

Universidade do planalto catarinense

Curso de sistemas de informação

(Bacharelado)

Luana rodrigues de andrade

ESTUDO AVALIATIVO Entre o DESENVOLVIMENTO DE

APLICAÇÕES Móveis HÍBRIDAS E NATIVAS

Lages (SC)

Ano (*2017*)luana rodrigues de andrade

ESTUDO AVALIATIVO Entre o DESENVOLVIMENTO DE

APLICAÇÕES Móveis HÍBRIDAS E NATIVAS

Trabalho de Conclusão de Curso submetido à Universidade do Planalto Catarinense para obtenção dos créditos de disciplina com nome equivalente no curso de Sistemas de Informação - Bacharelado.

Orientação: Prof(ª). Alexandre Rech. Esp.

Co-Orientador: Prof(ª). Marco Aurélio Cevey

Lages (SC)

Ano (*2017)*

Luana Rodrigues de Andrade

ESTUDO AVALIATIVO Entre o DESENVOLVIMENTO DE

APLICAÇÕES Móveis HÍBRIDAS E NATIVAS

Este relatório, do Trabalho de Conclusão de Curso, foi julgado adequado para obtenção dos créditos da disciplina de Trabalho de Conclusão de Curso, do 8º. semestre, obrigatória para obtenção do título de:

Bacharel em Sistemas de Informação

Lages (SC), Dia de Novembro de 2017 (data da defesa)

|  |  |
| --- | --- |
| Prof. Alexandre Rech, esp.  **Orientador** | Prof. Marco Aurélio Cevey  **Co-Orientador** |

Banca Examinadora:

|  |  |
| --- | --- |
| Prof. Hugo Estevam Romeu Longo, Esp.  **UNIPLAC** | Prof. Ismael Dos Santos, Esp.  **UNIPLAC** |

|  |
| --- |
| Prof. Claiton Camargo de Souza  **Coordenador de Curso** |

Dedico este trabalho aos meus pais, Caio e Elaine, por todo amor, dedicação e carinho durante a graduação. Ao meu cachorro Malcolm, que me fez sorrir durante todos os momentos de frustação além de permanecer ao meu lado, mesmo quando assustado ao me ouvir estudar para alguns seminários. Ao meu namorado Filipe, por todas as vezes que não me deixou desistir e pelas noites em que ficou até tarde me auxiliando. E ao meu irmão Luan, que nunca relutou em ajudar quando precisava. A eles, o meu eterno amor.

Item OPCIONAL, deve ficar posicionado ao final da folha.

É uma menção onde o autor presta homenagem ou dedica o trabalho a alguém.

Lista de Ilustrações

**FIGURA 1 -** Plataformas de desenvolvimento, linguagens e ferramentas. 18

**FIGURA 2 -** Estrutura e recursos de aplicações nativas 19

**FIGURA 3 -** A pilha de Softwares do Android 21

**FIGURA 4 -** Estrutura e recursos de uma Web APP. 24

**FIGURA 5 -** Estrutura e recursos de uma aplicação híbrida 25

**FIGURA 6 -** Funcionamento do Cordova 26

**FIGURA 7 -** Importância do desempenho em aplicações móveis 32

**FIGURA 8 -** Telas de login e visualização dos aplicativos 35

**FIGURA 9 -** Tela de cadastro de corridas 36

**QUADRO 1 -** Vantagens e desvantagens de aplicações nativas e híbridas 36

Lista de abreviaturas e Siglas

API - Application Programming Interface

APP - Application

CSS - Cascading Style Sheet

DOM - Document Object Model

ES6 - ECMAScript 6

GPS - Global Positioning System

HAL - Hardware Abstraction Layer

HTML - Hypertext Markup Language

IDE - Integrated Development Environment

OHA - Open Handset Alliance

SDK - Software Development Kit

SMS - Short Message Service

SO - Sistema Operacional

UI - User Interface

URL - Uniform Resource Locator

Sumário

1 Introdução 6

1.1 Apresentação 6

1.2 Descrição do problema 7

1.3 Justificativa 7

1.4 Objetivo geral 8

1.5 Objetivos específicos 8

1.6 Metodologia 9

2 Abordagens de desenvolvimento para dispositivos móveis 10

2.1 Abordagem Nativa 10

2.1.1 Android 12

2.2 Mobile Web APPs 16

2.3 Abordagem Híbrida 17

2.3.1 Apache Cordova 18

2.3.2 Angular 2 20

2.3.3 Ionic Framework 2 21

2.4 Conclusão do Capítulo 22

3 Benchmark 24

3.1 Parâmetros de Comparação 24

3.1.1 Esforço de Desenvolvimento 25

3.1.2 Desempenho 25

3.1.3 Interface 27

4 Aplicações Experimentais 28

4.1 Exemplo de Uso 28

4.1.1 Prototipação de Telas 28

4.2 Tecnologias Utilizadas 30

5 Resultados preliminares 31

Referências Bibliográficas 33

Bibliografia complementar 36

1. Introdução
   1. Apresentação

Nos dias atuais, é de senso comum que o uso de *smartphones* e aplicativos vem crescendo frequentemente. Dia após dia, especulações sobre um novo aplicativo revolucionário vêm sendo comentadas.

O mercado de desenvolvimento de aplicações móveis aumentou, fazendo com que empresas que antes só desenvolviam *softwares web/desktop*, tivessem um novo horizonte a ser descoberto e com isso a dúvida sobre qual abordagem de desenvolvimento móvel adotar, torna-se frequente.

No início, quando o desenvolvimento de aplicações móveis começou a tornar se comum, a única opção era desenvolver aplicativos exclusivos a cada plataforma de desenvolvimento, um código diferente para cada uma era necessário, utilizando os recursos de cada uma. Segundo Machado (2014, p.1) “Nos dias atuais, existem diversos sistemas operacionais para dispositivos móveis, com diferentes fatias de mercado e diferentes métodos de desenvolvimento. ” A respectiva abordagem, é chamada de nativa.

Atualmente, uma nova abordagem vem se popularizando no mercado, a abordagem híbrida, que consiste em criar um aplicativo com tecnologias web, e que são incorporadas em diferentes sistemas operacionais. Ou seja, um único código, que funciona em diferentes plataformas.

Sabe-se, que a competitividade no mercado de desenvolvimento é crescente, e que diversas variáveis podem fazer de um aplicativo um sucesso ou um fracasso. Além de tempo de desenvolvimento, desempenho e custo, muitas outras questões devem ser levantadas quando se inicia o desenvolvimento.

“Grandes empresas costumam realizar todos os tipos de pesquisas de campo antes de começar o desenvolvimento de qualquer produto. ” (Machado, 2014, p.1).

O presente estudo avaliativo, será divido em capítulos. A partir do capítulo 1 são apresentados o projeto do trabalho, a definição do tema escolhido, descrição do problema, justificativa, objetivos gerais e específicos, além da metodologia. Já no capítulo 2, o embasamento teórico é apresentado, descrevendo as abordagens de desenvolvimento nativas e híbrida com html5 embarcado e as tecnologias que as cercam. No capítulo 3 os parâmetros utilizados para auxílio da avaliação serão descritos. O capítulo 4 e 5, serão de elaboração de exemplos e avaliação final respectivamente.

* 1. Descrição do problema

O número de *smartphones* em uso, vem crescendo significativamente nos últimos anos, e diante disso é imprescindível que o aumento no desenvolvimento de aplicações móveis aconteça, e que novas tecnologias nasçam. Dos dias atuais até os próximos 20 anos, a maioria dos segmentos comercias serão consumidos por essas novas tecnologias, e variáveis como custo de produção, manutenção e evolução, ainda serão discutidas (LIMA, 2016, p.1).

Atualmente, a dúvida entre as diferenças no desenvolvimento e produto de aplicações híbridas com HTML5 embarcado e nativas vem crescendo. Sabe-se que o desenvolvimento de aplicações móveis pode tornar-se complexo e caro, quando não adotado o melhor método de desenvolvimento. Diante disso, como uma pesquisa científica pode auxiliar na tomada de decisão, da melhor abordagem de desenvolvimento de aplicações móveis?

* 1. Justificativa

*Smartphones* são considerados telefones inteligentes, e fazem parte da vida de bilhões de pessoas diariamente. De acordo com Capelas (2016), em maio de 2016, o número de *smartphones* em uso no Brasil chegou a cerca de 168 milhões, ultrapassando a venda de computadores e se tornando o principal meio de comunicação atual. Com o aumento no uso desses *smartphones*, o desenvolvimento de aplicações móveis vem crescendo consideravelmente. A todo instante, novos aplicativos são publicados em lojas, de diferentes plataformas, e paralelo a isso, o aumento nas opções de abordagens de desenvolvimento cresce junto.

