Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

BRUNO CAETANO SIMINES

ABORDAGEM PARA DESENVOLVIMENTO HÍBRIDO DE APLICAÇÕES MÓVEIS COM PLATAFORMAS IONIC E APACHE CORDOVA

Assis/SP
2017

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

BRUNO CAETANO SIMINES

ABORDAGEM PARA DESENVOLVIMENTO HÍBRIDO DE APLICAÇÕES MÓVEIS COM PLATAFORMAS IONIC E APACHE CORDOVA

Orientador: Prof. MSc. Guilherme de Cleva Farto

Nota do orientador:	Nota do avaliador:

Assis/SP
2017

“Não importa o quanto à vida possa ser ruim, sempre existe algo que você pode fazer, e triunfar. Enquanto há vida, há esperança.”

Stephen Hawking.

RESUMO

Nos últimos anos os dispositivos móveis tem se destacado no mercado, com o aumento do uso destes aparelhos, diversos usuários passam horas procurando e testando novos aplicativos que facilitam o seu dia-a-dia.

Desenvolvedores entram neste mercado visando abocanhar uma parte dos lucros, porém encontram algumas dificuldades com os códigos nativos de cada sistema operacional, o surgimento do desenvolvimento híbrido tem facilitado estes desenvolvedores.

As plataformas Ionic e o Apache Cordova têm tornado cada vez mais possível a criação de aplicativos robustos tanto quanto os nativos. A proposta deste trabalho foi desenvolver uma pesquisa sobre a plataforma Ionic e o Apache Cordova, sendo possível a criação de uma aplicação híbrida, mostrando os recursos providos destas ferramentas.

Palavras-chave: Dispositivos; Desenvolvimento Híbrido; Ionic; Apache Cordova;

ABSTRACT

In recent years mobile devices have stood out in the market, with the increase of the use of these devices, several users spend hours searching and testing new applications that facilitate their day to day.

Developers enter this market aiming to capture a part of the profits, But encounter some difficulties with the native codes of each operating system, the emergence of hybrid development has facilitated these developers.

The Ionic platform and Apache Cordova has made it increasingly possible to create robust applications as well as natives. The proposal of this work was to develop a research on the Ionic platform and Apache Cordova, being possible the creation of a hybrid application, showing the resources provided by these tools.

Key-words: Devices**,** Ionic, Apache Cordova, developmet hybrid.

LISTA DE ILUSTRAÇÕES

Figura 1 - Ábaco 13
Figura 2 - Eniac 14
Figura 3 – Camadas iOS. 21
Figura 4 – Plataforma XCode 22
Figura 5 – Interface Android 23
Figura 6 – Funcionamento de uma ViewGroup 24
Figura 7 – Divisões das camadas do Android 25
Figura 8 – Funcionamento de uma Activy 26
Figura 9 – Plataforma Eclipse com Android SDK 27
Figura 10 – Funcionamento do Cordova 29
Figura 11 – Aplicativo do National Museum of African History and Culture 33
Figura 12 – Aplicativo Ionic View 34
Figura 13 - Ações que usuário poderá exercer dentro da aplicação. 39
Figura 14 – Configuração da back-end da tela de login 40
Figura 15 – Provider do login 41
Figura 16 – Estrutura da aplicação, Parte 1 42
Figura 17 – Estrutura da Aplicação, Parte 2 43
Figura 18 – Tela de listagem das tarefas 44
Figura 19 – Tela de criação das tarefas 45
Figura 20 – Tela de configuração back-end da câmera, Parte 1 46
Figura 21 – Tela de configuração back-end da câmera, Parte 2 47
Figura 23 – Tela de configuração back-end da câmera, Parte 3 48
Figura 24 – Tela de configuração front-end da câmera. 49

SUMÁRIO

1 INTRODUÇÃO 9
1.1 OBJETIVOS 10
1.2 JUSTIFICATIVA 11
1.3 MOTIVAÇÃO 12
2 Computação Móvel. ..... **Erro! Indicador não definido.**
2.1 Contextualização. 13
2.2 Fundamentos da computação móvel. 15
2.3 Caracterização de um Dispositivo Móvel. 16
2.4 Aparecimento do Smartphone 17
2.5 Desenvolvimento Híbrido. 18
3 Plataforma de Desenvolvimento Móvel. 20
3.1.1 Fundamentos iOS 20
3.1.2 Arquitetura iOS. 20
3.1.3 Plataforma de Desenvolvimento iOS. 22
3.2 Plataforma Android. 23
3.2.1 Fundamentos do Android 23
3.2.2 Arquitetura Android 24
3.2.3 Plataforma de Desenvolvimento do Android 26
4 Plataforma Apache Cordova 28
4.1 Fundamentos da Plataforma 28
4.2 IONIC 30
4.3 Exemplos de Aplicações 32
4.4 Outros Recursos 34
4.4.1 HTML5 34
4.4.2 CSS3 35
5 PROPOSTA DE TRABALHO 37
5.1 Proposta de Aplicação 37
5. 2 Recursos da Aplicação 38
6 DESENVOLVIMENTO DO TRABALHO 40
6.1 Finalizando a Aplicação 51
7 CONCLUSÃO 51
REFERÊNCIAS 53

1 INTRODUÇÃO

Com o avanço tecnológico do setor da computação móvel e o seu aumento em vendas nos últimos anos, diversas empresas ganharam fama e mercado com aplicativos para estes pequenos dispositivos, se entende como dispositivo móvel, todo aquele equipamento que possui autonomia suficiente para que possa ser levado e usado em qualquer lugar, quanto maior foi o tempo de uso e menor o tempo de recarga da bateria, maior será a sua mobilidade, (PREZOTTO, BONIATI, 2014).

Alguns desenvolvedores acabam encontrando dificuldades ao ingressarem neste vasto mercado, por encontrarem problemas com o incompatibilidade de sistemas, como o caso de aplicações no Google Android e no Apple iOS, por possuírem pouco conhecimentos, sendo que são sistemas com códigos nativos totalmente diferenciados entre si, ocasionando um software diferente pra cada plataforma, precisando então de tempo e especialização em cada sistema, além disso a concorrência se torna desleal com empresas de grande porte.

Após o termino do desenvolvimento da aplicação nativa para o Apple iOS, o desenvolvedor irá enviar seu aplicativo à loja, que será submetido a um processo de verificação do código, para que não ocasione nenhum maleficio a nenhum dispositivo, esse processo pode levar tempo, e caso tenha alguma atualização futura, passará novamente por este processo (O'REILLY, 2014).

Com o desenvolvimento híbrido é possível criar aplicativos web, que não precisam estar disponíveis na loja virtual, utilizando a framework Ionic, o desenvolvedor ganha tempo, pois esta ferramenta auxilia com os recursos nativos de cada sistema operacional, devido a utilização de outra framework chamada Apache Cordova, que faz o back-end da aplicação, permitindo acesso à câmera, gps, e outros dispositivos (VASCONCELLOS, 2015).

Criando uma aplicação em HTML5, rodando em cima de uma webview é possível abrir em qualquer plataforma (LIMA, 2016), o que faz com que uma aplicação seja híbrida, pois o HTML5 é universal, e qualquer navegador moderno consegue interpreta-lo, com o CSS3 é possível dar um designer apreciável a aplicação, utilizando o Ionic é possível desenvolver a front-end utilizando estas linguagens, para o acesso de recursos

nativos é possível utilizar o AngularJS, chamado no Ionic de ngCordova que facilita o uso de plugins.(GRILLO, 2015).

No surgimento dos smartphones, as aplicações híbridas enfrentavam problemas, pois com poucos recursos de armazenamento de dados, espaços limitados de cache, e largura de banda pouco veloz, faziam com que desenvolvedores focassem para o desenvolvimento nativo (O'REILLY,2014).

Com o avanço dos navegadores, o surgimento do 3G e 4G, os principais problemas enfrentados no começo dos smartphones, foram solucionados, atraindo mais desenvolvedores para os aplicativos híbridos (O'REILLY, 2014).

Como é possível encontrar dificuldades no desenvolvimento hibrido tanto como no desenvolvimento nativo, deve-se elaborar um estudo de caso antes, para que possa levantar os principais requisitos que serão atingidos, sendo então possível escolher o melhor método de desenvolvimento.

1.1 OBJETIVOS

O principal objetivo desta pesquisa é explorar o desenvolvimento hibrido, facilitando a criação de sistemas multi-plataformas, sem a necessidade da utilização linguagens específicas para cada sistema operacional, atendendo da melhor forma a necessidade do usuário.

Outro objetivo que se pretende alcançar é apresentar uma solução para o mercado de desenvolvimento de aplicativos para dispositivos mobiles, no qual seja possível a criação de um sistema que seja compatível com diversas plataformas.

Solucionando o problema de incompatibilidade que os desenvolvedores encontram com as diversas linguagens nativas.

1.2 JUSTIFICATIVA

Nos últimos anos o uso de celulares cresceu significativamente, ultrapassando o uso do computador pessoal, segundo uma reportagem publicada pelo portal EXAME “considerando acessos 3G e 4G, a banda larga móvel fechou o ano de 2015 no Brasil com 191,8 milhões de acessos, contra 25,4 milhões em banda larga fixa”, com isso da base a acreditar que os usuários vêm preferindo a praticidade dos *smartphones* aos *desktops* ou *notebooks.* (EXAME, 2016).

Ainda levando em conta a mesma reportagem, pode-se notar também o crescimento do uso de aplicativos, que segundo o portal, “Os donos de *smartphones* no Brasil possuem, em média, 15 aplicativos instalados, dados divulgados em dezembro de 2015, e o *Whatsapp* está presente em 93% dos aparelhos” (EXAME, 2016), abrindo assim caminho para os desenvolvedores ingressarem neste mercado, porém com tantas variedades de sistemas operacionais, acabam encontrando problemas na hora de começar a programar, pois estes precisam estar atentos a diversos tipos de linguagens, o que aumenta o custo do projeto.

Com o surgimento do HTML5, tornou-se possível a criação de aplicativos híbridos, que são desenvolvidos em códigos que não pertencem a um sistema exclusivo e que possam ser abertos em diversos dispositivos (LIMA, Victor, 2016), a plataforma Ionic trouxe ainda mais facilidade na utilização desta linguagem, com recursos agregados do Apache Cordova, como plugins, é possível ter acessos a parte nativa de cada aparelho (GRILLO, 2015).

