INTRODUÇÃO A DESENVOLVIMENTO DE APLICAÇÕES HÍBRIDAS

Henrique Leal Tavares[1]

henrique.tavares@fatec.sp.gov.br

Faculdade de Tecnologia de Garça – Fatec

Curso de Tecnologia em Analise e Desenvolvimento de Sistemas

Resumo. *Esse artigo tem como objetivo demonstrar e exemplificar como o desenvolvimento de aplicações para dispositivos móveis pode deixar de ser ímprobo e complexo, fazendo o uso de ferramentas que auxiliam para a evolução de uma determinada aplicação, farei esta demonstração utilizando frameworks e tecnologias específicas para este tipo de atividade. O estudo foi realizado por meio de consultas bibliográficas, criações de protótipos e aprofundamento de linguagens e bibliotecas que nos permitem essa facilidade, neste artigo citarei o Cordova, AngularJS e Ionic que nos auxiliam neste avanço híbrido.*

Palavras-chave: *Aplicações para dispositivos móveis. Frameworks. Cordova. AnguarJS. Ionic.*

Abstract. *This article aims to demonstrate and exemplify the development of application for mobile devices may no longer be unrighteous and complex, making the use of tools that assist in the evolution of particular application, I will make this demonstration using frameworks and specific Technologies for this type activity. The study was conducted through bibliographic queries, prototype creation and deepening of languages and libraries tha allow us this facility, this article will quote the Cordova, AngularJS and Ionic who assist us in this hybrid advance.*

Keywords: *Applications for mobile devices. Frameworks. Cordova. AngularJS. Ionic.*

---

[1]Alunos do Curso de Tecnologia em Analise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Garça-FATEC.

1. Introdução

Por diversas vezes é dito que os *smartphones* estão conquistando um enorme espaço no mercado e do consumidor, no Brasil não é diferente, nas projeções do estudo, o Brasil terá um smartphone ou tablete por cada habitante, com um total de 208 milhões, para o biênio 2017-2018. (MEIRELLES; FERNANDO, 2015).

Com isso surge uma nova modalidade de desenvolvimento *mobile*, que não limita-se a *smartphones*, o mesmo abrange dispositivos como *tablets*, relógios inteligentes e qualquer outra tecnologia vestível ou não.

O termo "computação vestível" ou "tecnologia vestível" refere-se a uma nova abordagem de computação, redefinindo a interação humano-máquina, onde os aplicativos estão diretamente conectados com usuário, em termos gerais, o usuário estaria "vestindo seu aplicativos". (CUETO, 2014).

Entre os sistemas que operam os dispositivos *mobile*, o que mais se destaca é o *Android*, o sistema operacional da *Google*, segundo o site sobre tecnologia Tecmundo: "domina 70.3% do mercado brasileiro de dispositivos móveis, ficando a Apple com o *IOS* em segundo lugar com 22% e *Microsoft* em terceira com 2.7% do mercado", sendo assim este artigo irá se basear no sistema do *Android* como plataforma para construir aplicações móveis, tanto pelo domínio no mercado quanto pela facilidade de desenvolvimento. De acordo com Ballve (2014):

> "Para começar, podemos destacar que existe cerca de 1,2 bilhão de usuários ativos em todo o mundo, e esse número é duas vezes maior que o total de usuários do iOS, por exemplo, o que mostra que as outras plataformas precisam percorrer um longo caminho para tentar igualar os números da plataforma da Google. "

2. Ferramentas de Desenvolvimento para Android - IDE

Existem várias formas de desenvolver aplicações móveis, por MWA (*Mobile Web Application*), *Nativamente* ou pela forma *Híbrida* (falaremos sobre isso depois).

As *IDE's* vieram para facilitar o cotidiano do analista/desenvolvedor, sua definição seria: *IDE Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. (LOZANO, 2010).

Atualmente existem várias *IDE's* para desenvolvimento nativo em *Android*, como o *CppDroid*, uma ferramenta bastante simples que pode ser executada até mesmo em um navegador.

