desenvolvedores, quais as suas experiências técnicas e suas prioridades, quais *frameworks* eles preferem quando vão desenvolver uma aplicação multiplataforma, quais os desafios encontrados durante a utilização de determinado *framework*.

A realização da aplicação das métricas no sistema operacional iOS ou em mais sistemas operacionais móveis, a fim de abranger os maiores e mais populares sistemas operacionais móveis.

Realizar a análise nos aplicativos finais desenvolvidos em cada *framework* em diferentes dispositivos de diferentes marcas a fim de abranger maiores dispositivos em diferentes configurações de hardware.

REFERÊNCIAS

9TO5MAC STAFF. **Jobs' original vision for the iPhone: No third-party native apps**. 9To5Mac, 2011.
Disponivel em: <http://9to5mac.com/2011/10/21/jobs-original-vision-for-the-iphone-no-third-partynative-apps>. Acesso em: 24 out. 2015.

ADOBE SYSTEMS INC. **PhoneGap**, 2015. Disponível em: <http://phonegap.com/>. Acesso: 16 Out. 2015.

ALVES, Wilson. **T-Mobile G1: Full Details of the HTC Dream Android Phone**. gizmodo, 2010. Disponivel em: <http://gizmodo.com/5053264/t+mobile-g1-full-details-of-the-htc-dream-android-phone>. Acesso em: 23 Set. 2015.

ALLEN, Sarah; GRAUPERA, Vidal; LUNDRIGAN, Lee. **Desenvolvimento profissional multiplataforma para smartphone**: iPhone, Android, WIndows Mobile e BlackBerry. 1 ed. Rio de Janeiro, RJ: ALTA BOOKS, 2012. 264 p.

AMBROS, Luisa. **Diferença entre aplicativos nativos, híbridos e *mobile* web apps**. Disponível em: <http://www.luisaambros.com/blog/diferenca-entre-aplicativos-nativos-hibridos-e-mobile-web-apps/>. Acesso em: 26 set. 2015.

ANDROID. **Google android**. Disponível em: <https://www.google.com/mobile/android/>. Acesso em: 05 out. 2015.

APACHE CORDOVA. **Overview**. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/>.Acesso em: 06 out. 2015.

APPCELERATOR. **Titanium SDK & Titanium Studio**. Appcelerator Docs, 2015. Disponivel em: <http://docs.appcelerator.com/titanium/latest/>. Acesso em: 31 set. 2015.

APPLE. **Iphone**. Disponível em: <http://www.apple.com/br/iphone/>. Acesso em: 17 set. 2015.

CÂMARA, M. **Como o iPhone mudou o rumo dos smartphones**. Gizmodo, 2012. Disponivel em: <http://www.gizmodo.com.br/artigos/noticia/2012/06/como-o-iphone-mudou-o-rumo-dossmartphones.html>. Acesso em: 20 out. 2015.

CHARL, **Mobile Application Development: Web vs. Native**. Communic Ations of the
Acm, Maio 2011. 49-53.

CORDOVA. **Apache cordova.** Disponível em: <https://cordova.apache.org/>. Acesso em: 24 set. 2015.

CYGNIS MEDIA. **Benefits (and disadvantages) of developing cross-platform *mobile* apps**. 2013. Disponível em: <http://www.cygnismedia.com/blog/developing-cross-platform- mobile-apps>. Acesso em: 20 out. 2015.

FONTOURA, Lisandra Manzoni ; BENETTI, Jean Carlo. **SCAM – Software para Coleta e Análise de Métricas**, Santiago. 2006.

GOOGLE. **Android Developers**, 2015. Disponivel em: <http://developer.android.com/tools/index.html>. Acesso em: 20 out. 2015.

GUERRA, Ana Cervigni. COLOMBO, Regina Maria Thience. **Tecnologia da informação: Qualidade de Produto de Software**. Brasília: Secretaria de Politica de Informática. 2009.

GUINTHER, Marcel. **Desenvolvimento *mobile* multiplataforma com Phonegap**. 2011. Disponível em: <http://www.mobiltec.com.br/blog/index.php/desenvolvimento-mobile- multiplataforma-com-phonegap>. Acesso em: 23 set. 2015.