Como desenvolvedores com poucas experiências acabam sofrendo por não possuírem conhecimentos nos diversos sistemas móveis, encontram dificuldades com o desenvolvimento nativo, para solucionar isso o desenvolvimento híbrido é possível desenvolver uma aplicação que seja compatível em quase todas as plataformas, através do Apache Cordova, que da suporte aos recursos dos celulares para a aplicação feita no Ionic, através de plugins, que são códigos já prontos que o desenvolvedor pode utilizar, ou até mesmo criar, porém deve ter conhecimento na linguagem nativa do aparelho. (VASCONCELLOS, 2015).

1.3 MOTIVAÇÃO

O desenvolvimento desta pesquisa consiste no crescimento significativo do uso dos celulares nos dias atuais, como se pode ser visto, o uso pessoal deste aparelho vem aumentando gradativamente, surgindo oportunidades para desenvolvedores ingressarem neste vasto mercado, desde a criação de aplicativos que visam facilitar a vida de empresários ou até mesmo pessoas normais, com programas que sirvam para a distração do usuário.

Outro ponto importante a ser levado em conta é o surgimento da programação híbrida, uma lacuna em expansão e pouco explorada ainda, acrescentando a variedade de formas na criação de aplicativos mobiles.

1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado nas seguintes partes:

- **Capítulo 1 – Introdução**
- **Capítulo 2 – Computação Móvel**
- **Capítulo 3 – Plataformas de Desenvolvimento Móvel**
- **Capítulo 4 – Plataforma Apache Cordova**
- **Capítulo 5 – Proposta de Trabalho**
- **Capítulo 6 – Desenvolvimento do Trabalho**
- **Capítulo 7 – Conclusão**
- **Capítulo 7.1 – Trabalhos Futuros.**
- **Referências**

2 COMPUTAÇÃO MÓVEL

Neste capitulo será apresentado uma definição de computação móvel, além de informações sobre o surgimento dos principais sistemas operacionais destes dispositivos, o avanço de cronológico deles, sendo possível a percepção das diferenças existentes de cada plataforma.

2.1 Contextualização.

Os primeiros indícios de computação aconteceram em 5.500 AC, com o Ábaco, a primeira calculadora que se tem registro, eram utilizadas pelo povo da Mesopotâmia para fazerem cálculos do dia a dia, posteriormente reapareceram também nos povos da Babilônia, Grécia, Egito, Roma, Índia, China e Japão (GUGIK, 2009).

Figura 1 - Ábaco
(In: TECMUNDO, 2009)

O termo computação vem do latim, tem como principal significado o processamento de dados, como a realização de contas matemáticas, que se tem a entrada de valores, Inputs, gerando um resultado, Outputs. Em 1642, o matemático francês Blaise Pascal, desenvolveu a primeira calculadora mecânica, chamada Máquina de Pascal, eram utilizados quatros rodas que giravam na realização de algum cálculo, sua ideia era criar uma máquina que realizasse as quatro operações matemáticas, porém ela foi capaz de realizar apenas adição e subtração, caindo assim no desuso. Allan Turing entrou para a história da computação criando o Collossus, durante a segunda guerra mundial, foi utilizado para quebrar a criptografia contida nas mensagens compartilhadas pelo exercito nazista (GUGIK, 2009).

Figura 2 - Eniac
(In: TECMUNDO, 2009)

A computação moderna teve origem com a substituição dos componentes analógicos, pelos sistemas digitais, o surgimento do ENIAC, que por ser digital, dispensava a necessidade de movimentar peças de forma manual, aumentando assim a velocidade

na hora de processar seus dados, porém tinha ainda ocupava um grande espaço, medindo em torno de 5,5 metros de altura e pesando mais de 25 toneladas, não eram máquinas voltadas para o consumidor comum. O primeiro computador pessoal surgiu em 1976, o Apple I, criado por Steve Jobs, com o sucesso de vendas, foi desenvolvido o Apple II, e posteriormente o Lisa (1983) e Macintosh(1984), que contavam com mouse e interface gráfica, paralelamente a Apple, Bill Gates fundou a Microsoft, dando origem ao MS-DOS, utilizando processadores 8086 da Intel, futuramente numa parceria com Jobs, copiou toda a ideia, e deu origem ao Windows, se tornando a principal concorrente do Macintosh, trazendo a computação para o cenário que é conhecida na atualidade (GUGIK, 2009).

Com a proposta de conectar diversos tipos de cultura, trazendo informações em tempo real de vários lugares do mundo, o uso destas máquinas caiu no gosto de seus consumidores, trazendo ainda mais a dependência dos usuários se manterem conectados a qualquer momento, foi então que surgiu a Computação móvel, aonde o surgimento de celulares inteligentes, pequenos e de fácil locomoção, e graças a conexões de internet via 3g, era possível ter acesso a redes sociais, jornais e noticias em qualquer lugar.

2.2 Fundamentos da computação móvel.

A Computação móvel tem como tem como principal fundamento estar sempre disponível ao usuário, independente do lugar onde estiver sem a necessidade da locomoção de um lugar próprio com tecnologia para acessar o que se pretende, como o caso dos celulares que é de fácil locomoção, tem acesso a internet através do 3g e 4g, podendo gravar informações em situações cotidianas, para serem revisadas porteriormente (MONTEIRO, 2006).

Outro fundamento importante da computação móvel é a Usabilidade, que é divida em três partes, característica do usuário, podendo haver alterações devidos aos usuários, conforme o tamanho e a força, um notebook pode ser facilmente transportado por um adulto, mas seria uma tarefa complicada para uma criança. Outro ponto que afeta a usabilidade ocasionada pelo usuário é a questão da flexibilidade e a destreza, uma pessoa com dedos mais grossos pode ter problemas na digitação em um teclado

menor, e por ultimo a falta de conhecimento e capacidade também podem afetar a utilização destes dispositivos. (GOMES, 2016).

Ainda sobre a usabilidade, pode ter o desempenho diminuído por causa do ambiente, como as condições de uso, um tablet será mais eficaz para um usuário que trabalha por horas em pé do que um notebook, outro ponto a ser levantado é a condição extrema, como caso de pessoas que trabalham nestas condições como frio, temperaturas altas, chuvas, e que precisem de dispositivos que consigam suportar este ambiente. (GOMES, 2016).

E por fim a característica do dispositivo também pode afetar a sua usabilidade, como o tempo de inicialização, se um usuário precisa acessar alguma informação em tempo critico, será necessário um dispositivo mais veloz, outro ponto é a integridade dos dados, um dispositivo não pode perder informações armazenadas, e deve conter uma capacidade de armazenamento adequada. A interface com o usuário pode também afetar sua utilização, pois um tablet com uma caneta se tem facilidade para desenhar do que um notebook utilizando um mouse. A robustez/ resistência são outros fatores que influenciam diretamente na usabilidade do dispositivo, um aparelho mais resistente pode perder a mobilidade, um notebook é mais resistente que um celular. (GOMES, 2016).

2.3 Caracterização de um Dispositivo Móvel.

Dispositivos móveis têm como característica a portabilidade, podendo ser levado para todos os lugares que o usuário precise, quanto menor tamanho e maior a autonomia de sua bateria, maior será sua mobilidade, por serem dispositivos pequenos, tendem a ter uma interface mais fácil de usar, com recursos diretos, seus custos operacionais são menores por não possuírem disco rígidos e discos flexíveis, diminuindo consideravelmente os custos dos mesmos. (ROMEIRO, 2005).

2.4 Aparecimento do Smartphone.

O primeiro Iphone, o 2G, foi lançado em 2007, as primeiras versões não tinham sua própria SDK, e então os aplicativos eram feitos na WEB, como naquela época ainda a conexão era de modo EDGE, pouco veloz comparado com o 4G, os arquivos não podiam passar de Kbytes, Nicole Lazarro e mais três pessoas criaram o primeiro jogo para este dispositivo, o TILT, que usava capacidade de detecção de movimentos, já Dori Smith criou o *Iphone* Bingo. O'Reilly, Richard Herrera, Ryan Christianson e Wai Seto criaram o *Pickleview*, uma junção de TWITTER com a *Major League Baseball*, baseado em AJAX (O'REILLY, 2014).

Em 2008 com o lançamento da primeira SDK, foi possível a criação de aplicativos nativos para o celular, com a abertura da loja virtual no mesmo ano, desenvolvedores passaram a ter a possibilidade de desenvolver aplicativos e colocarem na APP STORE para vendas, o foco passou intensamente para aplicativos nativos, deixando um pouco de lado os *web apps*, pois com poucos recursos, pois o Safari limitava o tamanho dos arquivos para 10MB, a falta de armazenamento e o cache limitadíssimo, tornavam os aplicativos nativos mais poderosos (O'REILLY, 2014).

O Android, principal concorrente do iOS, foi criado em 2003, por um grupo de 4 amigos empresários, no inicio do projeto visavam o comercio de câmeras fotográficas, porém vendo o mercado com poucas opções, decidiram então partir para o desenvolvimento móbile, criando o que chamaram de "O Sistema mais inteligente", com códigos abertos e baseado no Kernel do Linux surge o Android Inc, que em 2005 é comprada pela Google (MEYER, 2016).

No mesmo ano do lançamento da Apple, marcas que corriam por fora, como a Samsung, Sony HTC e empresas americanas como Nextel e T-Mobile, e fabricantes de hardwares, se uniram a Google, para o lançamento de uma plataforma com código aberto, surgindo o primeiro Android comercial. Com os principais sistemas lançados no mercado, começa a dar inicio a briga pela liderança nas vendas, prêmios para desenvolvedores são distribuídos visando terem os melhores aplicativos para sua plataforma (MEYER, 2016).

O Android se popularizou com as diversas marcas de celulares, hoje está presente nas principais fabricantes de aparelhos móveis, como a Samsung, LG, Asus, Motorola,

recentemente comprada pela Google, enquanto a Apple se manteve firme fabricando e utilizando exclusivamente seu sistema. Entrando no mercado de dispositivos mais baratos e nos tops de linha, o sistema popular esteve em 81% dos aparelhos vendidos no ano de 2014, Segundo a Strategy Analytics (MEYER, 2016).

