DESENVOLDIMENTO MOBILE NATIVO OU HÍBRIDO; QUAL É A MELHOR FORMA DE DESENVOLVIMENTO?

Edjenal Ferreira Tavares Júnior [*]

RESUMO

Os telefones móveis, *smartphones*, deixaram de ser usados somente para ligações e hoje eles podem nos oferecer uma gama de recursos, como o acesso a serviços financeiros, entre outros. Tudo isso é possível devido a evolução tecnológica do hardware, permitindo o uso de sistemas operacionais mais avançados. Estimulado por este crescimento, o mercado de aplicativo para aparelhos celulares mostra-se em crescimento acelerado.

Este trabalho acadêmico visa pesquisar e analisar as principais formas de desenvolvimento mobile atuais, para que se possa proporcionar à elaboração de um quadro que auxilie na escolha da arquitetura, proporcionando ao gerente de projetos, engenheiro de software, entre outros, reduzir custos com a criação da aplicação mobile, garantir a qualidade, facilidade de criação e manutenção.

PALAVRAS-CHAVES: Desenvolvimento mobile, *smartphone*.

[*] Tecnólogo em sistemas para internet, desenvolvedor web, FANESE, edjenal.ferreira@gmail.com.

INTRODUÇÃO

É crescente o uso de dispositivos móveis (*smartphone*), tendo em vista que esses nos trazem a facilidade de transporte, com grande quantidade de recursos e os mais variados aplicativos, como os aplicativos bancários, compras, jogos, interação social, entre outros.

Segundo Grønli (2014, p. 635), cada empresa fabricante de dispositivos mobile, pode adotar seu padrão de S.O. (sistemas operacionais) diferente, como por exemplo: *Android* (*Google*), *iOS* (*Apple Inc*), *Windows Mobile* (*Microsoft Corp*) e cada sistema operacional possui sua *IDE* (ambiente integrado para desenvolvimento de software), que por sua vez adota diferentes linguagens de programação específica como, *Java* para *Android*, *Objective-C* ou *Swift* para *iOS* e *C++* ou *C#* para *Windows Mobile*.

O desenvolvimento de um aplicativo mobile exige conhecimentos específicos a respeito das tecnologias utilizadas pela plataforma na qual se deseja executá-lo.

Uma má escolha na arquitetura de desenvolvimento do projeto, tende fortemente a tornar mais custoso o desenvolvimento e a manutenção no projeto, gerando mais esforço e custo para a disponibilização de um mesmo aplicativo para mais de uma plataforma.

Moro e Terezinha (2014, p. 164) definem que considerando o cenário, o qual contém uma grande diferença de tecnologias adotadas, desenvolver um aplicativo para um dispositivo uma vez e poder reutilizá-lo em outras plataformas tende a ser um dos maiores desafios da computação móvel.

Diante desses desafios, eis que surge uma forma de desenvolvimento que pode ser usada para a criação de aplicativos móveis para várias plataformas, denominado *Cross-Platform Mobile development* (Multiplataforma de desenvolvimento móvel), permitindo assim, que o projeto possua compatibilidade com diversas plataformas.

São exemplos de *Cross-Platform Mobile development* o *PhoneGap*, *Corona*, *Intel® XDK*, etc. Estes *frameworks* são úteis porque reduzem custos e aumentam a velocidade com que os aplicativos são desenvolvidos. Além disso, essas ferramentas de desenvolvimento moveis são simples de usar,

tendo em vista que são baseadas em linguagens comuns para *scripting*: *CSS*, *HTML* e *JavaScript*.

Essas ferramentas oferecem muitos benefícios, mas ainda existem contras nesta forma de desenvolvimento. Um dos gargalos mais relevantes desta abordagem de desenvolvimento multiplataforma é o desempenho na execução de aplicativos que usam a *API* do *JavaScript* e além disso, existem limitações como: o acesso a recursos nativos do dispositivo, a dependência da comunidade que mantém a ferramenta de desenvolvimento e a experiência de uso limitada, nelas também existem diferenças de técnica de desenvolvimento, empacotamento, variação no desempenho geral do aplicativo, entre outras.