Uma das decisões mais importantes a se tomar por uma empresa antes de iniciar no desenvolvimento de um aplicativo para dispositivos móveis é a abordagem a ser adotada. Sabe-se que nem sempre a escolha de uma abordagem demanda apenas “afinidade”, pois o ciclo de desenvolvimento é complexo e pode se tornar caro.

Na prática, acreditamos que o ciclo de desenvolvimento de um produto mobile passa por três fases: descobrir, desenvolver e gerenciar. Essas fases não formam um processo, mas um ciclo. Isso porque ao desenvolver podemos ter um feedback que nos faça voltar à fase de descobrir, bem como gerenciando podemos entender que é melhor desenvolver de outra forma. (RIVA, 2014, p.1)

Quando se fala de abordagem híbrida e nativa, sabe-se que ambos os modelos de desenvolvimento possuem vantagens e desvantagens aparentes, o que torna a escolha ainda mais desafiadora. Quando se trata de analisar o desenvolvimento nativo e híbrido, é necessário estudar a fundo as características de cada abordagem, para que o objetivo de atender o mercado de forma positiva, seja atendido.

Sabendo disso, o interesse pelo tema proposto surgiu a partir da necessidade de conhecer as abordagens de desenvolvimento híbrida e nativa e analisar suas viabilidades, auxiliando em escolhas futuras de qual seguir.

* 1. Objetivo geral

Analisar as abordagens de desenvolvimento móveis híbrida e nativa, a fim de compara-las, levando em consideração suas vantagens e desvantagens.

* 1. Objetivos específicos

Os objetivos específicos desse trabalho são.

1. Explorar e descrever as abordagens de desenvolvimento híbrida e nativa.
2. Identificar as vantagens e desvantagens das abordagens híbrida e nativa.
3. Realizar um estudo de caso comparativo entre as abordagens, utilizando Ionic *Framework* 2 e Android
   1. Metodologia

Este trabalho caracteriza-se por uma pesquisa científica, e para alcançar os objetivos finais, algumas etapas serão necessárias, conforme descritas a seguir.

A primeira etapa deste trabalho, consistiu em levantar pesquisas já realizadas que evidenciem o desenvolvimento móvel, para que então se obtenha, uma base do que seria necessário enfatizar e enriquecer no/o trabalho, além de traçar os seus objetivos.

A segunda etapa deste trabalho, é caracterizada pelo estudo teórico e descrição acerca das abordagens de desenvolvimento móveis nativa e híbrida com HTML5 embarcado, mostrando a partir do referencial teórico obtido, as ferramentas utilizadas para cada abordagem, *frameworks* e objetivos. Afim de limitar o escopo do trabalho, durante essa etapa, também foram definidas as tecnologias de desenvolvimento utilizadas, a abordagem nativa será representada/baseada no sistema Android, e a abordagem híbrida no Ionic *Framework* 2.

A terceira parte desse trabalho, descreve os parâmetros analisados durante a elaboração dos protótipos, desenvolvidos utilizando o Ionic *Framework* 2 e o Android, para que então na quarta etapa desse trabalho, os resultados da análise possam ser descritos.

Por fim, como descrito acima, as abordagens de desenvolvimento móvel híbrida e nativa serão comparadas, levando em consideração os parâmetros selecionados, e o embasamento teórico

1. Abordagens de desenvolvimento para dispositivos móveis

Segundo Karch (2016), aplicativos (APPs) são um pedaço de software que pode ser executado na *Internet*, no computador e em dispositivos móveis. Essas APPs, quando desenvolvidas para dispositivos móveis são disponibilizadas pelos fornecedores através de lojas virtuais como a *Apple Store* (loja virtual da Apple), e a *Play Store* (loja virtual da Google), e são baixadas diretamente no celular.

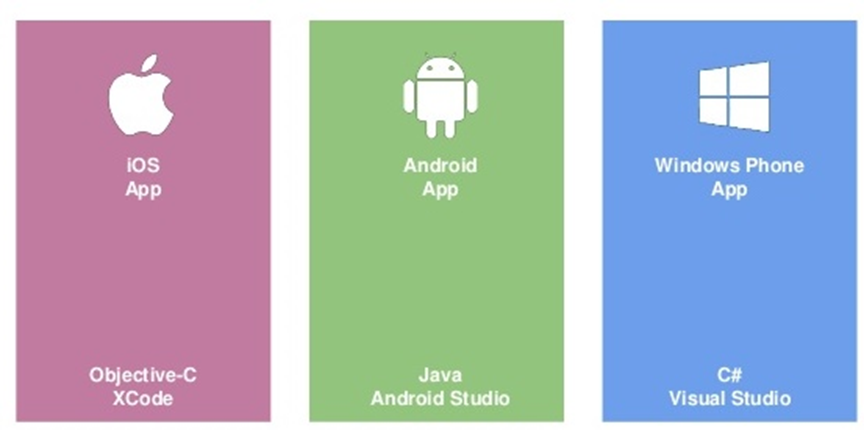
Ao iniciar o desenvolvimento de uma aplicação móvel, a escolha do tipo de aplicativo é algo minuciosamente importante. O mercado de dispositivos móveis, possuí diferentes fabricantes, sistemas operacionais e *hardware* e talvez uma das decisões mais importantes a se tomar antes de entrar no mundo do desenvolvimento móvel, é qual abordagem adotar.

A partir disso, esse capítulo apresentará as abordagens de desenvolvimento de aplicações móveis Nativa (exemplificando com Android), *Web APP*, e híbrida (exemplificando com Ionic).

* 1. Abordagem Nativa

Aplicativos nativos, são programas desenvolvidos para uso em uma plataforma ou dispositivo específico. Nessa abordagem de desenvolvimento quando pretende-se atingir uma gama de plataformas de dispositivos é necessário criar uma versão do aplicativo separada para cada plataforma existente. Além disso é preciso levar em conta as diferentes linguagens, sistemas operacionais, SDKs (kits de desenvolvimento de *software*), IDEs (ambientes de desenvolvimento integrado) e arquiteturas de cada plataforma, o que segundo Arruda (2014) torna o desenvolvimento mais demorado e caro. Para o desenvolvimento para a plataforma Android, como exemplo, utiliza-se as ferramentas, linguagens e bibliotecas disponíveis para Android, como a linguagem Java, e os aplicativos são publicados na loja da Google, a *Play Store*. Para desenvolver aplicações destinadas a plataforma IOS, utilizando a abordagem nativa, utiliza-se ferramentas, linguagens e bibliotecas disponíveis pra IOS, como o *Objective C*, e publica-se na loja da Apple, *App Store*. Já para desenvolver aplicações voltadas ao Windows Phone, utiliza-se a linguagem C#, com Visual Studio e o resultado é publicado na loja do Windows. Isso pode ser visto de forma resumida na Figura 1.

1. Plataformas de desenvolvimento, linguagens e ferramentas.



(FONTE: NOSER BLOG, 2015)

Como os aplicativos nativos são desenvolvidos para uma plataforma específica, eles são capazes de aproveitar ao máximo os recursos disponíveis do sistema operacional e dos *softwares* instalados nessa plataforma, como câmera, GPS, calendário e álbum de fotos. Além disso, a interação dentro de aplicativos nativos, se assemelha com a maioria dos outros aplicativos nativos instalados no dispositivo, o que segundo ABED (2016) faz com que o usuário final fique mais propenso a aprender rapidamente como utilizar o aplicativo, pois a medida em que os usuários navegam, o conteúdo, estrutura e elementos já estão no aparelho, disponíveis para serem carregados instantaneamente.

A Figura 2, ilustra a estrutura de uma aplicação nativa, e os recursos disponíveis a ela.

1. Estrutura e recursos de aplicações nativas



FONTE: (BERGANTIN, 2014)

* + 1. Android

Segundo Meyer (2016), Android é uma plataforma nativa voltada para dispositivos móveis, que surgiu em 2003 na Califórnia, pelos empresários Andy Rubin, Rich Miner Nick Sears e Chris White fundadores da Android Inc. A plataforma foi vendida em 2005, dois anos após seu lançamento, para a Google, e atualmente é mantida pela Open Handset Alliance (OHA), um grupo de aproximadamente 84 empresas unidas para inovar e acelerar o desenvolvimento de aplicações, trazendo aos consumidores de dispositivos móveis uma rica experiência.