Mesmo a Apple fabricando seus próprios aparelhos e tendo o iOS de forma exclusiva, consegue manter o seu nicho, busca atrair o mercado dos celulares premiuns, a Forbes fez um levantamento em 2016, no qual a Apple foi considerada pela sexta vez a empresa mais valiosa do mundo, superando em 87% a Google, segundo colocado (G1, 2016). Como o sistema é desenvolvido exclusivamente para seus próprios dispositivos, isso reduz bastante os “bugs”, além dos aplicativos passarem por um revisão nos códigos antes de estarem disponível para compra na AppStore, reduzindo o risco de contaminação por malwares.

As duas principais empresas divergiram bastantes entre si, com sistemas e linguagens de programação distintas, grandes empresas tiveram que buscar equipes de programadores especializadas em cada plataforma, pois precisavam desenvolver dois aplicativos iguais em códigos diferentes, como o caso do WhatsApp, o aplicativo mais popular, que usuários encontram algumas diferenças no designer e até em alguns recursos. Outro ponto também são softwares exclusivos para tais dispositivos, ou mesmo lançamentos que ocorrem primeiro em alguma plataforma, para só depois de um tempo ser lançado na outra.

2.5 Desenvolvimento Híbrido.

O desenvolvimento híbrido consiste em um aplicativo web, que será aberto por um navegador disponível no dispositivo, terá acesso as APIs de cada plataforma, através de um framework é possível ter o rendimento esperado, capaz de fazer um aplicativo robusto, parecido com um nativo (VASCONCELLOS, 2015).

Para o desenvolvimento é necessário ter uma framework que tenha suporte para o HTML5, CSS3 e JAVASCRIPT, a ***front-end*** será escrita com estas linguagens, através do Apache Cordova será possível ter acesso as APIs das plataformas, permitindo a utilização de recursos de hardwares destas, como o GPS, câmera, entre outros (O’REILLY, 2015).

	Principais Vantagens	Principais Desvantagens
Web Apps	Executados pelo browser, proporcionando o uso em outras plataformas; Atualização e distribuição rápida e abrangente, não precisam ser baixados ou atualizados; Acesso rápido e fácil, os usuários têm acesso imediato pelo Smartphone.	Pouca ou quase nenhuma integração com o hardware do dispositivo em que está sendo executado; Mais lentos, dependendo da conexão com a Internet; Interação entre o usuário e o aplicativo menos rica em funcionalidades
Nativos	Interação entre o usuário e o aplicativo mais rica em funcionalidades e recursos; Velocidade na execução. Independente da Internet.	Uma nova aplicação escrita para cada plataforma diferente Distribuição e atualização dependentes de lojas online; (Apple Store, Google play).
Híbridos	Compartilhamento de boa parte do código entre plataformas; Possibilidade do uso de recursos da plataforma com código nativo; Pode ser distribuído lojas online; (Apple Store, Google play) Interoperabilidade.	Performance. Limitação de design.

(SILVA, PIRES, NETO, 2015).

As vantagens deste método estão no ganho de tempo, e por ser escrito em HTML5, que é baseado no HTML, os desenvolvedores já tem alguma experiência, como esta linguagem consegue ser interpretadas por quase todos os novos navegadores, elimina o processo de escrever dois códigos em linguagens diferentes para cada plataforma, ganhando tempo e evitando a necessidade de possuir mais de um desenvolvedor para cada plataforma (SILVA, PIRES, NETO, 2015).

A plataforma Ionic é utilizada para o desenvolvimento da interface aonde o usuário terá acesso, como esta framework trabalha simultaneamente com o Apache Cordova, consegue fazer a tradução de comandos utilizados na front-end, para códigios de baixo nível, o desenvolvedor não encontrará problemas quando for utilizar recursos de hardware, para facilitar estes processos os desenvolvedores do Ionic criaram plugins, chamados de ngCordova.(VASCONCELLOS, 2015).

3 Plataforma de Desenvolvimento Móvel.

Neste capítulo serão apresentadas as principais plataformas da computação móvel, mostrando o funcionamento de cada uma delas, os fundamentos e as principais plataformas de desenvolvimento.

3.1.1 Fundamentos iOS.

O iOS é um sistema operacional móbile criado pela Apple, utilizado em todos os dispositivos móveis da corporação, não é um sistema de código aberto e é exclusivo da Apple, portanto só a marca pode usar em seus dispositivos(ROCHA, NETO, 2011).

O iOS é dividido em 4 camadas, baseado no MAC OS X, com algumas alterações feitas para ser um sistema leve e portátil, os aplicativos são escritos em Object C/C++, após serem desenvolvidos passam por uma revisão, realizada pela loja a App Store, aonde será verificado o código para que não contenha arquivos prejudiciais ao dispositivo, após ser aprovado, estará disponível na loja para que seja feito o *download* (O'REILLY, 2014)**.**

.

3.1.2 Arquitetura iOS.

O iOS é dividido em quatro camadas, no nível superior estão os aplicativos e serviços mais robustos, os softwares desta camada quase raramente se comunicam com o hardware, para isso se tem uma interface responsável por toda a comunicação, protegendo assim os aplicativos de qualquer alteração de hardware, para que seja possível o desenvolvimento existem frameworks que possuem abstração orientada a objeto de níveis inferiores, resultando na diminuição da escrita de códigos (ROCHA, NETO, 2011).

.

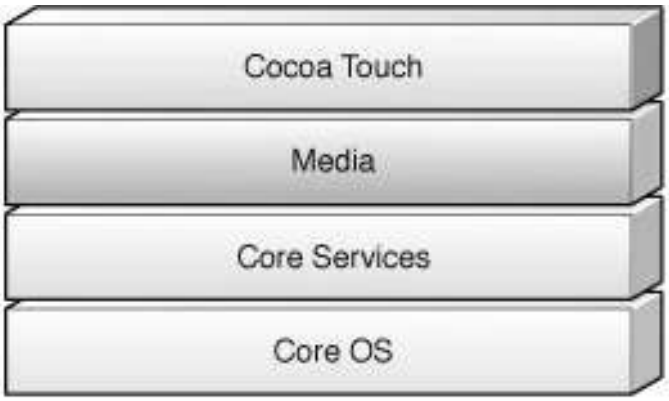


Figura 3 – Camadas iOS.
(In: ROCHA, NETO, 2011).

Na camada Cocoa Touch, a de nível superior, estão estes frameworks, nela é esta disponível o sistema de multitarefas, o serviço de notificação Apple push e proteção de dados. Para economizar bateria, os aplicativos não são finalizados quando a tecla HOME é pressionada, eles ficam guardados na memória principal do dispositivo, ficando em segundo plano, sem que nenhum código seja executado (ROCHA, NETO, 2011).

A proteção de dados é uma forma segura para que o usuário tenha seus arquivos confidenciais guardados, quando o dispositivo é bloqueado, estes dados são armazenados no disco em forma criptografada, porém quando o aparelho enfim é desbloqueado pelo o usuário cria-se uma chave de decodificação para que o aplicativo possa ter acesso. No sistema de notificação o programador pode criar textos, ícones e sons de alerta, para que possa chamar a atenção do usuário (ROCHA, NETO, 2011).

A camada media é responsável pelo gráfico, vídeos, e tecnologia de áudios, nela também é possível encontrar frameworks que auxiliam o desenvolvedor na criação de seus aplicativos, esta tecnologia é foi criada para facilitar a reprodução de arquivos multimídia (MARQUES, 2015).

A camada Core Services é responsável pelos serviços fundamentais que quase todos os aplicativos utilizam, como por exemplo a criação de banco de dados, ou até mesmo a venda de serviços em seus aplicativos, mesmo que o programador não utilize esta

camada diretamente, diversas partes do sistema foram construída com base nela(MARQUES, 2015).

A camada de baixo nível é conhecida como Core OS, nela é possível fazer a comunicação com o hardware, existem também frameworks para facilitar o desenvolvimento de aplicativos que tenham acesso direto ao hardware(ROCHA, NETO, 2011).

3.1.3 Plataforma de Desenvolvimento iOS.

O desenvolvimento nativo para o iOS é escrito em Object-C, as APIs são fornecidos para o desenvolvedor através da plataforma, a mais comum é o XCode, que é fornecida de forma gratuita pela Apple (BRAWERMAN, ALESSANDRO).

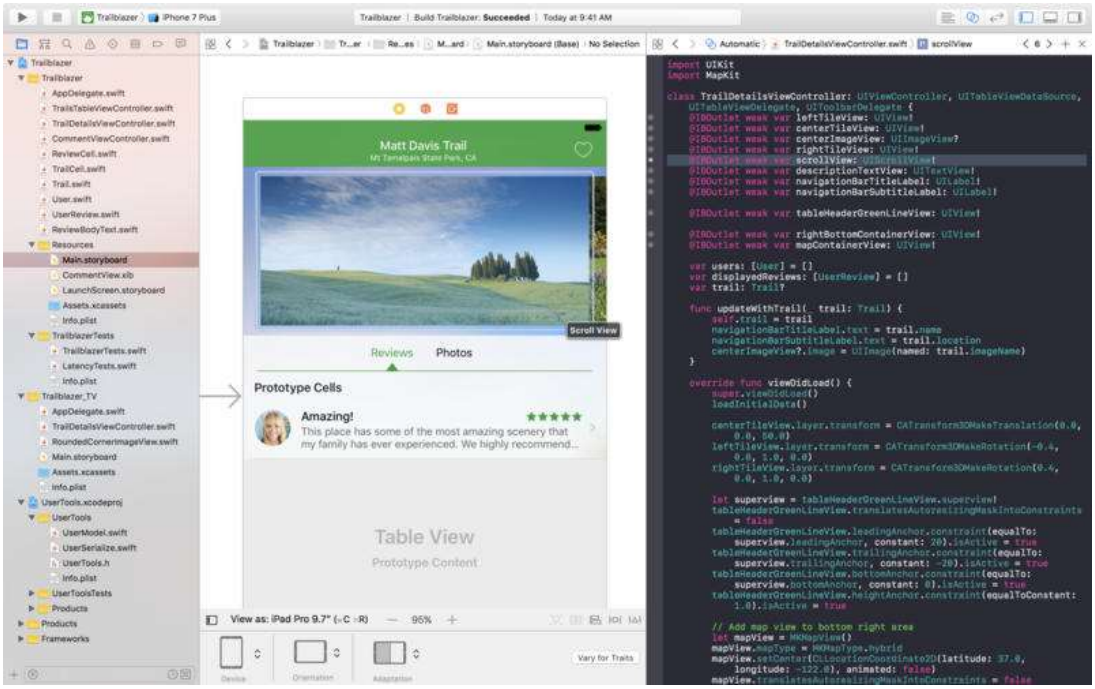


Figura 4 – Plataforma XCode
(In: ITUNES, 2017)

O XCode contém diversas ferramentas afim de facilitar a vida do desenvolvedor como o compilador Apple LLVM, que além de compilar o código, ainda fornece uma ferramenta inteligente para completar o código, o Interface Builder, a plataforma fornece também o iOS Simulator, um simulador do sistema operacional iOS, para que o desenvolvedor consiga testar o seu aplicativo (BRAWERMAN, 2017).