Não há uma forma única de desenvolvimento que atenda à todas as plataformas de forma perfeita, isso faz com que gerentes de projetos tenham que analisar qual padrão de desenvolvimento ele deverá adotar, pois cada uma tem o seu ponto positivo e negativo, suas vantagens e desvantagens, como por exemplo:

Escolher desenvolver aplicativos nativos para cada plataforma e demandar gente qualificada na equipe para cada plataforma, ou;

Desenvolver de forma híbrida, usando uma mesma linguagem, e enfrentar as limitações do *framework* escolhido.

PROCEDIMENTOS METODOLÓGICOS

Neste trabalho foi realizado um estudo dos principais conceitos que fundamentam a forma de desenvolvimento de aplicações mobile, tanto no desenvolvimento nativo quanto no desenvolvimento híbrido. Também foi verificado casos de uso no mercado nacional e internacional.

Após estas análises, tornou-se possível a elaboração de um quadro para facilitar a escolha da arquitetura adequada à necessidade do projeto e equipe, agregando: qualidade, facilidade de criação, manutenção e baixo custo.

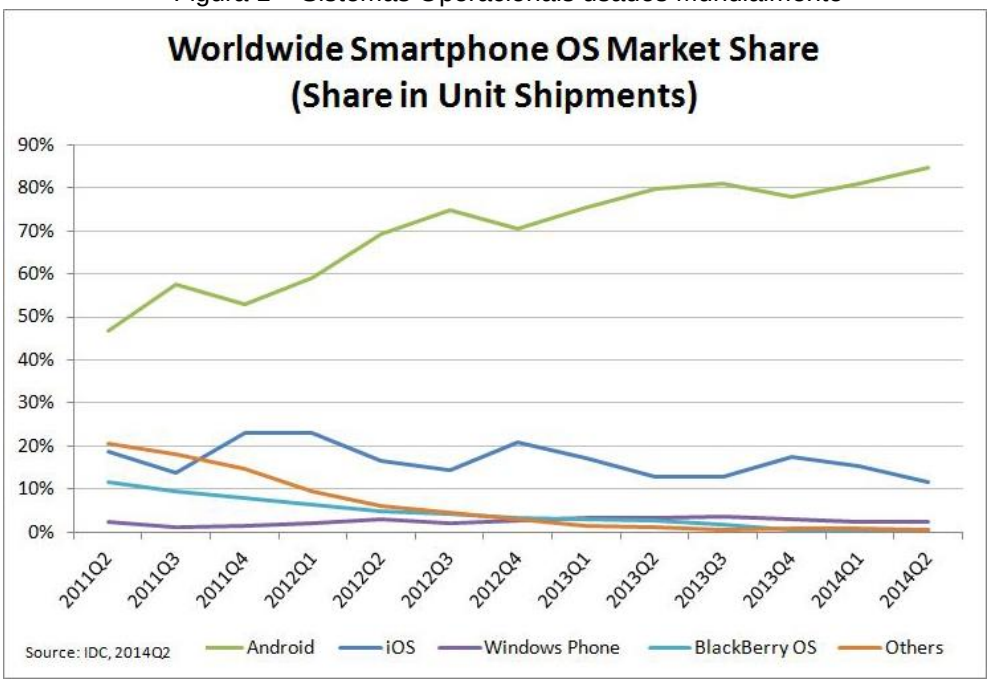
CONTEXTUALIZAÇÃO DO MERCADO MOBILE

O mercado de smartphones está bem aquecido e muitos consumidores já utilizam os aparelhos como principal forma de acessar a internet, tendo até mesmo substituído os computadores em diversos casos.

Segundo dados coletados em 2014, o Android é o sistema operacional mais utilizado em portáteis. Hoje existem mais de 1 bilhão de aparelhos ativos, significando que eles foram conectados à internet pelo menos uma vez nos últimos 30 dias. O iOS hoje possui 800 milhões de usuários ativos, considerando iPhones, iPads e iPods Touch. Na terceira posição do mercado surge o Windows Phone, que hoje possui cerca de 60 milhões de usuários ativos.

Para um melhor entendimento, é apresentado um infográfico feito pela empresa IDC:

Figura 1 – Sistemas Operacionais usados Mundialmente



Fonte: IDC - Smartphone OS Market Share 2015, 2014, 2013, and 2012.