Esta ferramenta tem sua biblioteca de linguagem com base em C++/C e a partir da versão 1.6 do *Android*, é possível compilar as aplicações desenvolvidas de forma nativa, tornando-se assim uma IDE muito utilizada pela sua praticidade.

Figura 1- Interface do CppDroid

Na disputa pelo mercado de ambiente de desenvolvimento integrado para Android também está o AIDE – *Android Java IDE*, a mesma suporta o ciclo completo de edição, compilação e execução do código fonte, demonstrando-se uma ferramenta completa com recursos de conclusão de código, verificação de erros, refatoração e até mesmo navegação de código fonte inteligente.

Sua linguagem é o *Java*, e o mesmo apenas compila em *Androids* na versão 2.2 ou superior, a versão paga desta ferramenta é integrada com o *DropBox*, onde suas alterações serão salvas em *cloud* em tempo real.

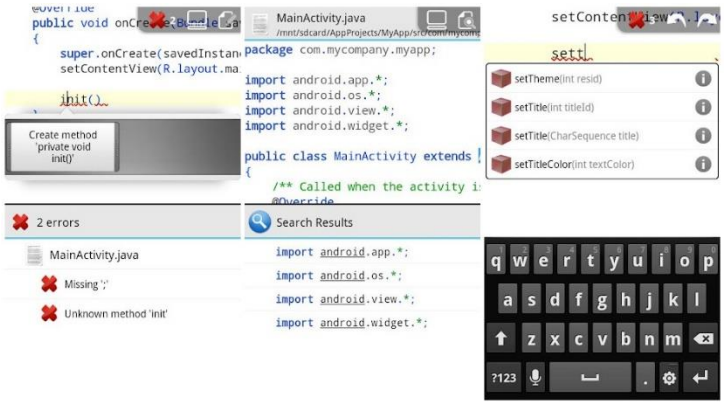

Figura 2 - Interface da ferramenta AIDE

Segundos dados levantados pela revista do DevMedia (2015): a ferramenta mais utilizada é o *Android Studio*, IDE desenvolvida pela *Google* para aplicações que são de origem da empresa, foi criada e apresentada no Google I/O de 2013, para auxiliar os desenvolvedores de aplicativos Android em algumas tarefas que parecem simples, porém são de importância vital para possíveis novos programas, a mesma tem compatibilidade para ser instalada em Windows, Linux e Mac OS X.

Sua linguagem nativa também é o Java, trazendo todos os benefícios que uma IDE necessite, mas sua proposta também é o desenvolvimento *RAD* (Rapid Application Development) ou Desenvolvimento Rápido de Aplicações, desta maneira traz diversos pacotes de componentes visuais como caixas de textos, botões, containers entre outros, também traz bibliotecas para acesso rápido ao *hardware* do *device*, como cache, câmera, memória, *services*, entre outros.

