Frameworks Mobile: Aplicações Híbridas X Nativas, Cordova X Android

[1] *Jonas Oliveira Francisco*
[1] *Aluno do curso de Análise e Desenvolvimento da FATEC-SP*
contato@jonasof.com.br – Github @jonasof
São Paulo, 2016

Introdução

Este texto aborda as vantagens e desvantagens das aplicações Híbridas em comparação com as aplicações Nativas no contexto de dispositivos móveis, a partir de uma abordagem histórica, teórica e prática.

As aplicações para dispositivos móveis são normalmente desenvolvidas usando as tecnologias preparadas pela fabricante do sistema operacional do aplicativo, e este tipo de aplicativo é aqui denominado chamado "Nativo". As aplicações acessíveis através de navegador, são chamadas "Web". Por fim, as aplicações que utilizam tecnologias web, mas são empacotadas e executadas como aplicativos instaláveis, são aqui definidas como "Híbridas".

História dos Dispositivos Móveis

Em 1984 surgiram os PDAs – Assistentes Digitais Pessoais, cujas funções mais utilizadas envolviam calendário, contatos, tarefas, entre outras. A partir de 1990, são lançados os primeiros modelos que permitiam a instalação de aplicativos de terceiros, instaláveis através da rede. É o caso do Pison EPOC e do seu concorrente, o PalmOS, este que aceitava aplicativos em C/C++ e já possuía tela sensível ao toque [3].

A partir de 2000, surgem os primeiros celulares que aceitavam a instalação de aplicativos Java J2ME, sendo que nesta época prevaleceram jogos, chats e apps básicos como browsers, mapas e calculadoras. Não se via, porém, aplicações atingindo as grandes economias e os negócios, como o Uber ou Internet Banking. Isto ocorreu tanto por limitações da banda, hardware, software, como principalmente por cultura [3].

Em 2007 foi lançado o iPhone com sistema operacional iOS, inaugurando uma nova indústria de smartphones, tanto em termos de hardware como em software. São característicos destes dispositivos o uso de tela multitouch, a possibilidade de realizar gestos como pinça e rolagem, o teclado virtual, a tela grande e os navegadores web "usáveis" [4].

Posteriormente em 2008 surgiu o sistema operacional móvel Android, o mais utilizado atualmente, bem como o Windows Phone, desbancando o produto anterior da Microsoft, o Windows Mobile. Posteriormente, os Sistemas operacionais móveis mais antigos perderam espaço de mercado até sumirem, como o Symbian (2011) e o Blackberry (2012).

Porém, em 2016 apenas duas plataformas, Android e iOS, são responsáveis por mais de 95% da quota de mercado de smartphones, segundo Gartner. Mesmo assim, são estes sistemas operacionais muito diferentes em termos de desenvolvimento de apps [6].

O problema da existência de diversos sistemas

Os anos posteriores à primeira versão do iOS foram acompanhados de várias tentativas de criação de sistemas concorrentes. De fato surgiram vários, exigindo dos desenvolvedores de apps a utilização de diferentes linguagens de programação. Por exemplo, para o desenvolvimento em iOS, era necessário utilizar Objective C e as respectivas APIs oferecidas pelo sistema operacional. O mesmo é aplicável para o Android, que utiliza Java, o Windows Phone - .NET, O Blackberry - também Java, O Ubuntu

Mobile – vários linguagens, o Firefox OS – HTML5, o Tizen – C, o Salfish – C++, entre outras.

Logo desenvolver para tantas plataformas diferentes, ou ao menos para as principais, tornou-se um problema para criadores de apps. Uma das opções adotadas trata da criação de uma página web que permita que o usuário acesse a aplicação via navegador, algo bastante usado em e-commerce por exemplo. Essa abordagem, apesar de ser a mais barata, tanto pela já sólida e aberta arquitetura web, tanto por funcionar em vários sistemas operacionais móveis, tem importantes desvantagens na interação humano-máquina. Por exemplo, o usuário não possui um atalho para abrir app web da mesma forma como pode abrir um app desenvolvido especificamente para o sistema operacional e disponibilizado numa loja de apps. Outro problema é dependência de conexão com a internet na maioria dos casos, e por fim a dificuldade em criar um design parecido com a de aplicações nativas.

Agora, ao desenvolver-se aplicações nativas, apesar dos ganhos em interação humano computador, perde-se tempo e recursos na criação e manutenção de diferentes bases de código.