Neste artigo são abordados os três maiores sistemas operacionais mobiles existentes no mercado, são eles: *Android* (*Google*), *iOS* (*Apple Inc*), *Windows Mobile* (*Microsoft Corp*).

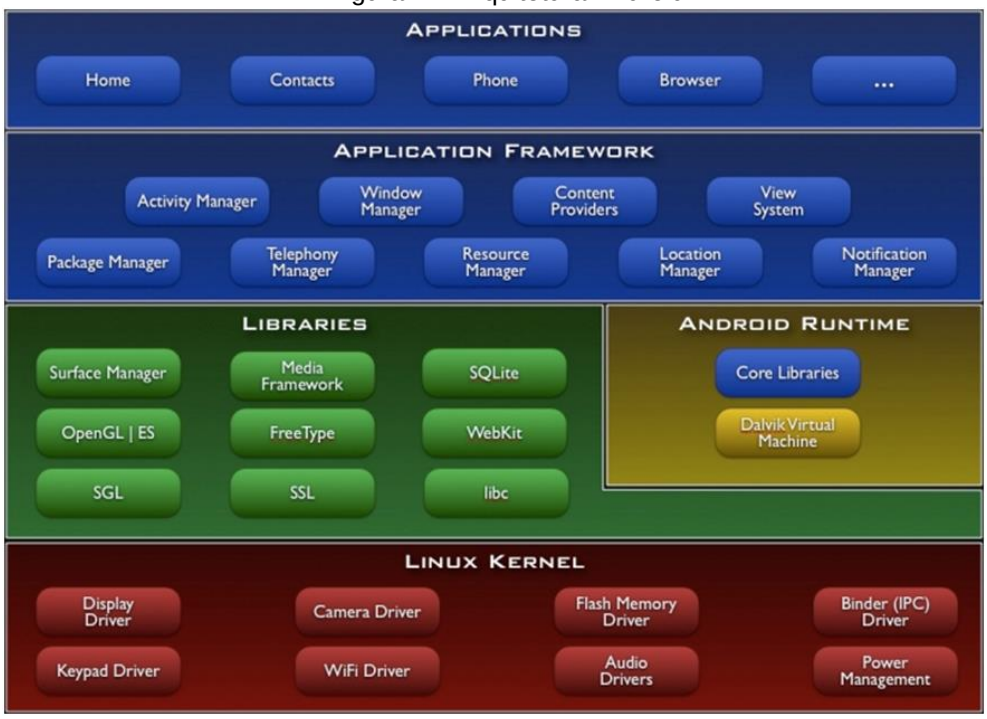
1 - Desenvolvimento Nativo

Para o desenvolvimento de uma aplicação de forma nativa para um dispositivo móvel é de vital importância que se conheça a arquitetura do sistema operacional e a forma como a sua aplicação é gerenciada (comumente conhecida como: ciclo de vida). A linguagem de programação também é um ponto bastante importante, pois pode demandar um tempo grande de aprendizado se a equipe não possuir conhecimento específico sobre a linguagem.

1.1- Android

O Google usou a versão 2.6 do Linux para construir o kernel do Android, o que inclui os programas de gerenciamento de memória, as configurações de segurança, o software de gerenciamento de energia e vários drivers de hardware. A figura abaixo detalha a arquitetura do Android:

Figura 2 – Arquitetura Android



Fonte: Android Developers.

A aplicação desenvolvida será executada na camada “Applications” e tem acesso somente a um restrito conjunto de recursos e bibliotecas, além de restringir também os arquivos e diretórios que a aplicação pode utilizar.

Para o desenvolvimento é necessário ter conhecimentos da linguagem de programação Java. A IDE (do inglês, Integrated Development Environment) recomendada pelo Google para desenvolver é o Eclipse. Os programas desenvolvidos para Android são compilados em byte codes e são executados em sua máquina virtual chamada Dalvik.

O sistema operacional Android usa máquinas virtuais para rodar cada aplicação como seu próprio processo. Isso é importante por algumas razões. Primeiro, nenhuma aplicação é dependente de outra. Segundo, se uma aplicação para, ela não afeta quaisquer outras aplicações rodando no dispositivo. Terceiro, isso simplifica o gerenciamento de memória.