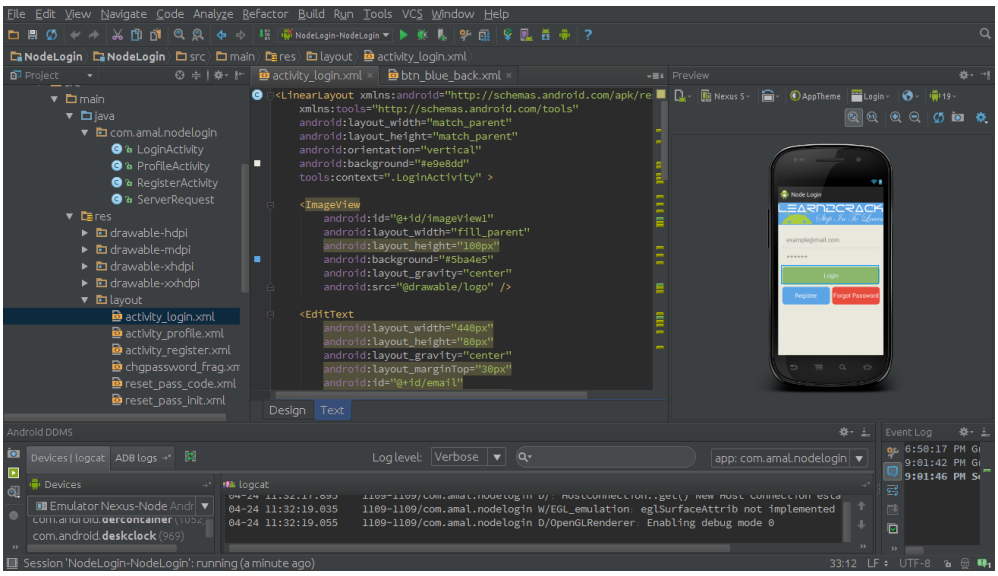

Figura 3 - Interface do Android Studio com emulador

3. Desenvolvimento Nativo

Antes de falar sobre o objetivo principal deste artigo (desenvolvimento híbrido) é necessário conhecer as outras formas para construir uma aplicação *Android*, os aplicativos nativos se definem como aplicações que residem no dispositivo e podem ser acessados através de ícones, por exemplo, os mesmos são instalados através de um aplicativo de loja, no nosso caso a Google Play, sendo que são desenvolvidos apenas para esta plataforma, podem aproveitar todas as funcionalidades do sistema operacional do dispositivo, como: câmera, GPS, acelerômetro, bússola, lista de contatos e afins. Também é possível aproveitar uso de gestos, sistemas de notificação nativos como o *push notification* (notificações de aplicativos foras de execução de primeiro plano) e funcionar sem conexão com a internet caso o conteúdo esteja embarcado.

Em suma, o desenvolvimento nativo é exclusivamente desenvolver para uma plataforma específica, porém seu desempenho acaba sendo mais rápido por ser embarcado no dispositivo, usando as funções de seu próprio hardware sem necessitar de importações externas, porém se optar por desenvolver nativamente terá que reescrever e planejar a aplicação para cada plataforma escolhida (Android, Windows Phone, IOS e etc), ou seja, o nativo é uma maneira direta de se comunicar com o smartphone/tablete e não fica dependente de outros recursos.

4. Desenvolvimento Mobile Web Application

Web Apps não são considerados aplicativos raízes, na realidade são sites que de diversas formas parecem com um aplicativo nativo. Eles são executados através de um navegador e tipicamente escritos em *HTML5*. Os usuários acessam inicialmente como fariam com um site: eles acessam determinada *URL* e tem a opção de “instala-lo” na tela principal de seu dispositivo, mas na verdade é criado um atalho para aquela página onde hospeda o serviço a ser utilizado.

São acessíveis funcionalidades semelhantes a um aplicativo nativo como: Esconder botões do navegador, gestos de navegação, com o cache do navegador é possível visualiza-lo *off-line*, usar GPS e links para ligações diretas.

Quando falamos sobre MWA (Mobile Web Application) falamos sobre “aplicativos” mais leves e que consomem menos memória de nosso dispositivo, pois o serviço que utilizaremos não estará instalado em nosso celular, mas sim hospedado em

um link que acessaremos pelo ícone de acesso a ele, um exemplo de MWA é o *Facebook Lite*, o aplicativo nativo do *Facebook* é capaz de consumir muita memória de armazenamento quando de processamento de nosso dispositivo, por isso a alternativa do *Facebook Lite*, um MWA que utiliza a memória de cache do navegador que ele é acessado, e seu processamento é executado em seu host, diminuindo assim o consumo em nosso dispositivo, porém, as suas funcionalidades são limitadas perante ao *app nativo*.

5. Desenvolvimento Híbrido

Os aplicativos híbridos são parcialmente nativos e parcialmente MWA. Como os nativos, eles devem ser baixados através de um aplicativo de loja, ficam armazenados na tela principal do dispositivo e podem aproveitar todas as funcionalidades do dispositivo (câmera, GPS, acelerômetro, gestos e etc). Como MWA, eles podem ser baseados em HTML5 e exibidos através de um navegador embutido no aplicativo, tendo parte ou conteúdo total carregado da web.