O Android é considerado atualmente o sistema operacional móvel mais conhecido do mundo.

O Android é a plataforma mobile mais utilizada do planeta, existem mais pessoas com telefones e *tablets* rodando Android do que qualquer outra plataforma *mobile* no mundo, são milhões de usuários. Com esse crescimento acelerado, os aplicativos Android ganham cada vez mais força e prioridade no mundo *mobile*. (FLORENZANO, 2015, p.1)

* + - 1. Desenvolvimento

O sistema operacional Android é de código aberto, o que denota que ele não é ligado exclusivamente a um fabricante de *hardware*.

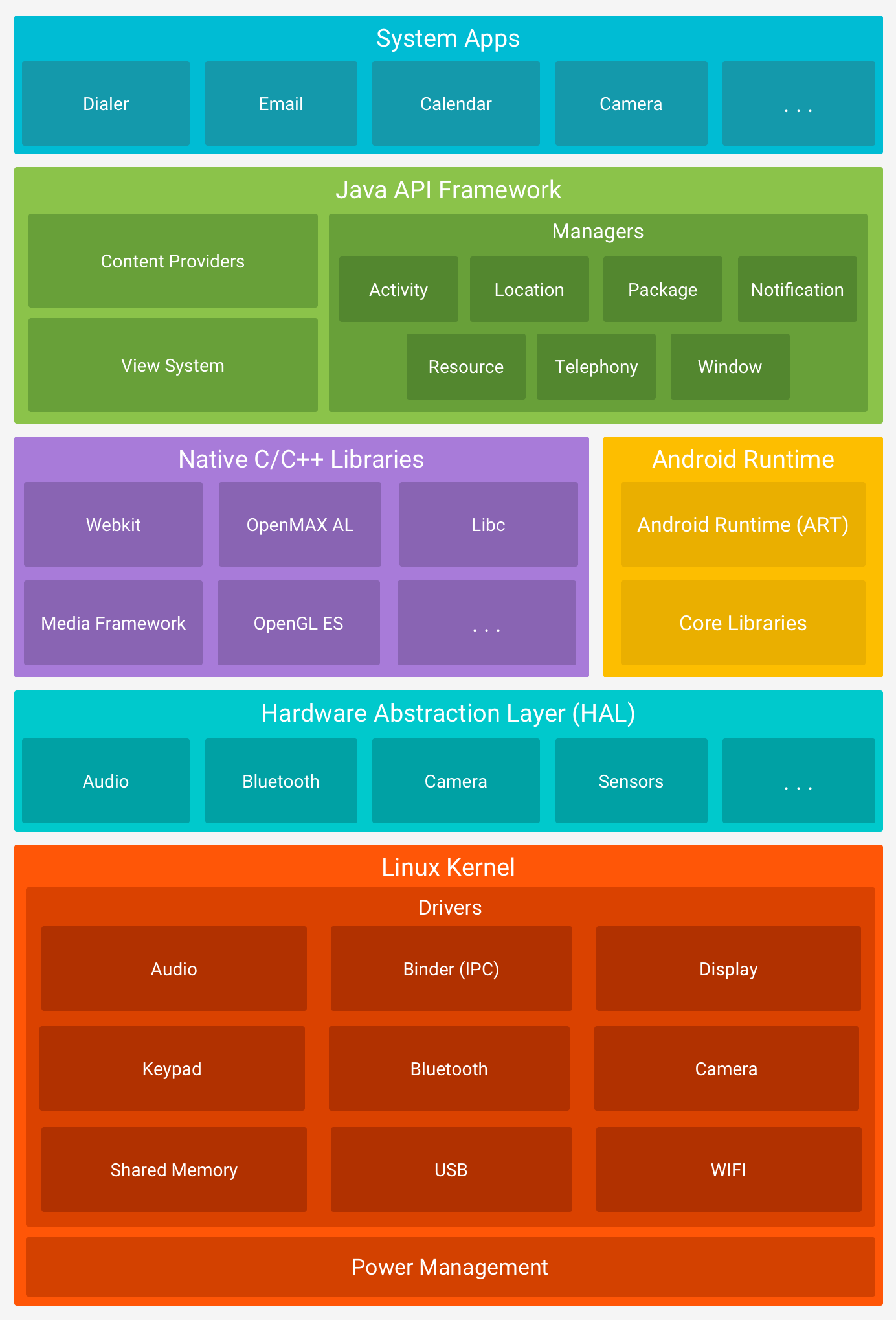
O código-fonte aberto permite que fabricantes de telefones criem novas *interfaces* para o usuário, e adicionem recursos internos para alguns dispositivos. Isso também coloca todos os desenvolvedores em um mesmo nível. (DELFINO, 2014, p.1)

Para desenvolver aplicativos para sistemas Android, é preciso utilizar da linguagem Java e um ambiente de desenvolvimento integrado (IDE), como Android Studio, recomendado pela Google e que provém as ferramentas necessárias para desenvolver e testar as aplicações. É possível desenvolver aplicações Android utilizando um computador com qualquer sistema operacional, como Linux, Windows, MAC OS X etc.

* + - 1. Arquitetura

O Android é uma pilha de *softwares* com base no Linux, que são agrupados em camadas. Essas camadas estão disponíveis na Figura 3, e serão detalhadas a seguir.

1. A pilha de Softwares do Android

FONTE:(ANDROID, 2017)

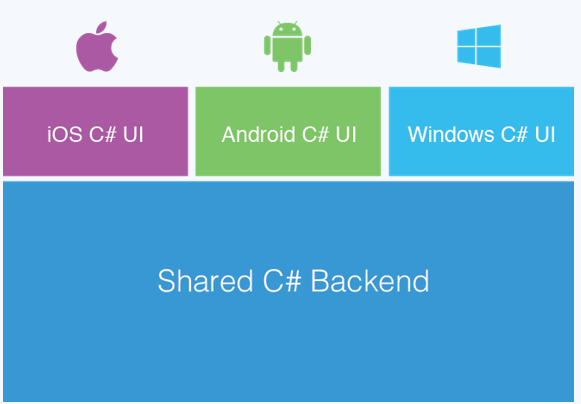
1. O primeiro nível a ser detalhado é a da camada *system apps,* nela encontram-se as aplicações e funções básicas do dispositivo. Segundo Gadelha (2011), a camada de aplicações é a responsável pela interação entre o usuário e o dispositivo móvel, nesta camada encontram-se aplicativos de e-mail, programa de SMS, calendário, mapas, navegador, entre outros.
2. No segundo nível, encontra-se a camada Java API *Framework*, ou como é mais conhecida “*application framework*”, essa camada abriga os recursos completos do sistema operacional, disponíveis em função das APIs, que formam blocos de código, utilizados para criação de aplicativos Android. De acordo com Android (2017), os desenvolvedores possuem acesso as mesmas APIs utilizadas no desenvolvimento do sistema operacional Android.
3. No terceiro nível temos duas camadas, a camada de bibliotecas do SO, e a camada Android *Runtime*.
4. A camada de bibliotecas, é responsável pelo conjunto de instruções que descrevem ao dispositivo como lidar com diferentes tipos de dados. “Vários componentes e serviços principais do sistema Android, são implementados por código nativo que exige bibliotecas nativas programadas em C e C++.” (Android, 2017).
5. No terceiro nível, também se encontra a camada Android *Runtime* (tempo de execução). Segundo Gadelha, essa camada inclui um conjunto de bibliotecas do núcleo Java (*Core Libraries*), e a máquina virtual Dalvik, que é utilizada pelo Android para rodar cada aplicação com seu próprio processo. Isso permite que nenhuma aplicação seja dependente da outra, ou seja, múltiplas instâncias executando ao mesmo tempo.
6. No quarto nível, fica a camada de abstração de *hardware* (HAL), essa camada fornece interfaces padrão, que exibem as capacidades de *hardware* do aparelho para a Java API. Segundo a Android (2017), a HAL, tem módulos de biblioteca, que implementam uma interface para cada tipo de componente de *hardware*, como a câmera ou *bluetooth*. Quando algum *framework* da API, faz uma chamada para acesso ao *hardware* do aparelho, o Android carrega um modulo da biblioteca para esse componente de *hardware*.
7. O último nível da arquitetura, ou a base, é a camada de kernel do Linux, segundo a Android (2017), o Android é fundado no kernel do Linux, o que permite que ele utilize os recursos de segurança principais, e que os fabricantes desenvolvam *drivers* de *hardware* para um kernel conhecido.
   1. Abordagem Nativa Multiplataforma: Xamarim

O Xamarim é uma plataforma de desenvolvimento móvel, que assim como aplicações híbridas com HTML 5 embarcado, visa criar aplicações utilizando uma única linguagem de programação, o C# (*C-Sharp*).