1.2 – iOS

No iOS, a arquitetura do sistema e as tecnologias são semelhantes as encontradas no Mac OS X, tendo em vista que o kernel é baseado em uma variante do mesmo kernel base que é encontrado no Mac OS X. No topo deste kernel estão as camadas de serviços que são utilizadas para implementar aplicações na plataforma.

A linguagem de programação nativa da plataforma iOS são o Objective C e o Swift, havendo possibilidade de utilização da linguagem C e C++. O Objective C é uma linguagem que teve origem nos anos 80, mas não era tão presente no mercado, ganhando força junto ao crescimento da Apple. Arquitetura iOS:

Figura 3: Arquitetura iOS

Fonte: iOS Developer Library.

Recentemente a Apple desenvolveu o Swift, visando que o código fonte fique mais simples e conciso. Além de facilitar a vida dos programadores, que poderão desenvolver aplicativos com menos linhas de

código, a Swift possui uma proposta de melhorar também a vida dos instrutores, pois é possível partir diretamente do ensino dos recursos da linguagem. O aprendizado de Objetive-C, ao contrário, exige a explicação de muitas regras (e suas exceções) e a abordagem de temas técnicos mais profundos, dificultando o entendimento de quem não é da área de informática. Mas por ser uma linguagem nova no mercado, não existe tanto conteúdo em fóruns e sites comparado com C# e Java e a principal fonte de pesquisa da linguagem é a própria Apple. Ela disponibiliza conteúdo completo, suficiente para o bom desenvolvimento de projetos nessa linguagem.

Para que se possa desenvolver um aplicativo para o iOS é necessário possuir um Mac OS e ter instalado a interface de desenvolvimento da Apple que é o Xcode.

1.3 – Windows Phone

Anteriormente, o sistema operacional móvel criado pela Microsoft foi chamado de Windows Mobile. Após as alterações introduzidas pela Apple (iOS) e Google (Android) em 2007, a Microsoft decidiu tomar um novo rumo e criaram Windows Phone. Semelhante a outras alternativas, tais como Android, o Windows Phone é um sistema operacional para smartphones. É normalmente usado em dispositivos de tela de toque, e oferece funcionalidades como redes, sensores e integração câmera.

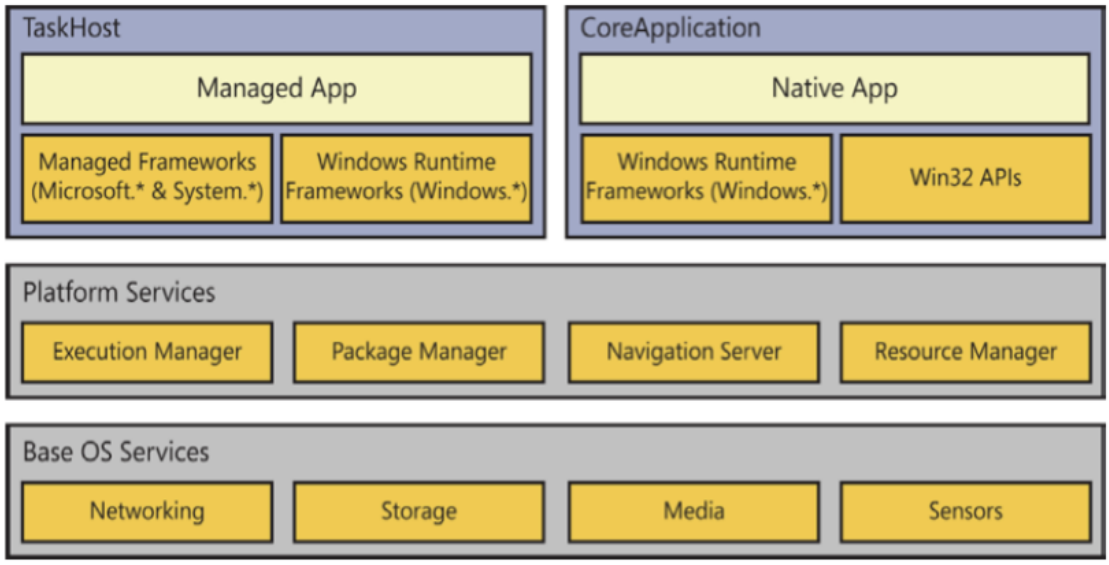
Programas para Windows Phone são escritos em código gerenciado .NET. O código gerenciado é código escrito em idiomas que estão disponíveis para utilização com o Microsoft .NET Framework, como por exemplo: C#. Um dos benefícios é que muito dos erros propensos e muitas vezes das tarefas complexas, como a verificação de tipo de segurança, gerenciamento de memória e destruição de objetos desnecessários, são tomadas de cuidados pelo .NET.