3.2 Plataforma Android.

3.2.1 Fundamentos do Android.

O Android é uma plataforma para *smartphones*, foi baseada no Linux, possui diversas ferramentas para criação de aplicativos, bibliotecas de recursos e diversos componentes. O projeto chamou a atenção da Google pelo fato de ser uma plataforma aberta aos fabricantes, deixando assim um sistema flexível e atualizável, permitindo mobilidade ao usuário (SIMÕES, 2014).

O Android é dividido em 5 (cinco) camadas, Kernel, runtime, bibliotecas, framework e aplicativos, como é uma plataforma aberta (SIMÕES, 2014), pode ser encontrada em diversos dispositivos que não são comuns, como em *notebooks*, *smartbooks*, câmeras digitais, *smart's tvs*, tocadores de mp3, relógios e entre outros, dando um grande incentivo para a internet das coisas (MEYER, 2016) .

O Android possui a tela de bloqueio, aonde o sistema pode requerer um código definido pelo usuário para que possa ter acesso a tela inicial, após o desbloqueio o utilizador do sistema consegue deixar amostra os principais aplicativos para ter acesso rápido a eles, no menu principal estão todos os softwares instalados e recursos disponíveis.

Figura 1 – Interface Android
(In: TECHTUDO, 2013)

A Interface de um aplicativo em Android é criada usando objetos chamados de ***View***, que desenha algo na tela aonde o usuário pode interagir, para definir o *layout* da pagina é utilizado o ***ViewGroup***, que contém outros objetos ***View***. (Fonte: DEVELOPERS, 2017).

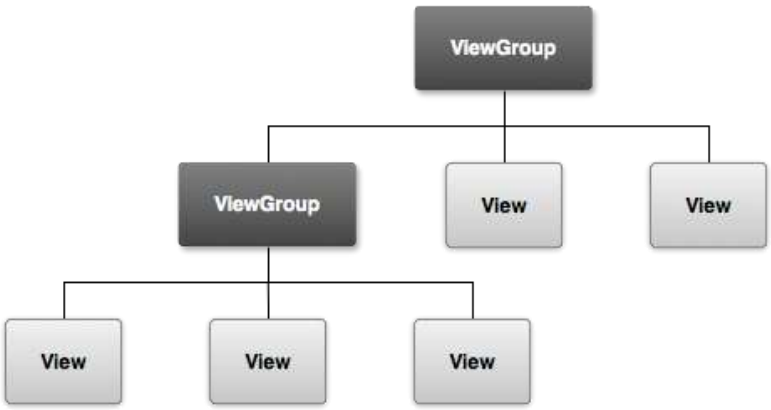


Figura 6 – Funcionamento de uma ViewGroup
(In: DEVELOPERS, 2017)

3.2.2 Arquitetura Android

O Android foi baseado no Kernel do Linux 2.6, porém foram realizadas alterações para alcançar um aproveitamento melhor nos dispositivos móveis, entre essas modificações é possível destacar o Binder, um mecanismo entre comunicação de processos, que permite que um processo possa chamar uma rotina em outro processo. Como o Android foi criado para atender a computação móvel, foi necessário também a adição de um gerenciador de energia, isso resulta no principio de portabilidade, outra mudança foi a criação do Android Debug Bridge, além de um sistema de log separado do Linux. (BORDIN, 2012)

.

Figura 7 – Divisões das camadas do Android
(In: Embedded Systems, 2011).

A arquitetura do Android é composta por cinco camadas, acima do kernel é possível ter acesso as bibliotecas C/C++, WebKit, banco de dados, entre outrass. Os aplicativos escritos em java, são executados pela Java Virtual Machine, já o Android Runtime é composto pela maquina virtual Dalvik VM, nesta não é possível a interpretação de códigos em Java, transformando arquivos .class compilados por compiladores Java normal em .dex, que não são específicos da Dalvik VM, foi otimizada para dispositivos com pouca memória e desenhada a permitir varias instancias de modo eficiente, e que deixa a cargo do kernel Linux algumas tarefas(BORDIN, 2012).

As Frameworks de aplicação são compostas por atividades, que são ações especificas que os usuários podem realizar dentro de um aplicativo, estas ações podem incluir a inicialização de novas atividades, tanto dentro como fora da aplicação,

conhecidas como Intent, criando uma nova tarefa, estas podem conter diversas atividades, organizadas em pilhas. Quando o usuário começa uma nova tarefa ou volta para o menu inicial, a aplicação fica em segundo plano, caso o sistema de multi-tarefas esteja com diversas aplicações em segundo plano, o sistema pode começar a destruir atividades para a liberação de memória (BORDIN, 2012).

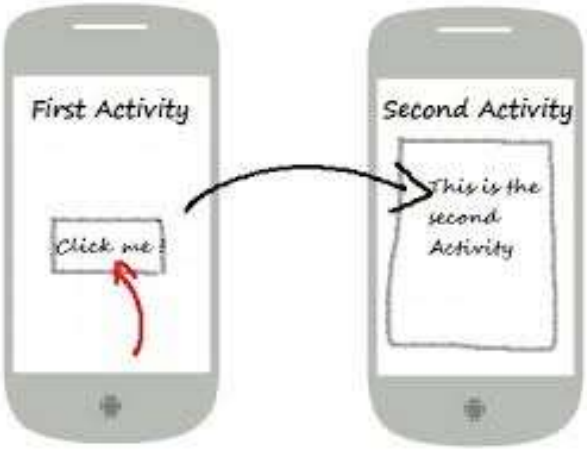


Figura 8 – Funcionamento de uma Activy

O Activity Manager é responsável pela organização do ciclo de vida das atividades, O Resource Manager prove acesso a tudo que não faz parte de um aplicativos, como arquivos XML, bitmap e sons, diferente destes o Service é destinado a execução em segundo plano, como a comunicação entre web service e os Broadcast Receivers, que responde a anúncios feitos para todo o sistema, como quando a tela desliga ou a bateria esta acabando(BORDIN, 2012).

3.2.3 Plataforma de Desenvolvimento do Android

O desenvolvimento de aplicativos para Android, tem como linguagem principal o Java, e a plataforma mais utilizada é o Eclipse, por ser uma ferramenta de código aberta e com extensões origina códigos em Python e C/C++ (KAMADA, ISHIDA, GOMES, CARPEJANI, 2012).

Com um plug-in do Android SDK que é compatível com o Eclipse, consegue se ter um emulador da plataforma móbile dentro da ferramenta, conseguindo assim utilizar

o *debugging,* e obviamente testar o aplicativo sem a necessidade de um celular com o sistema instalado (KAMADA, ISHIDA, GOMES, CARPEJANI, 2012).

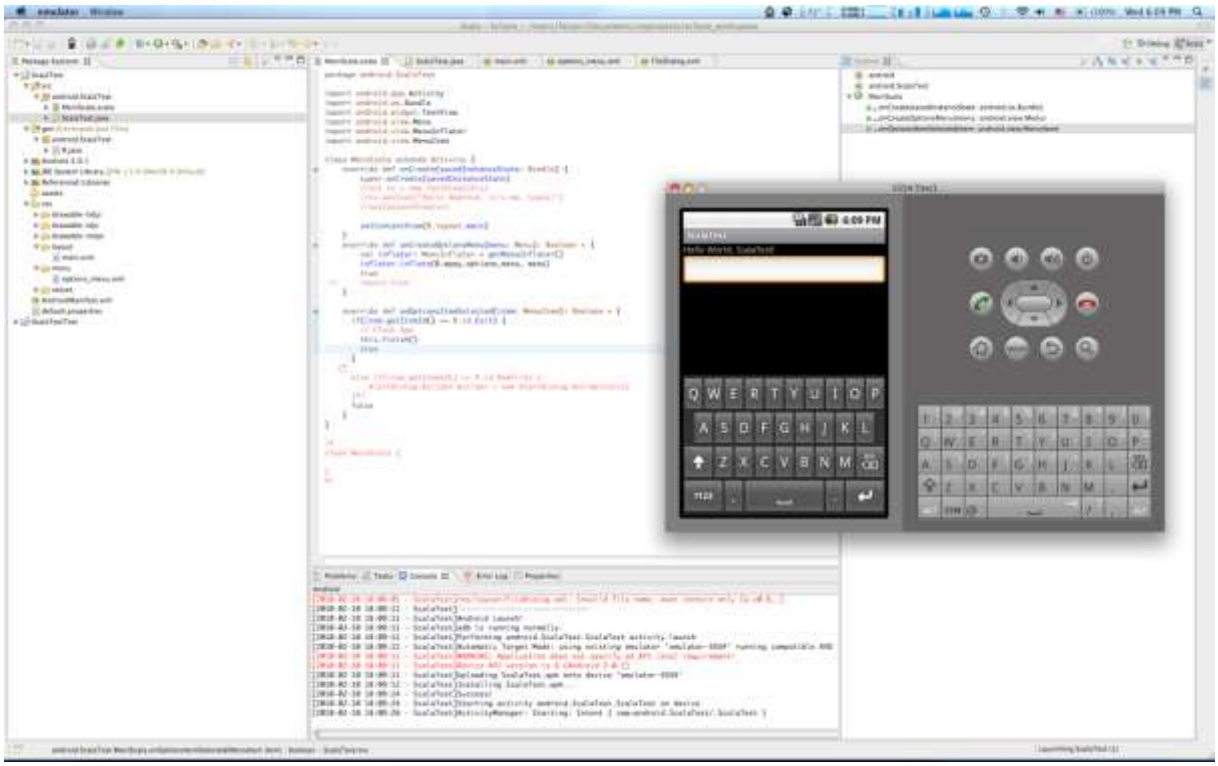


Figura 9 – Plataforma Eclipse com Android SDK
(In: TILIMAN, 2010)

Outra ferramenta que vem ganhando destaque para o desenvolvimento de aplicativos para Android é o Android Studio, para sua instalação é apenas necessário ter o kit de desenvolvimento Java (CRIS, 2017).