Com Xamarin é possível desenvolver apps móveis nativas utilizando C# (ou F#). De quebra, além de simplesmente poder escrever código utilizando C#, é possível utilizar features do C# e do .NET no desenvolvimento destes aplicativos, coisas como async/await e lambdas por exemplo. (QUAIATO, 2016, p.10)

Quaiato (2016, p.10) aponta, que ainda que as aplicações desenvolvidas utilizando o Xamarim, não utilizem das linguagens específicas de cada plataforma, elas são nativas. Isso porque o Xamarim utiliza o C# para fazer chamadas nativas do sistema operacional. O Xamarim não possui uma UI (*User Interface*) exclusiva, que emula a interface de aplicações nativas, portanto, todas as características do sistema operacional, estarão incorporadas em aplicações Xamarim

Tendo em vista que a UI de aplicações Xamarim, não é igual para cada plataforma, pode-se dizer que o código desse tipo de aplicação, não é 100% reaproveitado. De acordo com Machado (2016), o core de aplicações Xamarim é compartilhado, mas é necessário aprender conceitos relacionados a cada plataforma, como a configuração de arquivos de layout e recursos. Isso pode ser visto na Figura 4.

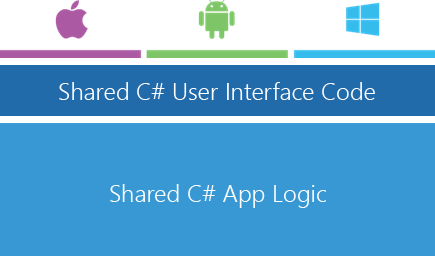
1. Desenvolvimento Tradicional - Xamarim

FONTE: (MACHADO, 2016)

* + 1. Xamarim.Forms

O Xamarim.Forms facilita a criação de aplicações fazendo uso de reaproveitamento de código de UI. A plataforma permite facilmente desenvolver interfaces que podem ser compartilhadas entre Android, iOS e Windows Phone.O desenvolvedor cria *views* e o Xamarim mapea para cada componente da plataforma. (NICOLLI, 2017, p.4). Isso pode ser visto na Figura 5.

1. Xamarim.Forms: UI Compartilhada e controles nativos



FONTE: NICOLLI (2017)

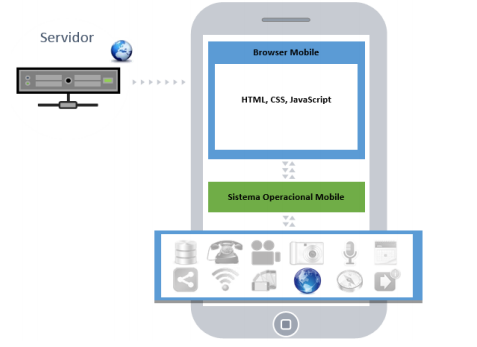
* 1. Mobile Web APPs

Aplicações *Web* ou *Web Apps*, assim como páginas *web*, são desenvolvidas com linguagens e tecnologias suportadas pelo navegador do dispositivo como HTML, CSS e *JavaScript*. Segundo Stark (2010, p.1, apud CAFÉ, 2012, p.9) o fato de serem acessadas pelo navegador através de uma URL, faz-se uma das maiores características de aplicações *web*, pois não utilizam linguagem nativa do dispositivo e não são instaladaas. Diferentemente de sites responsivos, onde as páginas e conteúdo são adaptáveis a telas de menor proporção, as web *apps* são projetados especificamente para dispositivos móveis.

Segundo Bergantin (2014, p.25), as Web App não têm acesso aos recursos nativos do dispositivo, o que faz com que a experiência do usuário seja inferior à de aplicações nativas ou híbridas.

A Figura 6, ilustra a estrutura de uma Web APP, e os recursos disponíveis a ela.

1. Estrutura e recursos de uma Web APP

.

FONTE: (BERGANTIN, 2014)

* 1. Abordagem Híbrida

O mercado de aplicações móveis cresce cada vez mais rápido, e a partir da necessidade de ter aplicações lançadas de forma acelerada e para diferentes plataformas, surgiu uma nova abordagem de desenvolvimento, uma nova “espécie” de aplicativos, a abordagem híbrida.

Aplicações híbridas, são uma junção de *Web APPs*, e aplicações nativas, que são executadas juntas, mantendo o acesso a recursos do dispositivo. Segundo Felix (2015) a abordagem híbrida de desenvolvimento consiste em desenvolver uma única aplicação que se mantem compatível em todas as plataformas, utilizando a mesma linguagem de desenvolvimento, SDK e IDE. Por esse motivo, tem um menor custo de desenvolvimento, e menor tempo de lançamento no mercado. As aplicações híbridas, assim como as Web APPs são desenvolvidas utilizando linguagens e tecnologias web, como HTML5, *Javascript*/*Typescript*, Angular 2 e CSS3, e utilizam ferramentas como o Cordova, para construir uma ponte entre tecnologias nativas (câmera, GPS).

Sendo assim, aplicações híbridas utilizam *frameworks* de componentes ou *wrappers* como o Ionic, que através do Cordova, convertem-na em um aplicativo instalável.

Segundo Poulsen (2012, p. 27 apud Café, 2012, p.10) assim como aplicações nativas, as híbridos também podem ser instalados nas lojas de aplicativos oficias, como a *App Store* da APPLE, e não requerem conexão ativa com a *internet* para execução, além disso são de fácil manutenção e atualização.

A Figura 7, mostra a estrutura de uma aplicação híbrida, além dos recursos disponíveis para esse tipo de aplicação.

1. Estrutura e recursos de uma aplicação híbrida

FONTE: (BERGANTIN, 2014)

Aplicativos híbridos são exibidos em um grande *container* chamado *webview*, que segundo Bristowe (2015), funciona como uma janela do navegador, sem croma (*interface*), que é configurada para rodar em tela cheia, ou seja funciona como um *browser* dentro do aplicativo.

A seguir, algumas tecnologias utilizadas no desenvolvimento de aplicações híbridas são listadas.

* + 1. Apache Cordova

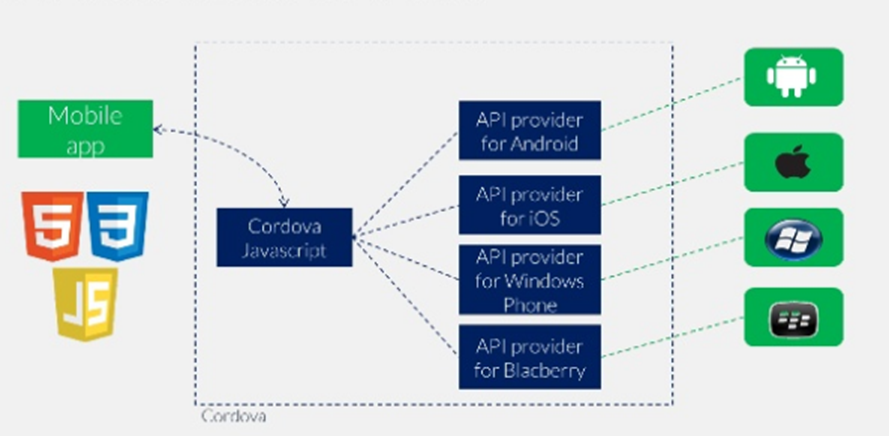
Antes de se chamar “Cordova”, e sendo desenvolvido pela empresa “Nitoboi”, a plataforma se chamava “PhoneGap”, mas foi comprada pela Apple em 2001, que então doou o código fonte para a Apache *Software Foundation*, chamando assim de Apache Cordova.

Segundo Lino (2015) o Apache Cordova é uma plataforma de desenvolvimento móvel com APIs, que permite que aplicações não nativas tenham acesso a recursos nativos do dispositivo, como a câmera, calendário e agenda.

O desenvolvimento de aplicações a partir do Cordova, pode ser feito em conjunto com *frameworks* como Ionic e o Sencha Touch, permitindo assim, copiar a usabilidade de dispositivos nativos, segmentando para cada plataforma já que o Cordova sozinho, permite apenas executar o aplicativo como nativo.

Na Figura 8, o funcionamento do Cordova é ilustrado, mostrando seu relacionamento com a aplicação e API’s.