Windows Phone suporta duas plataformas de programação populares, Silverlight e XNA. Silverlight é uma evolução do Windows Presentation Foundation (WPF). Ele fornece aos desenvolvedores a capacidade de criar interfaces de usuário sofisticados. A segunda plataforma, XNA, é plataforma

de jogos da Microsoft. Ele suporta tanto gráficos em 2D quanto em 3D. Desenvolvimento para Windows Phone é feito em Visual Studio. Há uma gama de edições do Visual Studio, que vão desde o Free Visual Studio Express para a edição Ultimate. Embora a edição Express, seja o suficiente para começar, as limitações rapidamente ficar no caminho da produtividade. Por exemplo, uma das principais limitações é que não há suporte para plugins.

Há duas linguagens que podem ser usadas para escrever programas para Windows Phone, 1) Visual Basic .NET e 2) C #. Programas criados para Windows Phone são empacotados em Arquivos XAP, que é o pacote de aplicativos Silverlight. Arquitetura Windows:

Figura 4: Arquitetura Windows Phone



Fonte: Recursos e ferramentas para profissionais de TI | TechNet.

De acordo com a Gartner (2013), a Microsoft ocupa atualmente o 3º lugar no que diz respeito à quota de mercado (segundo trimestre de 2013). Pela primeira vez, a Microsoft tem uma maior quota de mercado em comparação com Blackberry. Mesmo com o recente aumento da popularidade, a plataforma Windows Phone ainda é relativamente pequeno, com uma quota de mercado de 3,3%.

2 – Desenvolvimento Híbrido

Na grande maioria dos casos, aplicações híbridas trazem a resposta para os desenvolvedores que desejam desenvolver para muitas plataformas, sem grande perda de desempenho. Essa abordagem permite ao desenvolvedor ter acesso e controle sobre as funcionalidades nativas do dispositivo. Isso faz com que, a experiência do usuário que está utilizando a aplicação em um iOS seja a mesma que em um Windows Phone, por exemplo, o que é muito importante em grande parte das aplicações, garantido a portabilidade de aplicações.

Essa abordagem híbrida vem ganhando muito destaque nos últimos tempos, pois permite que o desenvolvedor aprenda apenas o básico sobre cada uma das plataformas, uma vez que a parte principal da aplicação será desenvolvida utilizando tecnologias web.

O conceito clássico de híbrido refere-se ao ato de misturar. Na mistura de dois ou mais elementos diferentes o resultado é a oposição da ordem natural das coisas. Os aplicativos híbridos surgiram a partir desse conceito, entre a junção do nativo e da web, Hartmann et al. (2011)

A utilização das linguagens de programação web, como HTML5 e Java Script, e o empacotamento no formato nativo resulta em um aplicativo que vai funcionar em plataformas diferentes. Hoje em dia uma grande quantidade de soluções mobile gira em torno de aplicativos híbridos, pois o conceito de escrever uma vez e rodar em qualquer plataforma torna esse modelo bastante utilizado no desenvolvimento.

Aplicativos híbridos vão funcionar da mesma maneira como os aplicativos nativos, contudo:

- São baseados em HTML5, CSS3 e JavaScript, sendo essas as principais tecnologias;
- Exige um menor custo no desenvolvimento comparado com os nativos, isso por manter um código fonte apenas, não precisando desenvolver um código para cada plataforma;
- Para aplicar atualizações periódicas nos aplicativos, há uma enorme vantagem sobre as nativas, aplicando a atualização uma única vez.