Desenvolvedores que migraram do Eclipse para o Android Studio, podem encontrar problemas com a interface, pois possuem algumas diferenças, porém quem utiliza encontra uma ferramenta mais didática, necessitando de menos configurações e aumentando significativamente a sua produção (CRIS, 2017).

4 PLATAFORMA APACHE CORDOVA

O Apache Cordova é um framework open souce, foi criado pela Nitobi, inicialmente com o nome de phonegab e posteriormente comprado pelo Adobe, que logo após doou o core do projeto para a Apache, uma empresa com muito prestigio, com o intuito que outras empresas ajudassem no projeto, como o phonegab foi registrado pelo Adobe, o novo projeto recebeu o nome de Cordova (COSTA, 2013).

As duas frameworks tem o mesmo proposito, trabalham fazendo a tradução dos comandos das aplicações de nível superior para o código nativo do aparelho, utilizando um sistema de plug-ins, é possível ter acesso a câmera, gps, bateria e entre outros, quando o Cordova é instalado vem sem nenhum plugin, dessa maneira o aplicativo só terá recursos que serão necessários, eles podem ser adicionados ou removidos pelo comando **plugin add** ou **plugin rm**, caso o desenvolvedor tenha experiência com o código nativo do aparelho, também é possível criar (COSTA, 2013)..

Os aplicativos híbridos são desenvolvidos em HTML5, pois todos os dispositivos tem suporte a esta linguagem, são utilizadas também o CSS3 e o JavaScript.

4.1 Fundamentos da Plataforma

Um dos problemas encontrados pelos desenvolvedores com a plataforma Apache é o delay, que causa uma sensação de lentidão para o usuário, este problema ocorre em diversas plataformas de desenvolvimento híbrido e tanto com os web apps, utilizando o **FastClick** e os seguintes comandos, é possível resolver este problema (COSTA, 2013).

```
<script type='application/javascript' src='/path/to/fastclick.js'> </script>
```

```
window.addEventListener('load', function() {
FastClick.attach(document.body);
}, false);
```

Com o Apache Cordova é possível criar códigos que serão compilados em diversos sistemas, como o Android, iOS, Windows Phone, Firefox OS, Amazon Fire OS, BlackBerry, Tizen e Ubuntu, todas as APIs são instaladas na própria aplicação (COSTA, 2013)..

Para se criar um novo projeto no Apache Cordova, deve-se ter um diretório, aonde todo o projeto deverá ficar armazenado, o comando **cordova create** irá gerar um arquivo esqueleto baseado em web com a home **www/index.html** (CORDOVA, 2017).

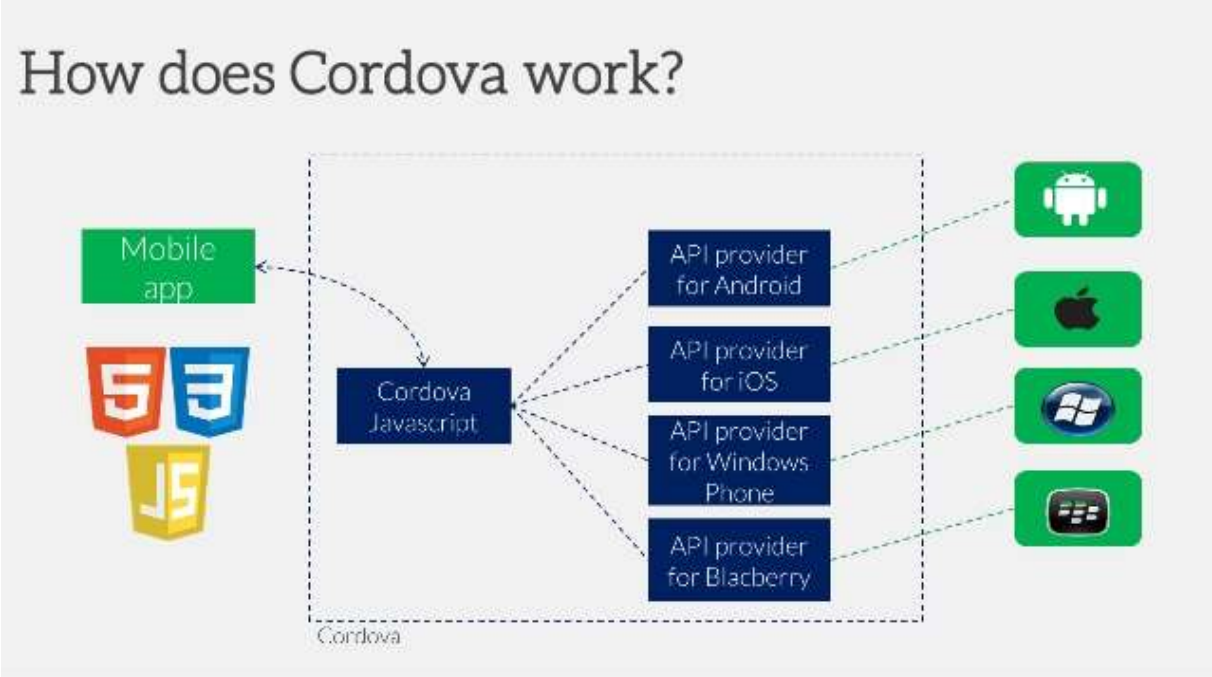


Figura 10 – Funcionamento do Cordova
(In: CORDOVA, 2017)

Para segmentar as plataformas iOS e no Android, deve-se utilizar o comando **cordova plataform add ios --save** e o **cordova plataform add android --save**, utilizando o --**save** a plataforma irá salvar as configurações realizadas no arquivo **config.xml** (CORDOVA, 2017).

Utilizando o **cordova build**, criará o projeto para todas as plataformas, caso o desenvolvedor queira criar seu projeto visando plataformas específicas deve utilizar o comando build e mencionar a plataforma desejada, como **cordova build ios** (CORDOVA, 2017).

O desenvolvedor pode também criar plug-ins, que são partes de códigos injetadas na aplicação para que a Webview do cordova consiga se comunicar com a plataforma nativa, são escritas em *JavaScript* e possuem partes de códigos nativos da plataforma desejada (CORDOVA, 2017).

4.2 IONIC

O Ionic é uma SDK responsável pela parte gráfica do desenvolvimento mobile, seus componentes HTML5, CSS e JavaScript são de fácil manuseio, como é focado na parte visual do sistema, é necessário que tenha o Apache Cordova instalado, para que faça a tradução dos comandos para o código nativo do aparelho(RAVULAVARU, 2015).

Com estas frameworks o desenvolvedor se livra de ter que codificar tudo do zero, outra função importante desta ferramenta é o **ngCordova**, que empacota os plug-ins do Cordova para dentro de serviço angular, o que facilita a utilização destes. O seu principal diferencial é ser baseado no angular, que é uma framework *JavaScript* bastante popular, desenvolvedores experientes nesta ferramenta, acabam encontrando facilidade com o Ionic, além também de conter um fórum com bastantes acesso aonde desenvolvedores se ajudam, facilitando ainda mais seu uso, a performance é um dos diferencias, pois procura sempre a otimizar, quanto maior a fluidez e a responsividade, melhor será a aceitação da aplicação híbrida.

O Ionic possui três modelos já prontos de projetos que podem ser utilizados como inicialização rápida, como o **Blank**, que gera um projeto em branco, **Tabs**, este é um

exemplo de aplicativo criado utilizando **Tabs** do Ionic, e o **Slide Menu**, que gera um menu lateral (RAVULAVARU, 2015).

Para se criar um projeto em branco, deve-se utilizar o **comando ionic start -a "Exemplo 1" -i app.exemplo.um exemplo1 blank**, que o **Exemplo 1** será o nome do projeto, -i app.examplo.um, este será o ID e domínio da aplicação, e o exemplo1 será o diretório aonde ficará armazenado o projeto (RAVULAVARU, 2015).

Enquanto o projeto estiver sendo criado, pelo **prompt de comando** serão impressos diversas informações, cinco **plug-ins** do cordova serão baixados para a pasta do projeto, **org.apache.cordova.device**, que é utilizado para obter informações do dispositivo, **org.apache.cordova.console**, este *plug-in* torna o **console.log** útil, **cordova-plugin-whitelist**, implementa uma lista de politicas de visualização web do aplicativo Cordova 4.0, **cordova-plugin-splashscreen**, mostra e esconde uma tela do aplicativo, **com.ionic.keyboard**, este *plug-in* faz com que a interação com o teclado seja mais fácil, enviando eventos para que possa mostrar ou oculta-lo (RAVULAVARU, 2015).

O servidor do Ionic vem como padrão a porta 8100, caso o desenvolvedor tenha algum aplicativo utilizando esta mesma porta, pode encontrar algum erro na hora de iniciar a aplicação, com o comando **ionic serve –p "numero da porta"**, o desenvolvedor pode alterar a porta padrão, para alguma de seu gosto, as configurações são armazenadas em um arquivo chamado **package.json** é aconselhado utilizar o Google Chrome ou o Mozilla Firefox, como navegadores para a realização dos testes (RAVULAVARU, 2015).

O projeto estará estruturado da seguinte forma:

- bower.json
- config.xml
- gulpfile.js
- hooks
- ionic.project
- package.json
- plugins
- scss
- www

No **bower.json** ficará armazenado as dependências carregadas através do ***Bower***, o **config.xml** será utilizado para armazenamento das informações necessárias das tarefas do Cordova, **gulpfile.js** armazenará as configurações de compilamentos utilizadas, o **ionic.project** estarão dentro dele todas as informações do aplicativo Ionic, o **hook** consiste em códigos que serão utilizados quando uma tarefa do Cordova for executada, os plug-ins instalados pela aplicação ficarão armazenados na pasta plugins, toda a parte da aplicação desenvolvida no ionic, como a interface, ficará armazenada dentro da pasta **www** (RAVULAVARU, 2015).