Meg Wagner explica sucintamente tais vantagens e desvantagens [7]:

> *"Fazer um aplicativo nativo para iPhone é geralmente mais difícil que fazer um app web multi plataforma. E se você quiser desenvolver aplicativos Android e Blackberry, terá que construir cada plataforma do zero."*
>
> *[....]*
>
> *Aplicativos tem uma vantagem clara, em geral, se bem feitos, podem prover uma experiência de usuário muito melhor que os melhores websites mobile."*
>
> *Tradução livre – Mag Wagner*

Cordova – uma ferramenta de apps híbridos

Ora, para resolver o problema de multiplataforma, mantendo as vantagens do desenvolvimento de apps com tecnologia web, foi criada uma ferramenta em 2008 pela Nitobi Software, com nome Phonegap. A ideia da mesma é simples: desenvolver aplicativos instaláveis utilizando tecnologias web, as quais funcionam de modo semelhante em várias plataformas [6].

Em 2011, a Adobe comprou a Notobi Software e lançou o Phonegap com código aberto, renomeado "Cordova" e gerido pela Fundação Apache. Posteriormente a Adobe adotou a marca Phonegap para seu serviço de compilação em nuvem de aplicações Cordova [6].

O funcionamento de tal sistema é: arquivos html, js e css são empacotados de acordo com o sistema operacional de destino, que podem ser distribuídos normalmente em lojas de apps. Quando o app é iniciado pelo usuário, o sistema inicia uma webview (motor de navegador) que executará o principal arquivo html, indicado em config.xml, normalmente index.html. Durante este processo de início também são preparados os eventuais plugins embutidos no pacote. [8]

Estes plugins, oficiais e não oficiais, oferecem funcionalidades extras, sendo notórios os: Geolocation, Camera, Dialogs, estes oficiais [8], como também X-socialsharing, Barcodescanner, onesignal, fingerprint-id, mantidos por indivíduos e organizações terceiras.

O Cordova atualmente suporta várias plataformas, sendo as principais IOS, Android e Windows Phone, como também Blackberry e Ubuntu Mobile.

Desenvolvendo

Para criar um novo projeto com Cordova, após a instalação do mesmo, deve-se executar "cordova create nomedoprojeto". Isto criará uma nova pasta com um formato padrão: o config.xml guarda as configurações básicas. A pasta "www" contém arquivos da aplicação. A index.html é a primeira página a ser executada, tendo um formato parecido com este:

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="UTF-8">
```

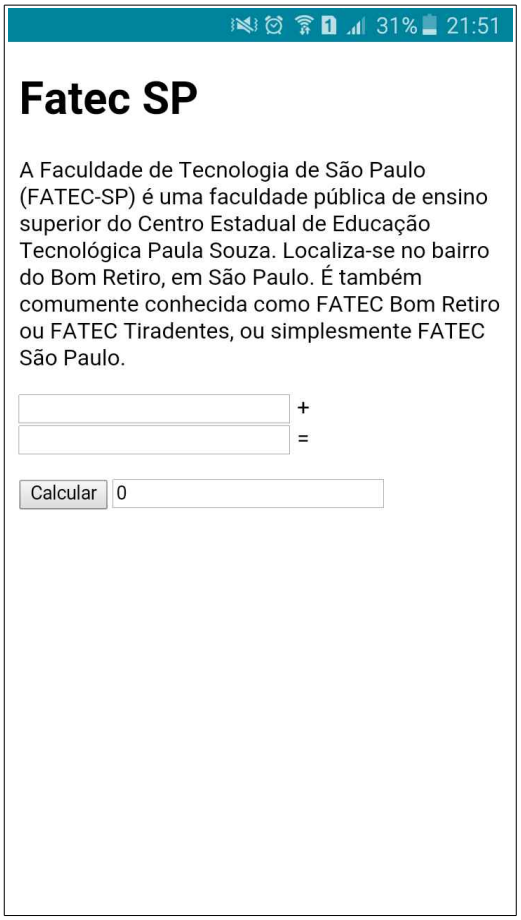
```
        <meta name="viewport" content="user-scalable=no,
initial-scale=1, maximum-scale=1, minimum-scale=1,
width=device-width">
        <link rel="stylesheet" type="text/css"
href="css/index.css">
        <title>Olá Mundo</title>
    </head>
    <body>
       <h1>Título</h1>
        <script type="text/javascript"
src="cordova.js"></script>
    </body>
</html>
```

Neste arquivo é possível usar qualquer elemento HTML. É importante ressaltar nele a importância do elemento "<meta name="viewport" content="user-scalable=no...", que garante que o usuário não realize o movimento de pinça para aumentar ou diminuir o zoom da aplicação. Também é fundamental o elemento "<script type="text/javascript" src="cordova.js"></script>", pois é ele que garantirá a utilização de plugins que acessem funcionalidades nativas do dispositivo.

É possível usar folha de estilos e scripts da mesma forma como aplicações web utilizam. É importante observar, porém, que scripts que necessitem realizar chamadas nativas através de plugins, devem ser executados após o dispositivo estiver pronto. É possível garantir isto usando o evento deviceready emitido pelo Cordova, por exemplo:

```
document.addEventListener('deviceready', function() {
    navigator.camera.getPicture(function(imageData) {
        var image = document.getElementById('myImage');
        image.src = "" +
imageData;
    });
}, false);
```

O resultado de um programa simples, que contém texto e um simples somador de dois números, está disponível em: https://github.com/jonasof/palestra_hibridoxnativo/tree/master/cordova_hello_pronto



Aplicação"Cordova Hello Pronto"

Conforme exibido na imagem acima, é muito custoso para o desenvolvedor fazer um aplicativo Cordova possuir um visual parecido com o de aplicações (botões, popups, menus, lista), isto porque não é a finalidade das tecnologias web (javascript, css e html) possuir tal acompanhamento. Por isso, existem frameworks que simulam elementos nativos das APIs nativas em elementos web. São os principais frameworks para esta tarefa o Ionic, o Framework 7 e o OnsenUI, sendo o primeiro o mais utilizado.

O resultado de um programa simples que utiliza o OnsenUI, que contém o texto e um simples somador de dois números, está disponível em: https://github.com/jonasof/palestra_hibridoxnativo/tree/master/onsenui

Aplicação "Onsen UI"

Desenvolvendo com Cordova

Apesar de ter tecnologias web, o processo de integração com as APIs dispositivo móvel exige a compilação através da SDK do respectivo sistema operacional de destino.

Qualquer que seja o destino, deve-se instalar as ferramentas básicas do Cordova. Elas são distribuídas através do NPM – Node Package Management.

Portanto deve-se instalar o Node.JS e, através do npm, o Cordova [13].

Desenvolvendo com Cordova - Android

Para compilar o aplicativo para o Android, deve-se instalar o Java JDK, o Android SDK e seus respectivos "Platform SDK" e "Build Tools", a Máquina Virtual AVD é opcional, que pode auxiliar o desenvolvimento [11].

Desenvolvendo com Cordova – IOS

Para o desenvolvimento para dispositivos IOS, é necessário instalar a IDE Xcode [12].

Publicando o aplicativo nas lojas

O framework Ionic possui uma suficiente documentação indicando os passos necessários para a publicação do aplicativo na Play Store e na App Store. O processo inclui a geração de assinaturas digitais [10]

Vantagens do Cordova

A principal vantagem do Cordova é o desenvolvimento de aplicações utilizando apenas uma base de código, que pode ser aproveitada não somente nos aplicativos mobile, como também entre as versões web desktop e mobile da aplicação.

Há também muito material sobre desenvolvimento web disponível em relação às plataformas nativas, portanto a curva de aprendizado pode ser menor.

Também existe a vantagem da tecnologia não ser proprietária.

Desvantagens do Cordova

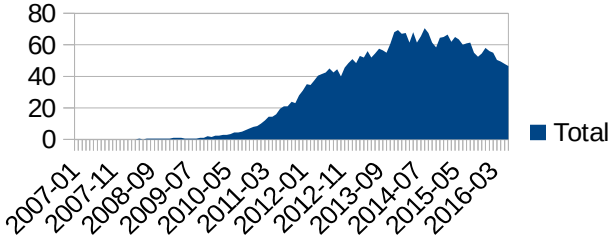
A principal desvantagem do Cordova refere-se ao seu desempenho inferior ao de aplicações mobile, consumindo mais memória RAM do dispositivo e processamento, sendo perceptível, em maior ou menor grau conforme potência do smartphones. [14]

Consumo de memória das aplicações "OnsenUI", "Cordova Hello Pronto" e a Calculadora padrão de smartphones Samsung

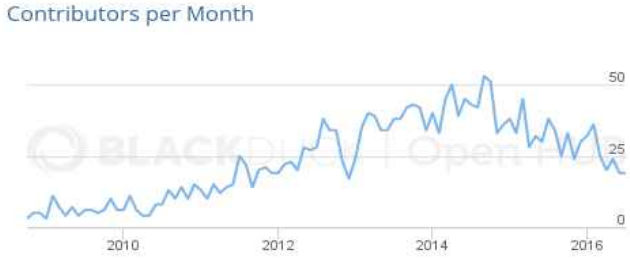
Nem todas as funcionalidades nativas estão disponíveis através de plugins, e muitos destes possuem precário suporte pois são mantidos por indivíduos sem muitas condições para o mantimento dos mesmos.

É possível identificar certa estagnação do projeto Cordova. Poucas novas funcionalidades são desenvolvidas, que possui tanto um lado vantajoso: desenvolver nesta ferramenta é seguro dado a menor ocorrência de *bugs*, como também ocorre o risco da ausência de interesse a longo prazo de manter a compatibilidade com novos e velhos sistemas operacionais.

Para ilustrar esta situação, o pico de popularidade do Cordova ocorreu entre 2014, medido tanto pelo Google Trends como pelo número de contribuidores ao projeto principal.