1. Funcionamento do Cordova



FONTE: (MALAVOLTA, 2016)

* + - 1. Build

Segundo Vasconcellos (2017), é a partir do *build* do Cordova, que o código não nativo (CSS, JS, HTML) é empacotado dentro de uma aplicação nativa. Ele pode ser considerado uma das funcionalidades essenciais do Cordova, dado que o *input* são os arquivos de tecnologia *web*, e o *output* a aplicação nativa, que ao ser executada, abre a *webview* que irá renderizar a aplicação.

* + - 1. Plug-ins

De acordo com a Apache *Software Foundation* (2015) os *plug-ins* do Cordova, fornecem uma *interface* para que ele consiga se comunicar com componentes nativos e APIs padrões do dispositivo. O conjunto de *plug-ins* do Apache Cordova, chama-se *Core Plug-ins,* e são eles que acessam recursos como câmera e contatos, no entanto, ao iniciar um projeto Cordova nenhum *plug-in* está incluído para que o build final da aplicação não fique sobrecarregado ou seja, todos os *plug-ins* de interesse do desenvolvedor devem ser explicitamente adicionados.

* + 1. Angular 2

De acordo com Lacerda (2014), Angular é um *framework* *open source* para desenvolver aplicações *web* e móveis, mantido pela Google. O Angular permite estender a sintaxe do HTML, deixando páginas *web* com conteúdo dinâmico, ou seja, aumenta a DOM do navegador, fazendo com que aja uma maior componentização dos elementos.

Em 14 de setembro de 2016, o Angular 2 foi lançado, esse então mais rápido, mais escalável e mais moderno. Segundo Rangle (2016) o Angular 2 foi pensado para celulares desde o início. As *views* e *controllers* são substituídos por componentes, e a linguagem de escrita se tornou o *typescript*, um superset do *javascript*, implementado sobre o ES6.

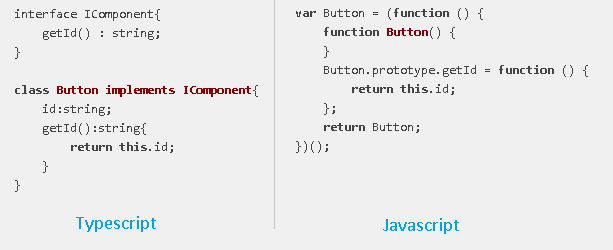
* + - 1. Typescript

De acordo com Dionisio (2016), o Javascript vem expandindo seu uso para além do desenvolvimento *front-end,* hoje é possível utiliza-lo para desenvolver aplicações móveis, internet das coisas, e *n*o *back-end* de aplicações*.* A partir disso, o Typescript surgiu como um *superset* do Javascript, oferecendo tipagem, classes e interfaces (opcionais).

O TypeScript é uma linguagem de programação criada e também mantida pela Microsoft, desenhada por Anders Hejlsberg (o designer do C#), que segue as especificações do ECMAScript 6. Com ele, é possível fazer uso dos principais recursos que toleram o uso de orientação a objetos, e princípios como: herança, abstração, polimorfismo e encapsulamento.

As *interfaces* Typescript permitem declarar tipos personalizados, para serem utilizados na aplicação, e ajudam a evitar erros em tempo de compilação ao utilizar objetos de tipos errados.

De acordo com Rathore (2016) o Typescript gera Javascript, ou seja todo código é ao final transformado em Javascript puro, fazendo com o que o *browser* o compreenda. Na Figura 9, pode-se ver uma uma *interface* e uma classe criada em Typescript, e depois o código transformado em Javascript.

1. Diferenças do código Typescript e Javascript.

FONTE: (SCHMITZ, 2015)

* + 1. Ionic Framework 2

Ionic é um *framework open source,* utilizado para desenvolvimento de aplicativos híbridos, baseado no Cordova e no Angular 2, que visa desempenho, *design* e otimização. O *Framework* permite ao desenvolvedor utilizar tecnologias já conhecidas como HTML, CSS (com SAAS) e *JavaScript* (através do *TypeScript*) para criação de aplicativos híbridos. “Em uma definição simples, podemos dizer que o Ionic é um conjunto de componentes *front-end* que permite escrever código HTML, CSS e *JavaScript* que se assemelha a uma aplicação nativa. ” (Machado, p.1, 2017)

O Ionic *Framework* foi criado pela empresa Drifty CO. em 2013. A Drifty CO, notou uma série de questões que precisavam ser melhoradas no *framework*, e então lançou em 2016, o Ionic 2, esse desenvolvido com base no Angular 2, que facilitou a construção de aplicações híbridas com Ionic.

* + - 1. Desenvolvimento

Igualmente ao Android, iniciar o desenvolvimento de aplicações com o Ionic *Framework*, não necessita de equipamentos extras, apenas um computador com qualquer sistema operacional, exceto para *APPs* iOS, que ainda necessitam de um MAC SO, com XCode. Não há nenhuma IDE recomendada na documentação do Ionic 2. A linguagem utilizada é o *Javascript* (com *TypeScript*), além de HTML5 e CSS (SCSS), para *front end*. Por ser baseado no Angular, os principais componentes de uma aplicação, têm escopo isolado, “[...] Cada “página” de um projeto tem seu próprio *template* visual (HTML), estilo (SCSS) e classe (ts). ” (CABRAL, 2016). O Ionic 2 foi desenvolvido com o conceito de temas, e como padrão, segundo Carvalho (2017), ele já segue as *guidelines* de *design* do iOS, do Material Design e do Windows Phone. O SCSS, na versão 2, passou a estar mais estruturado, facilitando a personalização de componentes.

* 1. Conclusão do Capítulo

Atualmente, muitos são os sistemas operacionais para dispositivos móveis existentes, os quais ocupam diferentes grupos de mercado e consequentemente diferentes métodos de desenvolvimento. Como descrito nesse capítulo, o sistema operacional mais usado nos dias atuais é o Android, seguido pelo IOS e o Windows Phone. Atingir determinada fatia de mercado específica, pode ser uma boa escolha, mas também pode ser inviável em alguns casos, e por isso, existem diferentes abordagens de desenvolvimento. Tanto a abordagem híbrida quanto a abordagem nativa possuem vantagens e desvantagens muito claras, que podem influenciar diretamente no tipo de aplicação a ser desenvolvida.

Antes de iniciar o desenvolvimento de uma aplicação móvel, é necessário saber tudo o que ela irá utilizar, o público que irá atingir, e qual (ou ainda quais) sistema operacional ela abrangerá. Planejar os detalhes da aplicação, faz com que a escolha a ser tomada seja a mais certa possível.

Quando se opta por desenvolver uma aplicação nativa, é necessário o conhecimento do sistema operacional escolhido, linguagem utilizada por ela e ambiente de desenvolvimento, por exemplo. Ao escolher diferentes sistemas operacionais para desenvolvimento, é necessário o conhecimento de cada um, as peculiaridades de cada um. Por outro lado, ao desenvolver uma aplicação híbrida, as tecnologias de cada plataforma não necessitam ser fortemente investigadas, a linguagem e o ambiente de desenvolvimento são um só, mas as características de cada plataforma nem sempre são respeitadas.

Toda a abordagem de desenvolvimento necessita de um bom tempo para ser desenvolvida, todas necessitam de estudo, e isso está diretamente ligado diretamente ao conhecimento/*expertise* de quem irá desenvolver.

1. Benchmark

De acordo com Buora (2016), o termo “*benchmark*” nasceu no universo coorporativo, mas migrou e vem se popularizando gradualmente no meio da computação. Na computação então tem o mesmo sentido original, que é o de comparar duas coisas distintas, tendo como base testes, variáveis e padrões semelhantes.

Segundo Gray (1993, apud FERREIRA, 2012, p.6), um *benchmark* deve atender os respectivos critérios:

* Relevância: As variáveis comparadas devem descrever a funcionalidade e expectativa de desempenho acerca do produto submetido ao *benchmark*.
* Portabilidade: Não deve haver privilégio na utilização de contextos, que sejam reservados a uma determinada abordagem.
* Escalabilidade: As variáveis comparadas devem atender a pequenos ou grandes.
* Simplicidade: As variáveis definidas devem ser simples e de fácil entendimento. Métricas muito complexas podem perder credibilidade na comunidade interessada.

Dessa forma, este capítulo tem como objetivo, elencar as variáveis utilizadas para comparação das abordagens de desenvolvimento híbrida e nativa.

* 1. Parâmetros de Comparação

Para realização complementar da comparação entre o desenvolvimento de aplicações móveis híbridas e nativas, seguindo as características de um *benchmark* citadas acima, foram levantados alguns parâmetros, listados a seguir.