4.3 Exemplos de Aplicações

Através do site showcase.ionicframework.com é possível encontrar algumas aplicações feitas com a plataforma ionic, como o **National Museum of African History and Culture,** desenvolvido pela Clearly Innovative com este aplicativo é possível ter informações sobre o prédio do museu, como a inspiração para a forma do prédio, também tem uma linguagem própria para jovens, dando dicas de perguntas para aumentar a comunicação dentro do museu, dicas de lanchonetes, história sobre os impactos dos afro-americanos no país, utilizando também a realidade aumentada, com imagens de objetos e a história de cada um deles.

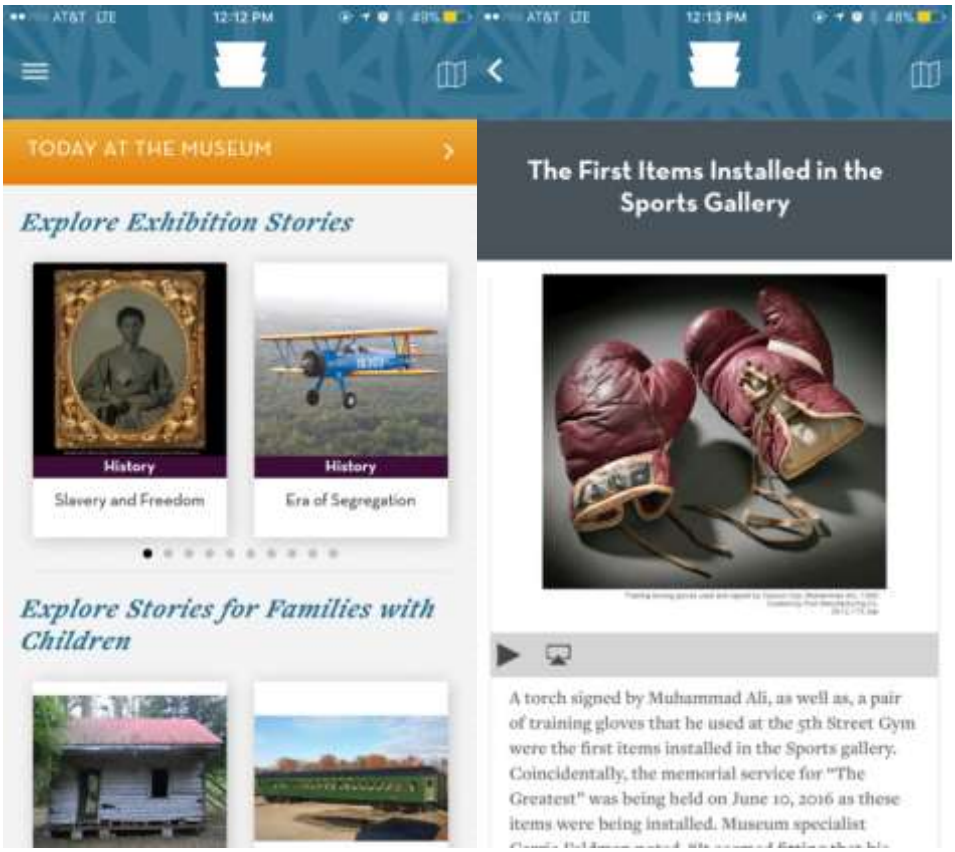


Figura 11 – Aplicativo do National Museum of African History and Culture (In: SHOWCASE, 2017)

Outro aplicativo que também pode ser encontra neste site é o **Ionic View**, desenvolvido pelo próprio Ionic, um aplicativo aonde desenvolvedores que utilizam a ferramenta podem publicar suas aplicações, para que usuários possam testa-las, conseguindo assim que seus aplicativos tenham uma visibilidade maior, e possam ter um *feedback* sobre eles.

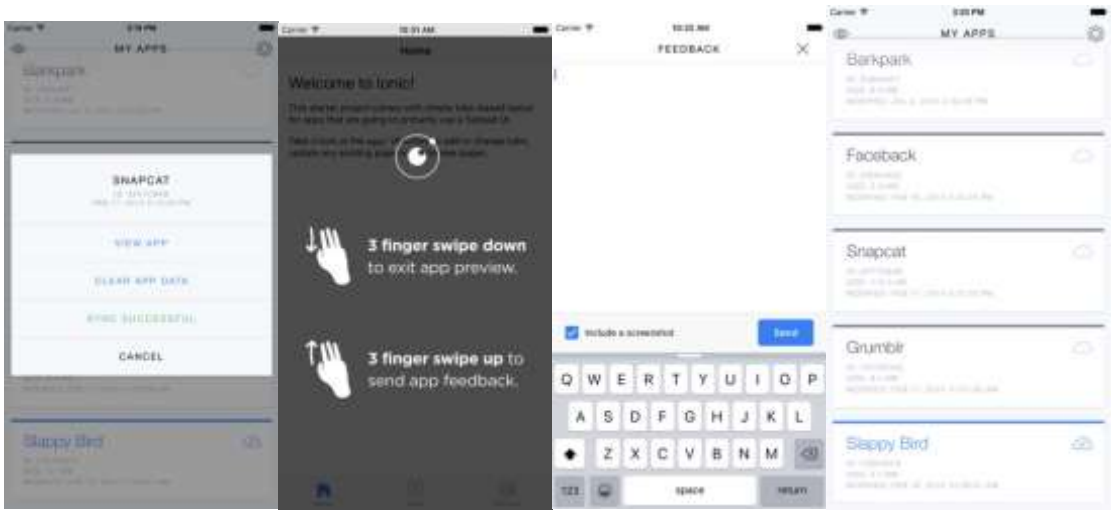


Figura 12 – Aplicativo Ionic View
(In: SHOWCASE, 2017)

4.4 Outros Recursos

4.4.1 HTML5

O HTML5 foi desenvolvido para suprir as principais falhas do HTML, a linguagem Web mais utilizada, precisava ser remodelada para então entrar de vez no ramo dos aplicativos mobiles, com os avanços dos principais navegadores e o suporte para esta nova tecnologia, foi possível criar uma nova linguagem que suportava arquivos de vídeos, animações e áudio, recursos que faltavam na antiga versão, que era preciso adicionar complementos, que no caso dos celulares podiam enfrentar alguns problemas de compatibilidade, com as atualizações tornaram-se possível competir com igualdade com os aplicativos nativos, atraindo ainda mais os desenvolvedores, pois de uma forma mais fácil, barata e prática. Como é uma linguagem universal, que todos os navegadores conseguem interpretar, faz com que não seja necessário o desenvolvimento de aplicativos diferentes para cada plataforma, mantendo TAGs antigas e adicionadas algumas novas, se tornou uma ótima opção para desenvolvedores com pouco conhecimento no desenvolvimento móvel (O'REILLY, 2014).

O HTML5 é um aperfeiçoamento do HTML4 e o XHTML, engloba elementos das versões anteriores, e foram retirados alguns que eram obsoletos, anteriormente os desenvolvedores tinham que criar um padrão e detalhar como os navegadores devem tratar esse padrão, as especificações do HTML5 detalham precisamente como os navegadores devem fazer isso. (O'REILLY, 2014).

O HTML5 adicionou novos atributos globais à lista de atributos básicos, como **accesskey**, **hidden**, **tabindex** e os interativos propostos **contentditable**, **contextmenu**, **speelcheck**, **draggable** e **dropzone**, para fazer a declaração do tipo de documento no HTML5 é usado o comando **<!DOCTYPE html>*** , nas antigas versões o comando era extenso e complicado para o desenvolvedor memorizar, outra diferença entre as versões antigas, não é mais obrigatório o uso da tag **<html>** , já o **<title>** e o **<body>** ainda é obrigatório a utilização da tag de encerramento (O'REILLY,2014).

4.4.2 CSS3

Utilizado como uma ferramenta de estilização para o conteúdo de um HTML, com ele é possível criar estilos para imagens, textos, áudio, vídeos ou qualquer outro item inserido na pagina, também é possível criar site responsivo, que se adaptam ao tamanho de tela do dispositivo, auxiliando bastante na criação de aplicativos para celulares, a maioria das regras de CSS seguem a seguinte forma:

```
selctor{
property1:value1;
property2:value2;
}
```

Os seletores mostram ao navegador que elemento procurar, a propriedade é o recurso que será afetado com o valor que é definido em seu campo, também é possível utilizar estilos prontos que são acessados através de links externos, o CSS também tem

recursos para estilização de áudios, como o desenvolvedor pode definir o volume da voz, definir voz masculina ou feminina (REIS, 2015).

5 PROPOSTA DE TRABALHO

Nos últimos anos o iOS e o Android se tornaram os sistemas operacionais mobiles mais populares, por sua facilidade de uso, diversos aplicativos, estão presentes na maioria dos aparelhos vendidos, porém são sistemas completamente diferentes entre si, tendo incompatibilidades em todos aplicativos nativos, fazendo com que desenvolvedores busquem a especialização na maioria das em vezes em apenas um destes.

Esta pesquisa busca explorar uma forma de desenvolver aplicativos que consigam superar esta incompatibilidade, ajudando assim desenvolvedores com pouca experiência e pouco investimento, consiga desenvolver um único aplicativo que tenha compatibilidade com a maioria dos sistemas operacional mobile.

A proposta de trabalho tem como finalidade explorar o método de Desenvolvimento Híbrido, apresentando as principais tecnologias para este desenvolvimento como o Ionic e o Apache Cordova, que no final do trabalho será desenvolvido um aplicativo como exemplo, aonde será realizado a interação do aplicativo com o hardware do dispositivo, de modo que os principais recursos do aparelho sejam utilizados e comprovando a compatibilidade com a maioria dos sistemas operacionais.

5.1 Proposta de Aplicação

A aplicação a ser desenvolvida após esta pesquisa tem como finalidade a apresentação da ferramenta utilizada e debatida pela mesma, com suas funcionalidades.