Google Trends – Cordova Programming e Phonegap Programming



Número de contribuidores no repositório cordova - https://www.openhub.net/p/apacheCordova

Há também alguns problemas ao se desenvolver com Cordova que devem estar claros. Um deles é o desempenho gráfico ruim em dispositivos que usam Android em versões inferiores a 4.4. A versão também é algo a se observar no aspecto de funcionalidades e recursos visuais faltantes em motores webkit antigos, alguns exemplos são o flexbox, calc() e o indexeddb. Por fim, o visual da aplicação nem sempre fica igual à dos aplicativos nativos, e existe um alguns problemas de acessibilidade.

Quando utilizar o Cordova

Deve-se questionar quais plugins serão necessários e se eles atenderão todos os casos. Algumas funcionalidades muito particulares podem não possuir biblioteca existente, ou depender de uma com pouco suporte [13].

Conforme explicado no item anterior, deve-se questionar quais versões de sistemas operacionais serão suportadas [13].

Deve-se questionar qual será o público-alvo da aplicação. Aplicativos de uso eventual e pouco intensivo podem perfeitamente serem desenvolvidos em Cordova. Usuários exigentes, porém, podem questionar o desempenho da aplicação.

Desenvolver em tecnologias nativas exige profissionais qualificados em cada plataforma, portanto deve-se verificar qual o orçamento do projeto a desenvolver para identificar o melhor custo-benefício.

Alternativas

Existem soluções mais atuais que provêm tanto a capacidade multiplataforma, quanto a performance nativa. São estas:

Microsoft Ace: Um projeto que possibilita a criação de elementos nativos utilizando código javascript ou XML. Isto permite integrar elementos de rápida renderização com o código web.

Native Script: Utiliza uma abordagem diferente que o Cordova: usam-se linguagens semelhantes à web, porém mais restritas. O Native Script então compilará o código em uma plataforma de execução nativa, permitindo uma performance muito superior à de aplicativos Cordova.

Native React: Desenvolvido pela Facebook, usa-se a plataforma react (sob linguagem typescript) para a criação de aplicativos próximos a performance nativos com a capacidade multiplataforma.

Xamarin: Desenvolvido pela Microsoft, utiliza C#. Com um runtime leve, permite a criação de aplicativos multiplataforma com performance superior à de aplicativos híbridos.

Recomendações

O site oficial do projeto Cordova contém vários tutoriais para iniciantes e já iniciados.[1].

O vídeo "Native, HTML5, and Hybrid Mobile App Development: Real-Life Experiences - Eran Zinman", apesar de antigo, cita as vantagens e desvantagens do desenvolvimento de aplicações híbridas.[2].

Os códigos, bem como os pacotes instaláveis para Android, usados para a criação dos aplicativos demonstrativos, está disponível em [15].

Referências

1. https://cordova.apache.org/docs/en/latest/
2. https://www.youtube.com/watch?v=We0byPckthQ
3. https://manifesto.co.uk/history-mobile-application-development/
4. http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad
5. https://en.wikipedia.org/wiki/Mobile_operating_system (gartner)
6. http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html
7. http://mashable.com/2013/11/01/native-or-hybrid-apps/#6w16aXhm7Eqz
8. https://cordova.apache.org/docs/en/latest/guide/overview/index.html
9. https://cordova.apache.org/docs/en/latest/guide/cli/
10. http://ionicframework.com/docs/guide/publishing.html
11. https://cordova.apache.org/docs/en/latest/guide/platforms/android/
12. https://cordova.apache.org/docs/en/latest/guide/platforms/ios/index.html
13. http://www.addthis.com/blog/2014/10/27/7-things-to-consider-when-making-ios-and-android-apps-with-cordova-or-phonegap/
14. https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-Ionic-framework-compared-to-native-apps
15. https://github.com/jonasof/palestra_hibridoxnativo