HAWKES, Rob. **What the hell is webkit anyway? an all-purpose guide**. Disponível em: <http://memeburn.com/2013/02/what-the-hell-is-webkit-anyway-an-all-purpose-guide/>. Acesso em: 09 set. 2015.

HEIDENHEIMER, A. J.; HUGH HECLO, C. T. A. **Comparative Public Policy**. [S.l.]: St. Martin's Press, 1983.

IDC. **Smartphone os market share, 2015**. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 04 out. 2015.

JANSSEN, C. ***Mobile* Operating System (Mobile OS)**. Techopedia, 2010. Disponivel em: <http://www.techopedia.com/definition/3391/mobile-operating-system-mobile-os>. Acesso em: 24 out. 2015.

MENDES, Emília. **Introdução a métricas e medição de software**. ESELAW - Experimental Software Engineering Latin American Workshop. Goiânia, GO: 7ª ed. 2010.

MORIMOTO, Rodrigo. **Identificação de edificações baseada em imagens com dispositivos móveis**. Universidade de São Palo: [s.n.], 2009. 14 p.

MICROSOFT. **The smartphone reinvented around you.** Disponível em: <https://www.microsoft.com/pt-br/windows/phones>. Acesso em: 02 set. 2015.

NUNES, Flávio. **Desenvolvendo aplicativos móveis multiplataforma**. São Paulo, 2013. Disponível em: <http://imasters.com.br/desenvolvimento/desenvolvendo-aplicativos-moveis-multiplataforma>. Acesso em: 24 set. 2015.

PINHEIRO, José Maurício. **Prova de conceito (poc)**. Disponível em: <https://desmontacia.wordpress.com/2010/12/21/prova-de-conceito-poc-no-projeto-de-redes-de-computadores/>. Acesso em: 07 set. 2015.

RENE, Gesmar. ***Frameworks* de desenvolvimento multiplataforma** – Parte 1. 2012. Disponível em: <http://gesmarjunior.wordpress.com/2012/04/01/frameworks-desenvolvimento-multiplataforma>. Acesso em: 25 set. 2015.

REZENDE, Vandir F. **Análise comparativa entre Groovy e Java, aplicado no desenvolvimento web**. 2011. 76 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SILVA, Jomar**. Desenvolvimento de apps multiplataforma para dispositivos móveis com HTML5**. 2013. Disponível em: <http://www.slideshare.net/IntelSoftwareBR/desenvolvimento-de-apps-multiplataforma-para-dispositivos-mveis-com-html5>. Acesso em: 01 set. 2015.

SOUZA, Édipo da Silva. **Uma análise comparativa de ferramentas de desenvolvimento multiplataforma para dispositivos móveis**. 2014. 46 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas Digitais) – Universidade Federal do Ceará, Ceará.

SOUZA, Carlos E. de. **Desenvolvimento de aplicativo móvel multiplataforma integrado ao sistema de alerta de cheias da bacia do Itajaí**. 2012. 97 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SHANKLAND, S. **Google's Android parts ways with Java industry group**. CNET, 2007. Disponivel em: <http://www.cnet.com/news/googles-android-parts-ways-with-java-industry-group/>. Acesso em: 24 out. 2015.

SILVA, Luciano Alves Da. Apostila de android. **Apostila de android – programando passo a passo 3ª edição**, [S.L.], v. 3, n. 3, p. 7-14, jan. 2012.

VISION MOBILE. **Cross-platform tools 2015**. Disponível em: <http://www.visionmobile.com/product/cross-platform-tools-2015/>. Acesso em: 05 set. 2015.

WAZLAWICK, Raul S. **Engenharia de software**: conceitos e práticas. Rio de Janeiro: Elsevier, 2015.

WEINBERG, D. **Considerations for Developing Applications for Smartphones**. Opposing Views, 2011. Disponivel em: <http://science.opposingviews.com/considerations-developing-applicationssmartphones-12052.html>. Acesso em: 19 set. 2015.

XAMARIN, **Aplicativos xamarin**. Disponível em: <https://xamarin.com/platform>. Acesso em: 06 out. 2015.