Pretende-se utilizar dentro da aplicação os recursos do Apache Cordova, para que a mesma tenha acesso aos recursos nativos do dispositivos, como por exemplo a Câmera, o GPS e outros.

Também será utilizado uma forma de armazenamento de dados, para que seja possível que o usuário consiga armazenar informações dentro da aplicação.

Esta aplicação será compatível com Android, IOs e o Windows Mobile, com recursos iguais, comprovando então o principal objetivo do Desenvolvimento Híbrido, a compatibilidade com diversas plataformas.

5. 2 Recursos da Aplicação

A aplicação contará com uma breve descrição da pesquisa, aonde o usuário conseguirá navegar através de um menu, aonde estarão listados todos as seções do aplicativo.

A página inicial da aplicação servirá como a introdução da pesquisa, mostrando o principal objetivo a ser alcançado pela mesma, vantagens no Desenvolvimento Híbrido e os principais problemas enfrentados pelos desenvolvedores nativos.

As próximas páginas foram utilizadas como uma breve introdução as plataformas utilizadas, descrevendo os principais pontos de cada uma delas.

Após a parte descritiva da aplicação, tem a demonstração de utilização dos recursos, o primeiro é o GPS, na qual foi utilizado um plugin que é fornecido pelo Google.

Outro recurso também disponível foi a implementação de código e o plug-in fornecido pelo Apache Cordova, para se ter acesso a Câmera do dispositivo.

E por fim uma parte aonde será possível o armazenamento de dados, inseridos pelo usuário.

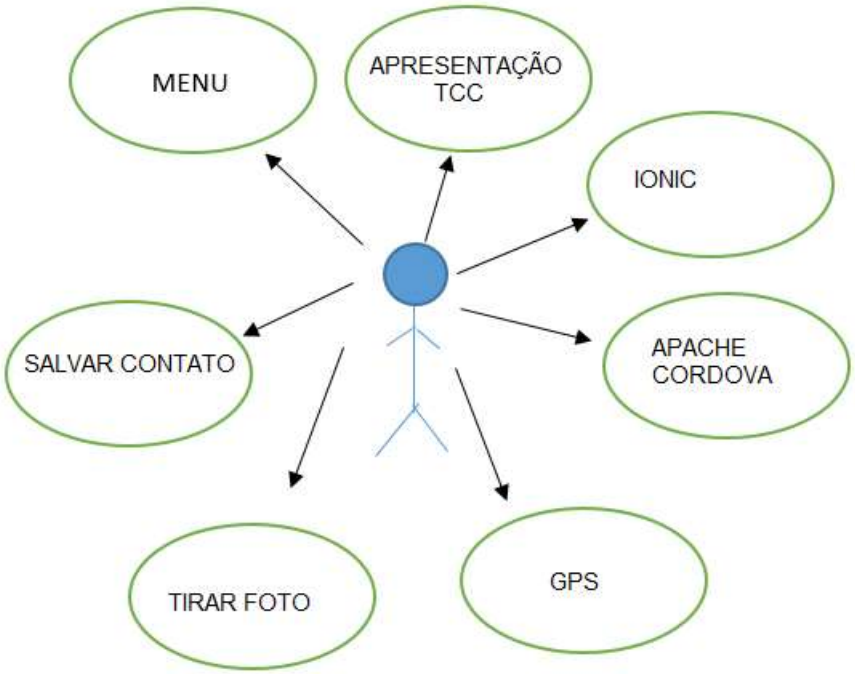


Figura 13 - Ações que usuário poderá exercer dentro da aplicação.

6 DESENVOLVIMENTO DO TRABALHO

Para o desenvolvimento do aplicativo foi utilizado o comando **ionic start tcc2 blank** aonde a plataforma criou o projeto em branco, sem nenhum plugin do Cordova.

O primeiro passo do desenvolvimento foi criar a página de login e registro, aonde foram utilizados os seguintes comandos **ionic –g page login registro**.

Após estes comandos as paginas foram criadas automaticamente pela plataforma, faltando apenas a declaração no **app.module.ts**, aonde todas as páginas e providers que a aplicação conterá, deverão ser declaradas naquele local.

As páginas criadas pela plataforma tem a seguinte estrutura: login.ts, login.scss e a login.html, aonde o arquivo com extensão TS, será utilizado para fazer a programação da back-end da página, enquanto o HTML será utilizado para fazer a front-end da mesma, e no scss é usado para a configuração do layout da página, com estilos em CSS.

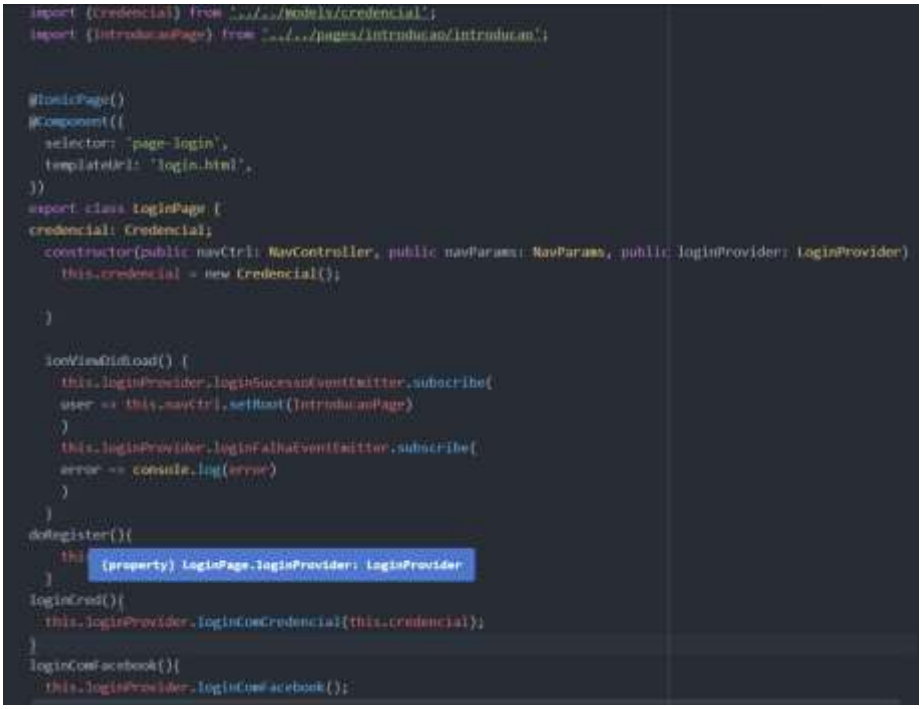


Figura 14 – Configuração da back-end da tela de login

Todas as páginas de navegação da aplicação foram criadas utilizando a própria plataforma, é possível fazer manualmente, desde que o desenvolvedor crie os arquivos em HTML, TS ou JS e o CSS.

Para manter o código-fonte organizado foi criado também provider, que fará a injeção de alguns códigos, a criação do mesmo, também foi utilizando o gerador da plataforma, utilizando o comando **ionic –g provider loginprovider**, a plataforma criou pastas para organização do código e um arquivo chamado login.ts, dentro da pasta login, que fica dentro da pasta providers.

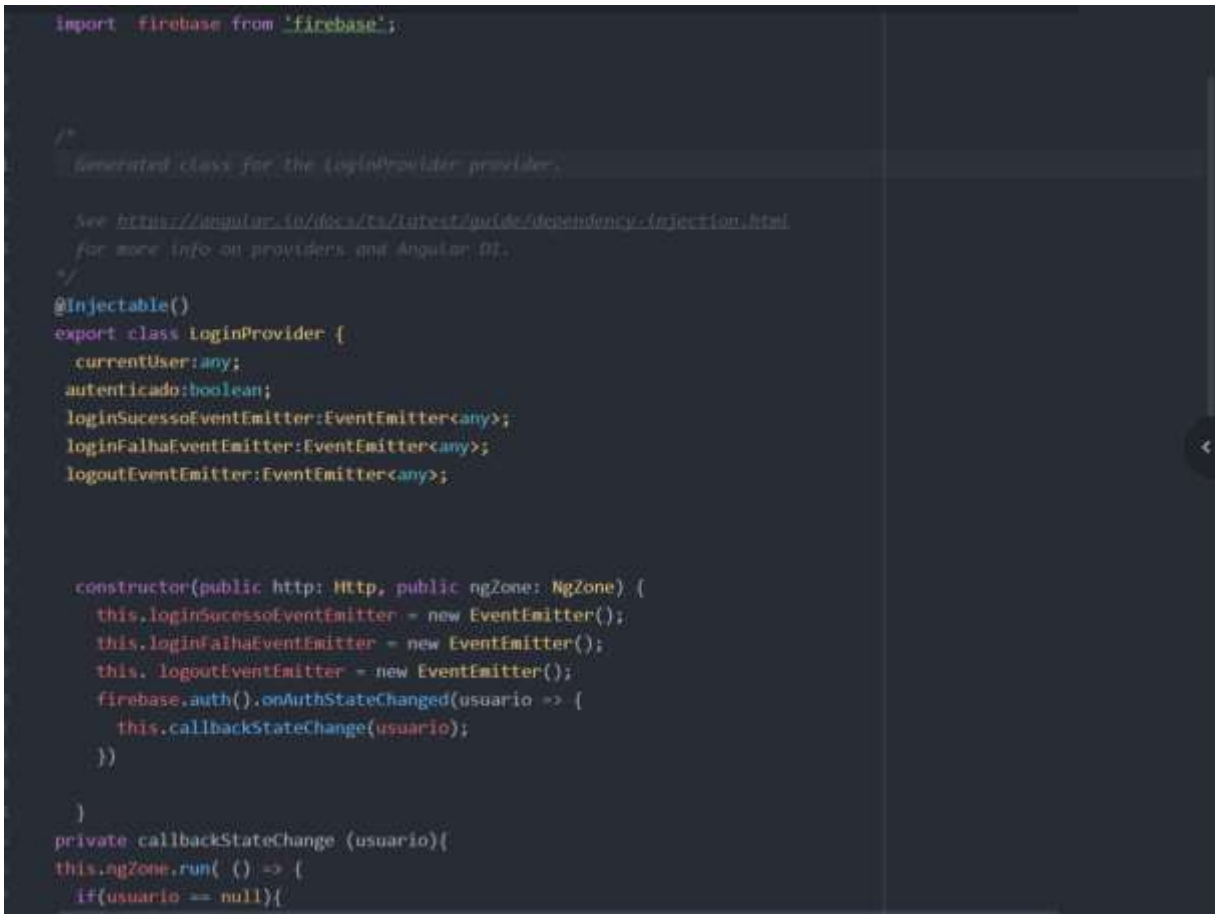


Figura 15 – Provider do login

Dentro do arquivo estão as configurações de login com o banco de dados, no qual foi utilizado o Firebase da Google, foi utilizado também eventos para que seja possível o controle para saber se o usuário está **logado** ou não.