Os aplicativos híbridos são populares porque permitem desenvolvimento em multiplataformas, utilizando o mesmo HTML para diferentes sistemas operacionais – como através de ferramentas como o Cordova, PhoneGap e Sencha Touch permitem, inclusive compilando parar o formato nativo, reduzindo custos de produção.

A principal vantagem do desenvolvimento híbrido seria por conta da compatibilidade com os sistemas operacionais dos dispositivos, caso você opte por desenvolver da maneira híbrida em Android e quiser expandir sua aplicação para Windows Phone, por exemplo, terá de reescrever seu código Java Nativo em C#, depois em Swift ObjectiveC para IOS, com o desenvolvimento híbrido um código apenas pode ser compilado para várias plataformas, tornando assim esta metodologia mais prática e econômica, além do *CSS* proporcionar customizações ilimitadas com o front-end da aplicação.

6. Definição de Tecnologias Utilizadas

6.1. AngularJS

AngularJS é o mais novo lançamento do time de desenvolvedores do Google. Diferentemente de outros frameworks JavaScript, ele adota uma abordagem mais ligada

à sintaxe HTML, funcionando como uma espécie de extensão da linguagem. (FERREIRA, 2012).

6.2. Cordova

O Cordova é uma peça essencial para a composição do Ionic Framework com os dispositivos mobile, pois é ele quem faz o vínculo entre o front-end e os recursos do *device*.

> "O Cordova oferece um grupo de APIs que permitem desenvolver uma aplicação com HTML, CSS e JavaScript encapsulada como aplicação móvel nativa. A aplicação é executada no dispositivo móvel e pode acessar as funções nativas do dispositivo, como GPS ou câmera. Usando as APIs Cordova, um desenvolvedor consegue criar uma aplicação móvel sem escrever qualquer código nativo. " (ARRUDA, GIAN. Conceitos Básicos sobre Criação de uma Aplicação Cordova, *NETBEANS*, 15 de Janeiro de 2013).

6.3. Ionic

Ionic é um framework para desenvolvimento de aplicações parar dispositivos móveis que vida o desenvolvimento de apps híbridas e de rápido e fácil desenvolvimento. (GRILLO, 2015).

7. Trabalhando com Ionic

Para começar um projeto em Ionic, é necessário primeiro ter instalado em sua máquina a última release do SDK do Java, o download pode ser encontrado na página de downloads do site da Oracle: http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html, também é necessário a última release do Android, para isto é precisa-se instalar o Download Manager do Android, o mesmo contém todas as versões, documentações e correções já lançadas pela Google: http://developer.android.com/intl/pt-br/sdk/index.html.

Também será necessário instalar um servidor local para podermos compilar o AngularJS entre outras funcionalidades do framework, será ele o NodeJS, o mesmo está disponível em sua página institucional: https://nodejs.org/en/. Feito isso o NodeJS será responsável por instalar o NPM (Node Package Manager), sua principal função para nós será a instalação de pacotes de repositórios online, para instalar o Cordova e Ionic no prompt de comando de sua máquina digite: *npm install –g cordova ionic*, esta instrução

será responsável por buscar no repositório online de ambas as tecnologias e instalar as mesmas em sua máquina.

Para verificar se todas as ferramentas foram instaladas com sucesso verifique da seguinte forma ainda no prompt de comando:

- *Java –version* (verificará a versão instalada do SDK do Java).
- *Android –v* (verificará a versão instalada do SDK do Android).
- *Cordova –v* (verificará a versão da biblioteca do Cordova instalada).
- *Ionic –v* (verificará a versão do framework do Ionic instalado).

Existem 3 maneiras básicas de iniciar um projeto em Ionic, *BlankApp*, *Tabs* e *SideMenu*.

O *BlankApp* iniciará uma aplicação totalmente em branco, sem qualquer tipo de componente Front ou Back-End, para criar esta aplicação, no prompt de comando deve ser inserido: *ionic start minhaAplicacao blank*.