* + 1. Esforço de Desenvolvimento

Sabe-se, que desenvolver qualquer produto digital é complexo, e que saber previamente o esforço de desenvolvimento é essencial para empresas ou desenvolvedores realizarem o planejamento que acarretará no sucesso do produto. A partir disso, esse parâmetro visa auxiliar no planejamento de aplicações, avaliando custo de desenvolvimento, documentação e comunidade das abordagens e recursos extras.

* + - 1. Custo

Esse parâmetro visa a avaliação indireta ligada aos custos de desenvolvimento de cada plataforma, o que segundo Leite (2016), auxilia no planejamento e tomada de decisão em projetos de *software*.

* + - 1. Documentação complementar/Comunidade

De acordo com Nascimento (2010), quando se inicia o desenvolvimento de alguma aplicação ou produto de *software*, é necessário entender os comandos e particularidades necessárias para o seu funcionamento, e por isso, a documentação auxilia o desenvolvedor a utilizar e manipular essas informações. Além disso, a comunidade a cerca de uma abordagem de desenvolvimento, ou seja, o número de adeptos contribuintes, pode fornecer soluções rápidas para problemas no decorrer do desenvolvimento. Esse parâmetro então, avaliará de forma indireta a documentação complementar e comunidade de cada abordagem.

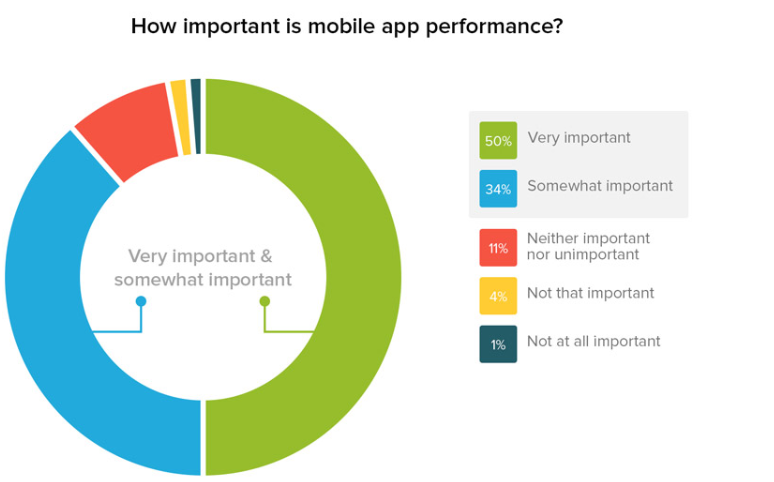
* + - 1. Recursos

Na elaboração de aplicações, recursos ou ferramentas de apoio extra, auxiliam o desenvolvimento. Esses recursos podem ser desde emuladores, ferramentas de prototipação ou conjunto de elementos de *interface*. Por esse motivo, esse parâmetro avaliará os recursos extras ao desenvolvimento, fornecidos por cada abordagem.

* + 1. Desempenho

Desempenho, pode ser descrito como a capacidade de comportamento correto de um *software*. De acordo com a Forrester (2015), ele é o primeiro atributo mais valorizado na avaliação de aplicativos móveis, e por isso, será avaliado de forma direta no desenvolvimento deste estudo.

A Figura 10 ilustra a importância do desempenho em aplicações móveis. Como ilustrado, é possível notar que 50% dos usuários entrevistados classificaram o parâmetro desempenho como “muito importante”.

1. Importância do desempenho em aplicações móveis

|  |
| --- |
|  |

FONTE: (ABED, 2016)

Os parâmetros analisados serão “tempo de resposta” e “consumo de bateria” descritos a seguir.

* + - 1. Tempo de Resposta

Atualmente, sabe-se a importância do conforto do usuário ao utilizar aplicativos móveis, e uma das formas de transmissão desse sentimento é através do tempo de resposta das ações. Segundo a OPUS Software (2016), ainda que uma aplicação resolva um problema real do usuário, se o tempo de respostas das ações for alto, o usuário tenderá a avalia-la negativamente.

* + - 1. Consumo de Bateria

A partir da necessidade de se manter sempre conectado, o uso da bateria em aplicações para dispositivos móveis é de suma importância. Ao desenvolver aplicativos que necessitam de alto processamento, é natural que o uso de bateria seja maior, portanto esse parâmetro será avaliado, pois a escolha da abordagem que otimize o uso de bateria pode ser importante.

* + 1. Interface

Uma *interface* que não foi adequadamente projetada a plataforma que irá atender, pode fazer com o que o aplicativo tenha desvantagem em relação aos concorrentes no mercado. De acordo com Abed (2016), a psicologia de um usuário de aplicativo móvel é direta, eles já aprenderem a utilizar seu dispositivo e não querem ter que absorver novos recursos de *interface*, portanto o estilo deve estar integrado a plataforma. Por isso, esse parâmetro analisará de forma macro, aspectos como a facilidade de customizar a interface de acordo com cada abordagem.

1. Aplicações Experimentais

Para fomentar o estudo avaliativo de forma imparcial, o desenvolvimento de duas aplicações móveis será realizado, uma utilizando a abordagem nativa e outra utilizando a abordagem híbrida, afim de obter na prática, as peculiaridades de cada plataforma. Sendo assim, esse capítulo elencará os passos para desenvolvimento das aplicações experimentais.

* 1. Exemplo de Uso

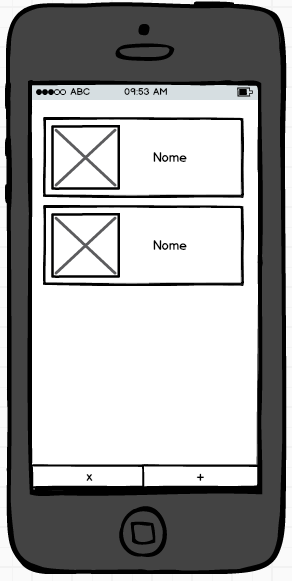
O exemplo de aplicativo a ser desenvolvido utilizando a abordagem híbrida e nativa será denominado “HelpRunning”, um aplicativo para cadastro de corridas. Por se tratar de um aplicativo exemplo, apenas para avaliação prática das abordagens apresentadas neste estudo, o aplicativo não tem cunho de negócio, e por isso apresentará funcionalidades simples.

O aplicativo utilizará recursos nativos, como:

* Acesso a câmera: esse recurso será utilizado para a tirar uma foto instantânea da corrida
* Acesso a galeria de fotos: seguindo o mesmo propósito do recurso acima, a galeria de fotos permitirá adicionar fotos já salvas no dispositivo.
* Cronômetro: esse recurso permitirá cronometrar o tempo de cada corrida cadastrada
  + 1. Prototipação de Telas

De acordo com Camarini (2013), a prototipação, é um método que tem como finalidade facilitar o entendimento, e alinhar requisitos ou funções de um determinado software. Nesse exemplo de uso, a prototipação será usada para nortear o desenvolvimento das aplicações nas diferentes abordagens.

A seguir, as telas dos aplicativos serão descritas, iniciando na Figura 11.

1. Telas de login e visualização dos aplicativos

FONTE: (PRÓPRIA AUTORA, 2017)

A tela de login será responsável por fazer a autenticação do usuário, será necessário a adição de e-mail e senha para entrar. Para a validação dessa pesquisa, a tela de login foi desenvolvida para comparação de layout e dificuldades, a autenticação real será desenvolvida futuramente.

A tela de visualização será responsável por exibir dados salvos da corrida, com foto do local, nome e data.

1. Tela de cadastro de corridas

FONTE: (PRÓPRIA AUTORA, 2017)

A tela de cadastro de corrida, obterá o nome, local e data da corrida, além de permitir adicionar uma foto, tanto capturada na hora, quanto salva na galeria de imagens, recursos nativos do dispositivo.

* 1. Tecnologias Utilizadas e Ambiente de Desenvolvimento

Nessa sessão, as tecnologias e o ambiente de desenvolvimento utilizados para a criação do aplicativo RunningHelper serão descritos. A aplicação foi desenvolvida, pela autora dessa pesquisa, com conhecimento intermediário em “codificação” de aplicações móveis, e avançado em qualidade de aplicações móveis.

Como evidenciado no capítulo 2 desse estudo, o exemplo de uso será desenvolvido de forma nativa utilizando o Google Android, e de forma híbrida utilizando o Ionic *Framework* 2.