Outro ponto fundamental é que existe uma enorme gama de frameworks que seguem a filosofia híbrida: jQuery Mobile com PhoneGap, Sencha Touch, Titanium, ZeptoJS, etc. Esses frameworks oferecem uma suíte de componentes prontos os quais agilizam o desenvolvimento do aplicativo.

O HTML5 chegou com intuito de suprir as necessidades do desenvolvimento web moderno. Os aplicativos mobile são os principais protagonistas que utilizam essa linguagem. Ele que permite que as aplicações hibridas se portarem semelhantes as aplicações nativas.

O principal recurso para esse fim é o armazenamento de variáveis e de bases de dados embarcados no dispositivo através do Local Storage e do WebSQL:

- Local Storage: o armazenamento é temporário, formado por par (chave / valor), permanecendo até que a sessão esteja ativa;
- WebSQL: a forma de armazenamento segue as especificações web de banco de dados SQL, o qual oferece toda estrutura já conhecida pela maioria dos desenvolvedores como tabelas, chaves primarias, e chaves estrangeiras. O Web SQL permite que o aplicativo mantenha os dados armazenados dentro do dispositivo sem necessidade de acesso à internet.

O JavaScript é uma linguagem de scripts (trechos de códigos) para web, totalmente dependente de um browser. É integrada nas páginas HTML do lado cliente sendo utilizada principalmente para manipular informações e objetos de uma página HTML.

Ela é carregada junto com a página web, sendo uma linguagem interpretada que não tem a necessidade de nenhum compilador e também é de fraca tipagem (a declaração de variáveis não necessita informar um tipo como String, int, double, etc.).

Nos primórdios de sua existência foi uma linguagem duramente criticada pela falta de segurança, ausência de logs de erros e por sua baixa produtividade no desenvolvimento. Mas hoje o JavaScript tornou-se uma linguagem imprescindível e de extrema utilidade no desenvolvimento web.

Ela pode ser considerada a principal linguagem para o desenvolvimento de apps mobile híbridos.

Não existe outra maneira de tornar seu aplicativo móvel híbrido sem pensar em JavaScript, sendo esta a única tecnologia que todas as plataformas nativas suportam. Além disso, existe hoje no mercado inúmeros frameworks que facilitam o desenvolvimento em JavaScript como jQuery, Angular, entre outras.

Um dos frameworks mais utilizados no mercado é o PhoneGap, ele permite construir aplicativos para dispositivos móveis usando JavaScript, HTML5 e CSS3. Ele torna possível realizar a transferência do código para a sua plataforma nativa (Android, iOS e Windows Phone), ou seja, o resultado de seu uso é um aplicativo mobile híbrido.

O núcleo do PhoneGap é composto por HTML5 e CSS para renderização e JavaScript para a lógica de negócio e programação. O HTML5 provê todo acesso aos hardwares nativos do dispositivo usando uma interface de função externa (FFI - Foreign Function Interface), sendo este um mecanismo que permite chamar funções e serviços escritos em outra linguagem de programação.

O PhoneGap também pode ser estendido com plugins nativos, o que permite ao desenvolvedor acessar as funções nativas dos dispositivos. Um plug-in são trechos de código que fornecem uma interface para componentes nativos do dispositivo móvel. Ao projetar um aplicativo híbrido que tem a necessidade de acessar recursos como câmera, coordenadas geográficas, tipo de conexão com a internet, informações sobre o dispositivo móvel (modelo, fabricante, uuid, etc.), base de dados (storage), deverá ser adicionado um plugin Cordova no projeto para isso. Hoje o Cordova Phonegap conta com aproximadamente 642 plugins disponíveis para uso que podem ser encontrados na documentação oficial do PhoneGap.