- *Ionic* (chama a biblioteca Ionic);
- *Start* (executa o comando start da biblioteca do Ionic, responsável por iniciar um novo projeto);
- *minhaAplicacao* (variável do nome da aplicação a ser criada, o nome pode ser definido a sua escolha);
- *blank* (parâmetro que define o modelo de projeto que vamos utilizar).

Tabs criará uma aplicação com abas e um menu de cabeçalho como exemplo, os mesmos são totalmente customizáveis, para criar esta aplicação, no prompt de comando deve ser inserido: *ionic start minhaAplicacao tabs*.

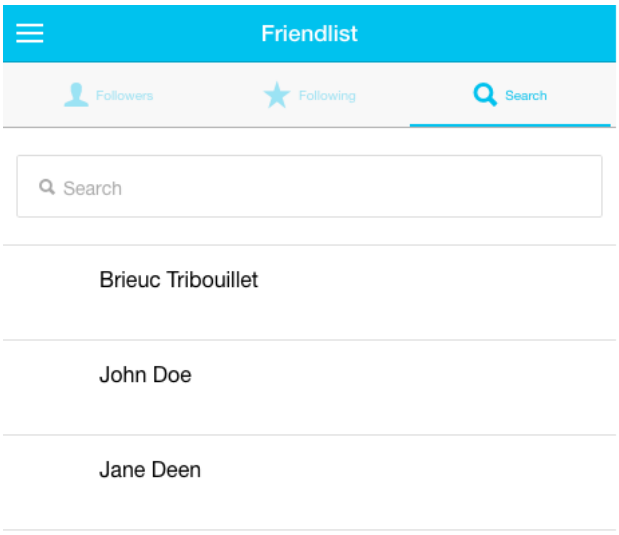

Figura 4 - Projeto iniciado com Tabs no Ionic

SideMenu criará por sua vez um projeto com um menu lateral e navegação entre abas quando uma opção do menu é selecionada, para criar esta aplicação, no prompt de comando deve ser inserido: *ionic start minhaAplicacao sidemenu*.

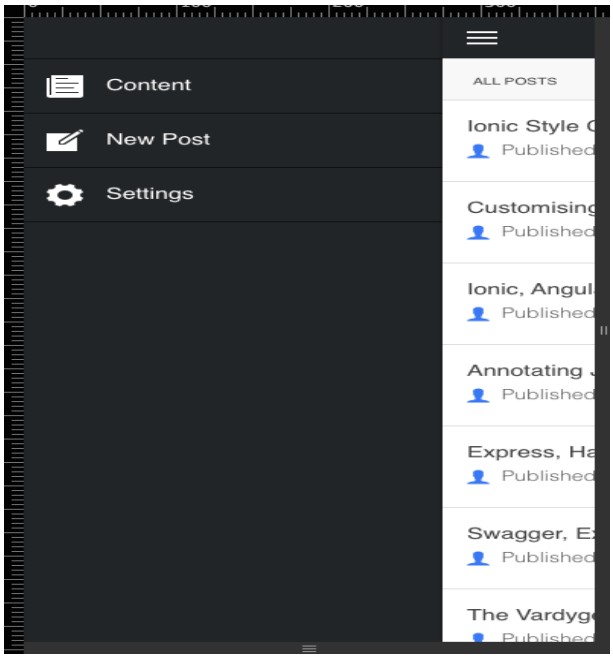

Figura 5 - Projeto iniciado com SideMenu no Ionic

Após escolhido o modelo de projeto o mesmo pode ser visualizado digitando o seguinte comando no prompt de comando: *ionic serve*, o mesmo iniciará o *browser* definido como padrão na máquina e irá simular o uso da aplicação como se estivesse sendo utilizado em um dispositivo móvel, para o desenvolvimento em Ionic.