As características do ambiente de desenvolvimento, serão listadas a seguir:

* Sistema Operacional: Computador 1 – Windows 8.1 Pro; Computador 2 – Windows 10
* Processadores: Computador 1 – Intel Core i3-2375M; Computador 2 – Intel Core i5 –
* Memória RAM: Computador 1 – 4GB; Computador 2 – 8GB.

As ferramentas de desenvolvimento das aplicações:

* Linguagem de programação: APP Google Android – Java; APP Ionic – Typescript, HTML5, CSS.
* Ambiente de desenvolvimento: APP Google Android – Android Studio 1.0; APP Ionic – Microsoft Visual Studio Code 1.7.1.
* Banco de Dados: SQL Lite
* Ferramentas auxiliares: Cmder.

Ambiente de teste:

* Sistema Operacional: Android Lollipop 5.0.
  1. Descrição do Desenvolvimento

Durante essa seção, será descrito o desenvolvimento dos protótipos híbrido e nativo citados nessa pesquisa, e descritos na sessão 4.1.

* + 1. Criando o projeto

As duas abordagens de desenvolvimento, possuem características próprias para a criação de projetos móveis, as particularidades serão descritas a seguir:

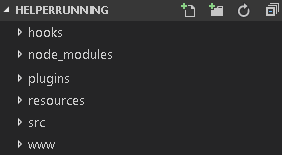
* + - 1. Criando o projeto: Ionic Framework 2

Para a criação do projeto HelperRunning utilizando o Ionic Framework 2, foi necessário a instalação do Node.Js, uma plataforma para desenvolvimento de aplicações server-side baseadas em rede (Lopes,2014). Após a instalação do Node.Js, seguiu-se instalando o Ionic Cordova. Todas as instalações, atualizações etc, no projeto no híbrido, foram executadas através de linha de comando. Para iniciar o projeto, o seguinte comando foi utilizado:

ionic start HelperRunning blank -–v2.

Esse comando, inicia um projeto, com nome “HelperRunning”, em branco, a partir da versão 2, do Ionic. Após isso, a estrutura do projeto é criada, conforme visto na Figura 13.

1. Estrutura do Projeto Ionic 2



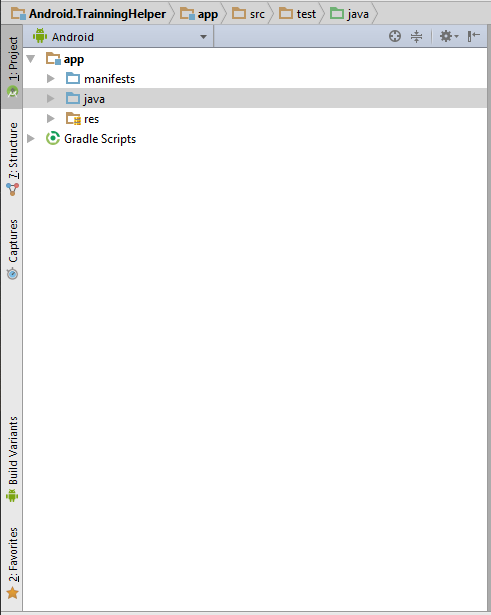
FONTE: (PRÓPRIA AUTORA,2017)

* + - 1. Criando projeto: Android Nativo

Para a criação da aplicação Android, através do Android Studio, é necessário a manipulação da própria interface do Android Studio. Para a criação do projeto, foi preciso acessar “New Project”, adicionar o nome do projeto “HelperRunning”, escolher a versão mínima de SDK ao qual o projeto suporta, e optar pela opção Blank Activity.

Após isso, a estrutura é criada, conforme visto na Figura 14.

1. Estrutura do projeto Android



FONTE: (PRÓPRIA AUTORA, 2017)

* + 1. Tela de Login

O desenvolvimento da tela de login (layout e chamadas de tela), nas duas abordagens foi consideravelmente simples. O layout desenvolvido em programas de editoração de imagens foi facilmente aplicado, e o design das duas aplicações ficou extremamente semelhante, conforme visto na Figura 15.

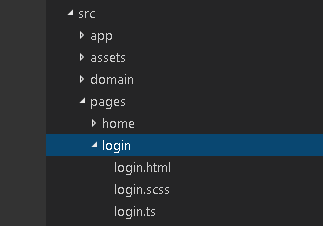
1. Tela de login – Nativo e Híbrido

|  |  |
| --- | --- |
| **ANDROID - NATIVO** | **ANDROID – HÍBRIDO/IONIC** |

FONTE: (PRÓPRIA AUTORA,2017)

* + - 1. Tela de Login: Híbrido

Para a criação da tela de login do protótipo híbrido, nenhuma particularidade foi necessária, o arquivo foi facilmente criado dentro da pasta “src”, onde todos os arquivos fonte do projeto estão localizados. Dentro da pasta “src”, há um diretório chamado “pages”, onde todas as páginas da aplicação ficam armazenadas, para a página de login, por exemplo, foi criada uma pasta específica, conforme visto na Figura 16.

1. Diretório de armazenamento da página de login – protótipo híbrido

FONTE: (PRÓPRIA AUTORA, 2017)

Dentro da pasta pages/login, estão todos os arquivos manipuláveis da página, como o arquivo HTML, o arquivo de estilo, e o arquivo Typescript.

As páginas podem ser geradas automaticamente com ionic, através do comando:

$ ionic generate page nome.

No entanto, apenas a pagina vazia é criada, para a adição de inputs, botões e afins, fez-se o uso de *UI components*, disponibilizados pelo Ionic Framework.

* + - 1. Tela de login: Android Nativo

A criação básica da tela de login no protótipo nativo, foi bastante simples. O Android Studio, fornece uma série de templates para nortear o desenvolvimento das telas. Ao criar o projeto, o template “Blank” foi escolhido, mas ao desenvolver a tela de login, foi utilizado um template disponibilizado pela ferramenta, o “Login Activity”, que disponibiliza uma tela simples, com inputs de e-mail, senha e botão entrar.

A customização da tela, foi razoavelmente fácil, o layout previamente desenvolvido foi aplicado corretamente a tela, sem grandes problemas.

A estrutura de pastas de desenvolvimento do protótipo nativo, não é semelhante ao protótipo híbrido, conforme podemos ver na Figura 17.

1. Resultados preliminares

O desenvolvimento de aplicações móveis em mais de uma plataforma é bastante popular nos dias de hoje. Ao criar um código fonte que alimenta um aplicativo nessas duas plataformas, uma empresa economiza tempo e dinheiro. Também foi possível perceber previamente, que os *frameworks* para desenvolvimento de aplicações híbridas têm evoluído de forma significativa no mercado, apesar de ainda não apresentarem uma singularidade tão grande quanto a de aplicações nativas.

O Quadro 1, apresenta de forma resumida, as vantagens e desvantagens de cada abordagem levantadas durante a execução do trabalho.

1. Vantagens e desvantagens de aplicações nativas e híbridas

|  |  |  |
| --- | --- | --- |
|  | Aplicações Nativas | Aplicações Híbridas |
| Vantagens | * Melhor experiência de usuário. * Acesso a todas as API’s nativas. * Melhor desempenho. * Têm a maior fatia de desenvolvimento no mercado. | * 1 aplicação abrange N plataformas. * Baixo custo de desenvolvimento. * Tempo de lançamento ao mercado é menor. * Demanda conhecimento de apenas uma linguagem e ambiente de desenvolvimento. * Fácil atualização. |
| Desvantagens | * Necessita de 1 aplicativo por plataforma. * Custo de desenvolvimento e manutenção elevados. * Necessita conhecimento de linguagem e ambiente para cada plataforma. * Tempo elevado de desenvolvimento. | * Acesso restrito a recursos nativos. * Experiência de usuário reduzida em comparação a aplicações nativas. * Desenvolvimento de aplicativos híbridos ainda é novo e ainda não existe tanto suporte quanto aplicações nativas. * A falta de recursos de UI puramente nativos do iOS ou do Android pode resultar em um desempenho mais lento do aplicativo. |

FONTE: (PRÓPRIA AUTORA, 2017)

* + - * 1. Referências Bibliográficas

ABED, Robbie. **Hybrid vs Native Mobile Apps – The Answer is Clear**. Disponível em: <https://ymedialabs.com/hybrid-vs-native-mobile-apps-the-answer-is-clear/> Acesso em 16 de abril de 2017

ANDROID. **Platform Architecture**. Disponível em: <https://developer.android.com/guide/platform/index.html>. Acesso em 15 de abril de 2017

ARRUDA, Saulo. **Design responsivo, Desenvolvimento Nativo e App Híbrido, quando usar?** Disponível em: <http://jera.com.br/blog/4931/design-ux/design-responsivo-o-que-e> Acesso em 16 de abril de 2017.