RESULTADOS

	Nativo	Híbrido
Prazo de Entrega	O desenvolvimento torna-se mais trabalhoso e demorado, pela necessidade de criar um aplicativo para cada plataforma do mercado.	O desenvolvimento fica mais ágil e barato, utilizando o Phonegap, por exemplo. O aplicativo iria abranger todas as plataformas do mercado, sem a necessidade de um retrabalho alto.
Verba disponível	O desenvolvimento do aplicativo nativo é praticamente multiplicado pela quantidade de plataformas que se deseja ter o software rodando, pois apesar do conhecimento de negócio ser aproveitado entre os aplicativos, o código desenvolvimento não possui nenhum tipo de reutilização. Além disto, será necessária mão de obra para desenvolver em plataformas distintas algo que pode encarecer significativamente o desenvolvimento. Para um programador bom exige uma boa curva de aprendizado pra que o desenvolvimento seja de qualidade.	Neste caso, os profissionais precisam conhecer apenas sobre desenvolvimento web para aplicativos móveis, algo que simplifica o desenvolvimento e diminuem os custos. Mas o maior impacto no orçamento advém do fato de não se ter a necessidade de desenvolver o mesmo aplicativo para as diferentes plataformas, o que pode tornar o preço de 3 até 5 vezes mais barato.
Performance, confiabilidade, objetivo e público alvo	Se o seu aplicativo necessita consumir mais recurso de hardware ou está volta para um público mais exigente por performance o aplicativo nativo tende a atender melhor sua necessidade por ser mais rápido e confiável	Quando não se sabe exatamente qual plataforma seu público alvo mais utiliza ou se este público alvo é muito heterogêneo, desenvolver uma solução mais genérica pode ser mais vantajoso. Além disto, se seu aplicativo não possui requisitos não funcionais exigindo alto performance e também não tem problemas em exigir que o usuário esteja conectado à internet para utilizar o aplicativo, o ideal realmente seja um aplicativo híbrido

CONCLUSÃO

Neste artigo, foi apresentado um conjunto de critérios para avaliar as abordagens de desenvolvimento multi-plataforma para aplicações móveis. Os resultados foram compilados numa tabela e podem ser utilizados como referências.

Foi revelado que o desenvolvimento nativo não é necessário ao implementar sistemas para dispositivos mobile. Mesmo que apenas uma única plataforma deva ser apoiada, uma abordagem multi-plataforma pode revelar-se como o método mais eficiente, observando que são devidas principalmente ao uso de tecnologias da Web: HTML, CSS e JavaScript, com alinhamento de paradigmas da Web.

Isso é altamente adequado para o desenvolvimento de aplicativos multi-plataforma, porque eles são padronizados, populares, razoavelmente simples e bem suportados. Combinado com medidas adicionais para utilizar os recursos especiais dos dispositivos móveis, desde que cumpram os requisitos da maioria dos cenários móveis. No entanto, essa escolha nem sempre será a melhor, como por exemplo: interfaces de jogos são um campo exemplar em que a abordagem híbrida disponível pode ser um fracasso de escolha.

Em resumo, como pode ser observado, não há uma resposta exata para qual a melhor opção entre o desenvolvimento de aplicativo híbrido ou nativo. O que deve ser feito é: analisar os requisitos do projeto, a equipe, o conhecimento da equipe, o tempo para o aprendizado, e escolher a solução mais adequada.

REFERÊNCIAS:

1. GRØNLI, Tor-Morten et al. Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS. In: **Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on**. IEEE, 2014. p. 635-641.
2. DA SILVA, Marcelo Moro; SANTOS, Marilde Terezinha Prado. Os Paradigmas de Desenvolvimento de Aplicativos para Aparelhos Celulares. **Revista TIS**, v. 3, n. 2, 2014.
3. **IDC: Smartphone OS Market Share 2015, 2014, 2013, and 2012** – Disponível em: http://www.idc.com/prodserv/smartphone-os-market-share.jsp - Acesso em 25/04/2015.
4. **Android Developers** – Disponível em: http://developer.android.com/index.html - Acesso em 13/05/2015.
5. **iOS Developer Library - Apple Developer** – Disponível em: https://developer.apple.com/library/ios/navigation/ - Acesso em 20/05/2015.
6. **Recursos e ferramentas para profissionais de TI | TechNet** – Disponível em: https://technet.microsoft.com/pt-br - Acesso em 25/04/2015.
7. HARTMANN, G., et al. (2011). **Cross-platform mobile development. Tribal, Lincoln House, The Paddocks, Tech. Rep.** Disponível em: https://wss.apan.org/1539/JKO/mole/Shared%20Documents/Cross-Platform%20Mobile%20Development.pdf – Acesso em 16/06/2015.