Para testar o aplicativo em um dispositivo mobile é necessário acessar o prompt de comando do computador e redirecionar sua pasta até onde nossa aplicação foi criada, por padrão o Ionic cria novos aplicativos dentro da pasta do usuário da sessão do computador seguido por o nome do aplicativo escolhido, por exemplo: *C:\Users\Henrique\minhaAplicacao*, onde *\Henrique* é meu usuário de acesso ao terminal da máquina e *\minhaAplicacao* foi o nome definido para o aplicativo em questão, ainda no prompt de comando é preciso estabelecer aonde nosso sistema operacional encontrará o arquivo *SDK* utilizado no nosso aplicativo, para isso utilizamos as seguintes instruções:

- *set ANDROID_HOME=C:\\android-sdk-windows* (cria uma variável no sistema operacional com o diretório onde nosso arquivo SDK foi instalado, recomenda-se colocá-lo na raiz das pastas do sistema).
- set PATH=%PATH%;ANDROID_HOME%\tools;%ANDROID_HOME%\platform-tools (instancia as ferramentas da plataforma do *SDK*).

Logo após será necessário introduzir os arquivos de compilação e execução dentro da nossa aplicação, para isso usaremos a seguinte instrução:

- *ionic platform add android*

  Agora para gerarmos os arquivos de instalação é utilizado:

- *ionic build andoid*

  Pronto, serão gerados dois arquivos dentro da pasta que o aplicativo se encontra, basta adicionar esses arquivos no dispositivo a ser testado e instala-los normalmente.

8. Considerações Finais

As metodologias de aplicações híbridas foram desenvolvidas para revolucionar e facilitar o desenvolvimento mobile, desta forma é apoiada por grandes empresas no ramo de software e versões e atualizações são publicadas com frequência, já está em teste a versão release do Ionic2 que irá acompanhar uma atualização do AngularJS pela Google, suas arquiteturas só tendem a melhorar, mas é preciso manter-se atualizado sobre novas funcionalidades e possibilidades que as ferramentas nos proporcionam, tanto quanto API que poderemos consumir para incrementar nossos projetos.

O projeto que foi desenvolvido para teste e argumentação deste artigo foi utilizado diversas opções que o leque de funcionalidades do IONIC com MVC proporciona, fazendo assim um projeto menos complexo em seu código fonte pela clareza da metodologia distribuída entre controllers e ganho de tempo, fazendo assim, ao final do projeto o mesmo conseguir ser executado em 3 plataformas mobile, com os mesmos códigos e interfaces.

9. Referências bibliográficas

ANGULARJS FRAMEWORK, 2016. Institucional. Disponível em: <https://docs.angularjs.org/tutorial> Acesso em: 01 de Abril. 2016.

ARRUDA, Gian. Conceitos Básicos sobre Criação de uma Aplicação Cordova, NetBeans, 2013. Disponível em: <https://netbeans.org/kb/docs/webclient/cordova-gettingstarted_pt_BR.html> Acesso em: 12 de Maio de 2016.

ARRUDA, Saulo. Design reponsivo, Desenvolvimento Nativo e App Híbrido, quando usar? Jera, 2015. Disponível em: <http://jera.com.br/blog/4931/design-ux/design-responsivo-o-que-e> Acesso em: 09 de Abril. 2016.

BALLVE, Marcelo. The Android Report, Business Insider, 2014. Disponível em: <http://www.businessinsider.com/android-e-commerce-and-market-numbers-2014-11> Acesso em: 04 de Abril. 2016.

GRILLO, Rafael. Introdução ao Ionic Framework, Tableless, 2015. Disponível em: <http://tableless.com.br/introducao-ao-ionic-framework/> Acesso em: 12 de Maio de 2016.

IONIC FRAMEWORK, 2016. Institucional. Disponível em: <http://ionicframework.com/getting-started/> Acesso em: 09 de Abril. 2016.

MEIRELLES, Fernando. Número de smartphones supera o de computadores no Brasil, Exame, 2015. Disponível em: <http://exame.abril.com.br/tecnologia/noticias/numero-de-smartphones-supera-o-de-computadores-no-brasil> Acesso em: 12 de Maio de 2016.