BERGANTIN, Carlos Eduardo. **Análise de boas práticas para o desenvolvimento de web wapps**. Disponível em: <http://aberto.univem.edu.br/bitstream/handle/11077/994/Carlos%20Eduardo%20Martinelli.pdf?sequence=1> Acesso em 22 de Maio de 2017.

BRISTOWE, John. **What is a Hybrid Mobile App?** Disponível em: <http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/> Acesso em 23 de Maio de 2017.

BUORA, Helito. **O que é benchmark? Entenda como funciona o teste de desempenho do celular.** Disponível em: <http://www.techtudo.com.br/noticias/noticia/2016/06/o-que-e-benchmark-entenda-como-funciona-o-teste-de-desempenho-do-celular.html> Acesso em 11 de Junho de 2017.

CABRAL, Carlos. **Criando uma aplicação móvel com Ionic 2 e Angular 2 em dez passos**. Disponível em: <https://tableless.com.br/criando-uma-aplicacao-movel-com-ionic-2-e-angular-2-em-dez-passos/>. Acesso em 24 de Maio de 2017.

CAFÉ, Adriel Almeida. **Desenvolvimento de Cross-Platform Mobile Apss utilizando Titanium Mobile**. Disponível em: <http://adrielcafe.com/images/adrielcafe/artigos/2012/12/tcc/adriel-tcc.pdf>. Acesso em 22 de Maio de 2017.

CAMARINI, Bruno. **Prototipação E Sua Importância No Desenvolvimento De Software.** Disponível em: <http://dextra.com.br/blog/prototipacao-e-sua-importancia-no-desenvolvimento-de-software/> Acesso em 16 de junho de 2017

CAPELAS, Bruno. **Brasil chega a 168 milhões de smartphones em uso.** Disponível em: <http://link.estadao.com.br/noticias/gadget,brasil-chega-a-168-milhoes-de-smartphones-em-uso,10000047873>. Acesso em 20 de Setembro de 2016.

CARVALHO, Jean Lucas de. **A introdução que faltava ao Ionic 2**. Disponível em: < http://frontinbrazil.com.br/ionic-framework-2/> Acesso em 28 de Maio de 2017.

DELFINO, Pedro. Razões para desenvolver para Android. Disponível em: <http://e-tinet.com/android/razoes-desenvolver-android/> Acesso em 08 de Abril de 2017.

DIONISIO, Edson. **Introdução ao TypeScript.** 2016. Disponível em: < http://www.devmedia.com.br/introducao-ao-typescript/36729>. Acesso em 06 de Agosto de 2017.

FELIX, Waldyr. **6 aspectos essenciais para decidir entre aplicações mobile híbridas e nativas**. Disponível em: <https://medium.com/@waldyrfelix/6-aspectos-essenciais-para-decidir-entre-aplica%C3%A7%C3%B5es-mobile-h%C3%ADbridas-e-nativas-51bce0dace68> Acesso em 14 de maio de 2017.

FLORENZANO, Cláudio. **Porque você deveria desenvolver aplicativos Android.** Disponível em: <http://www.cbsi.net.br/2015/11/porque-voce-deveria-desenvolver-aplicativos-android.html> Acesso em 08 de abril de 2017.

FORRESTER. **Why Good Apps Are Not Good Enough**. Disponível em: <http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=WH&infotype=SA&htmlfid=KUL12363USEN&attachment=KUL12363USEN.PDF> Acesso em 14 de junho de 2017.

GADELHA, Reginaldo. **Desenvolvendo para Android: Arquitetura Android.** Disponível em: <http://www.tiselvagem.com.br/geral/desenvolvendo-para-android-arquitetura-android/>. Acesso em 15 de Abril de 2017.

KARCH, Marziah. **What Are Apps?** - Definition and Examples. Disponível em: <https://www.lifewire.com/what-are-apps-1616114>. Acesso em 12 de Março de 2016.

LACERDA, Raphael. **Conheça o Angular.** Disponível em <http://blog.caelum.com.br/como-anda-o-angular-js-devo-embarcar-nessa/> Acesso em 12 de Junho de 2017.

LEITE, Ludimila. **A importância da documentação em software para usuários iniciantes**. Disponível em < https://otextolivre.wordpress.com/2010/06/02/a-importancia-da-documentacao-em-software-para-usuarios-iniciantes/>. Acesso em 15 de Junho de 2017.

LIMA, Victor **. Nativo x Híbrido – A Discussão Final! (Parte 1).** Disponível em: <http://blog.concretesolutions.com.br/2016/03/nativo-x-hibrido/> Acesso em 06 de Setembro de 2016.

LINO, Mariana. **O que é Apache Cordova?** Disponível em: <http://blog.marianalino.com.br/o-que-e-apache-cordova/> Acesso em: 24 de Maio de 2017.

LOPES, Cosme. **O que é Node.js e saiba os primeiros passos.** Disponível em: <https://tableless.com.br/o-que-nodejs-primeiros-passos-com-node-js/> Acesso em 08 de Setembro de 2017.

MACHADO, Henrique. **Xamarin, Ionic e Cordova: Conheça o que são e as principais diferenças.** Disponível em: <http://www.devmedia.com.br/xamarin-ionic-e-cordova-conheca-o-que-sao-e-as-principais-diferencas/37690> Acesso em 24 de Maio de 2017.

MALAVOLTA, Ivano. **Apache Cordova**. Disponível em: <https://www.slideshare.net/iivanoo/20152016-apache-cordova> cesso em 24 de Maio de 2017

MEYER, Maximiliano. **A história do Android.** Disponível em<https://www.oficinadanet.com.br/post/13939-a-historia-do-android> Acesso em 25 de março de 2017

NICOLLI, Letticia. **Entendendo o Xamarim.Forms.** 2017. Disponível em: < https://www.lambda3.com.br/2017/04/entendendo-xamarin-forms/> Acesso em 13 de Agosto de 2017.

PISULAK, Pisulak. **The art of Building Bridges for Android Hybrid Apps.** Disponível em: <https://www.slideshare.net/kropek88/the-art-of-building-hybrid-bridges>. Acesso em 23 de Maio de 2017.

QUAIATO, Vinicius. **O que é Xamarim?.** 2016. Disponível em: <https://www.lambda3.com.br/2016/10/o-que-e-xamarin/> Acesso em 13 de Agosto de 2017.

RATHORE, Deven. **Typescript Vs Javascript – Why Typescript Is Next To Big Thing.** 2016.Disponível em:<https://www.dunebook.com/typescript-vs-javascript-why-typescript-is-next-to-big-thing/>. Acesso em: 06 de Agosto de 2017.

RIVA, Fernando De La. **MobileFirst: nossa concepção de desenvolvimento mobile (Parte 1).** Disponível em: <http://blog.concretesolutions.com.br/2014/11/mobilefirst-desenvolvimento-mobile/>. Acesso em: 24 de Setembro de 2016.

SCHIMITZ, Daniel. **Diga olá ao TypeScript e adeus ao JavaScript.** 2015. Disponível em: <https://tableless.com.br/diga-ola-ao-typescript-e-adeus-ao-javascript/>. Acesso em 06 de Agosto de 2017.

THE APACHE SOFTWARE FOUNDATION. **Overview.** Disponível em: < https://cordova.apache.org/docs/en/latest/guide/overview/> Acesso em 24 de Maio de 2017.

* + - * 1. Bibliografia complementar

CRISTOPH, Rehmann. **Cross Platform Development mit Xamarin.** Disponível em: <https://blog.noser.com/tag/windows-phone-8/>. Acesso em 15 de Abril de 2017.

SANTOS, Guilherme. **Desenvolvimento híbrido x desenvolvimento nativo de aplicações para celular**. Disponível em: <http://nextecommerce.com.br/desenvolvimento-hibrido-x-desenvolvimento-nativo-de-aplicacoes-para-celular/>. Acesso em 07 de Setembro de 2016.

SILVA, Leandro et.al. **Desenvolvimento de Aplicações para Dispositivos Móveis: Tipos e Exemplo de Aplicação na plataforma iOS.** II Workshop de Iniciação Cientíﬁca em Sistemas de Informação. Goiânia- GO, p.25-26, maio/2015.

OPPUS, Software. **Como é Feita a Avaliação de Aplicativos Móveis?** Disponível em: <https://www.opus-software.com.br/avaliacao-de-aplicativos-moveis/> Acesso em 14 de junho de 2017.