A estrutura da aplicação ficou da seguinte forma:

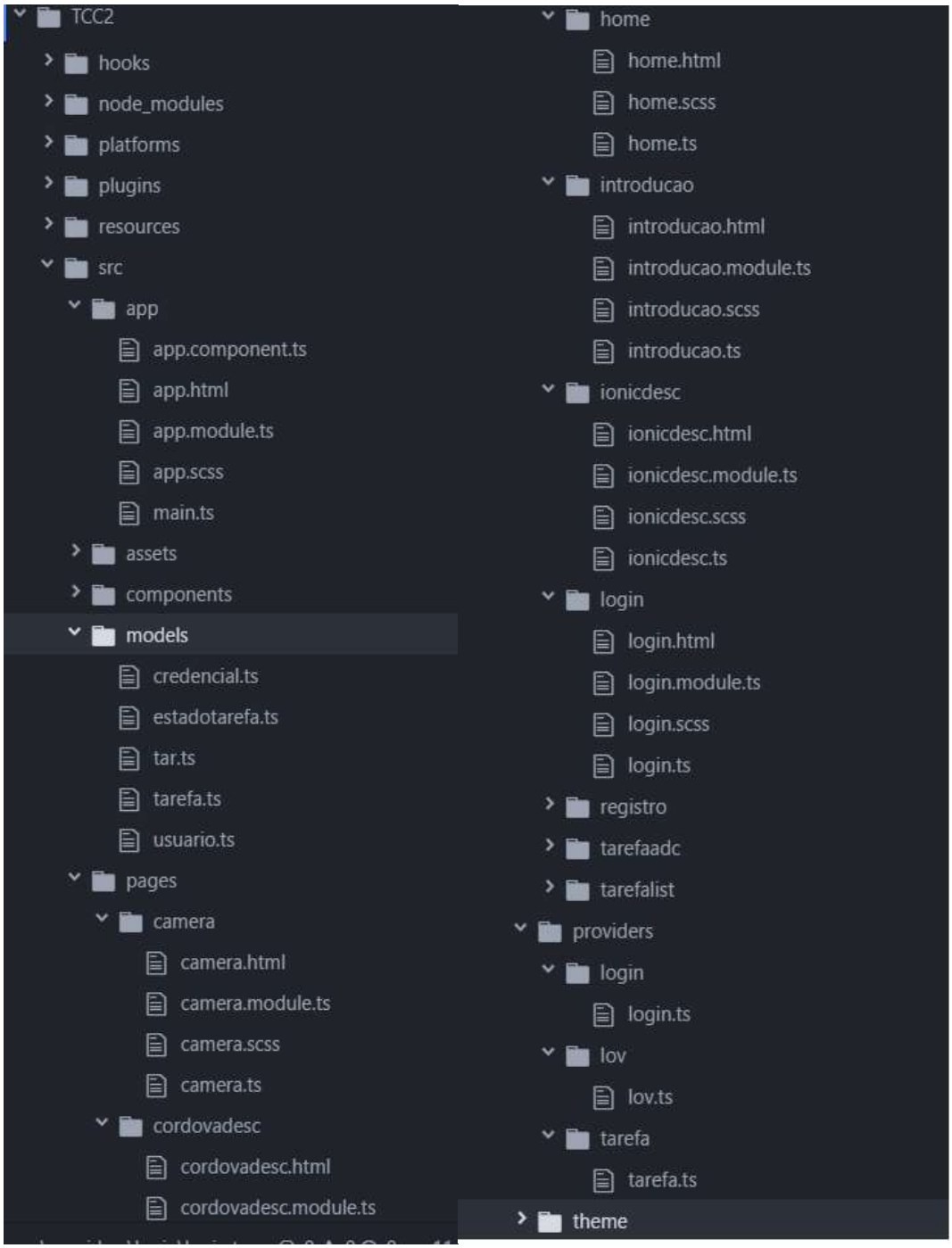


Figura 16 – Estrutura da aplicação, Parte 1

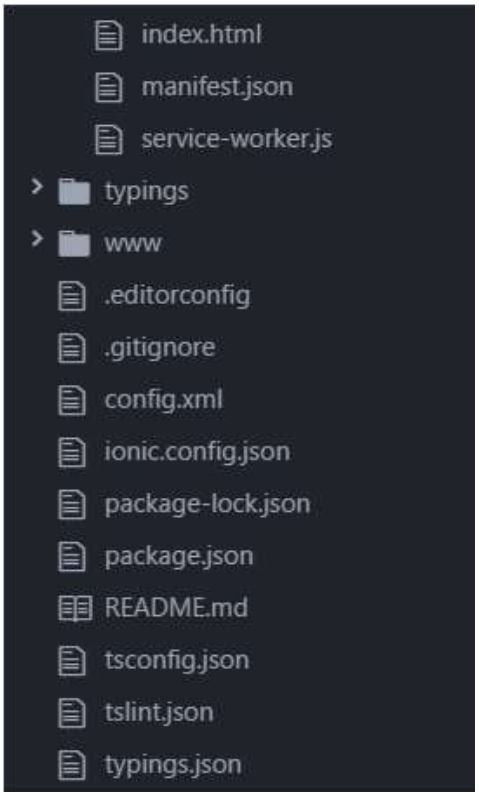


Figura 17 – Estrutura da Aplicação, Parte 2

A aplicação além de contar com a descrição das plataformas utilizadas, tem ainda um gerenciador de tarefas, que tem a conexão com o Firebase, aonde o usuário pode criar tarefas, com estado de Nova, Executando ou Finalizada, que serão listadas.

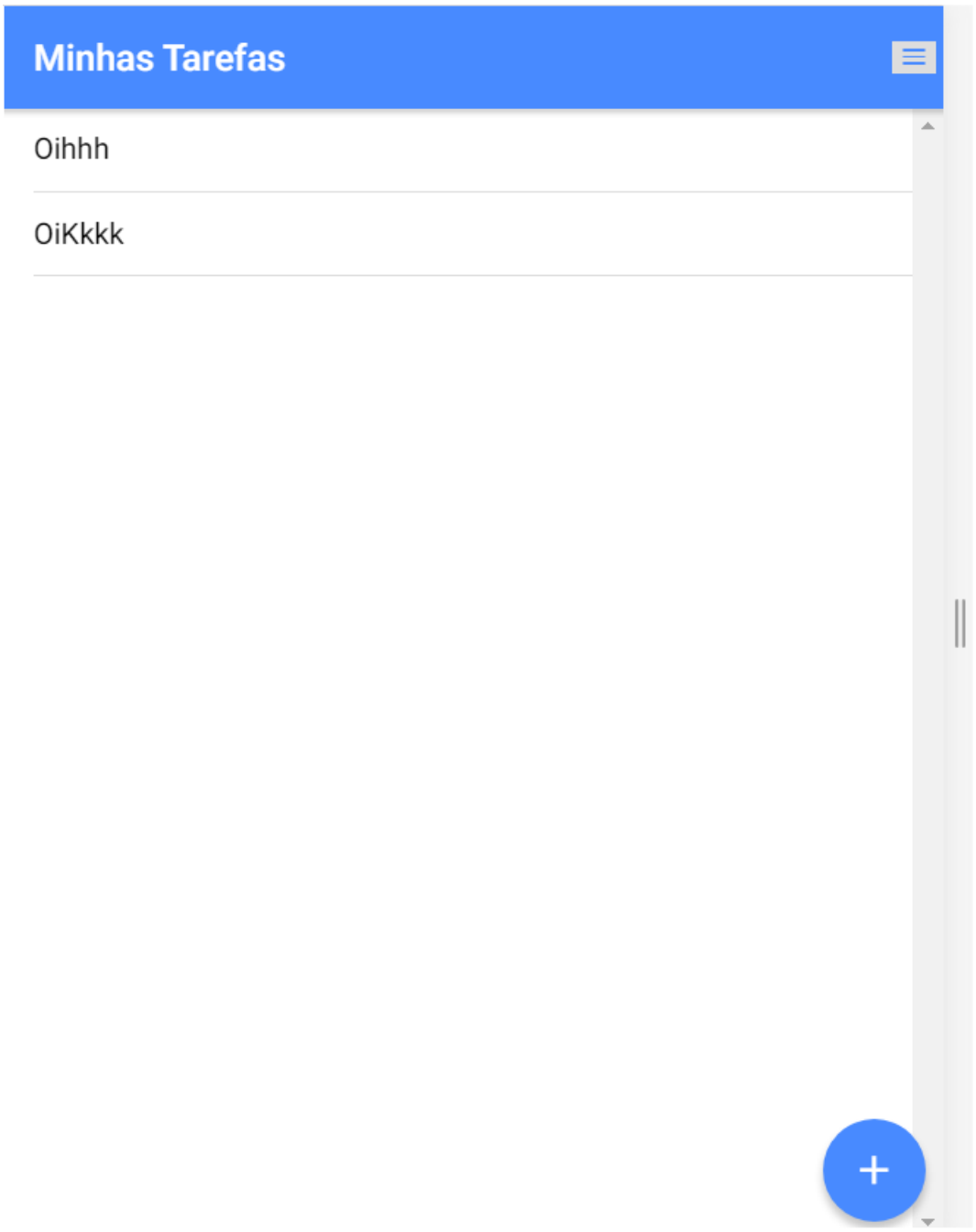


Figura 18 – Tela de listagem das tarefas

Esta é a pagina aonde todas as tarefas criadas são listadas, pressionando o sinal de mais no canto direito inferior, abrirá uma tela aonde será possível a criação de tarefas.

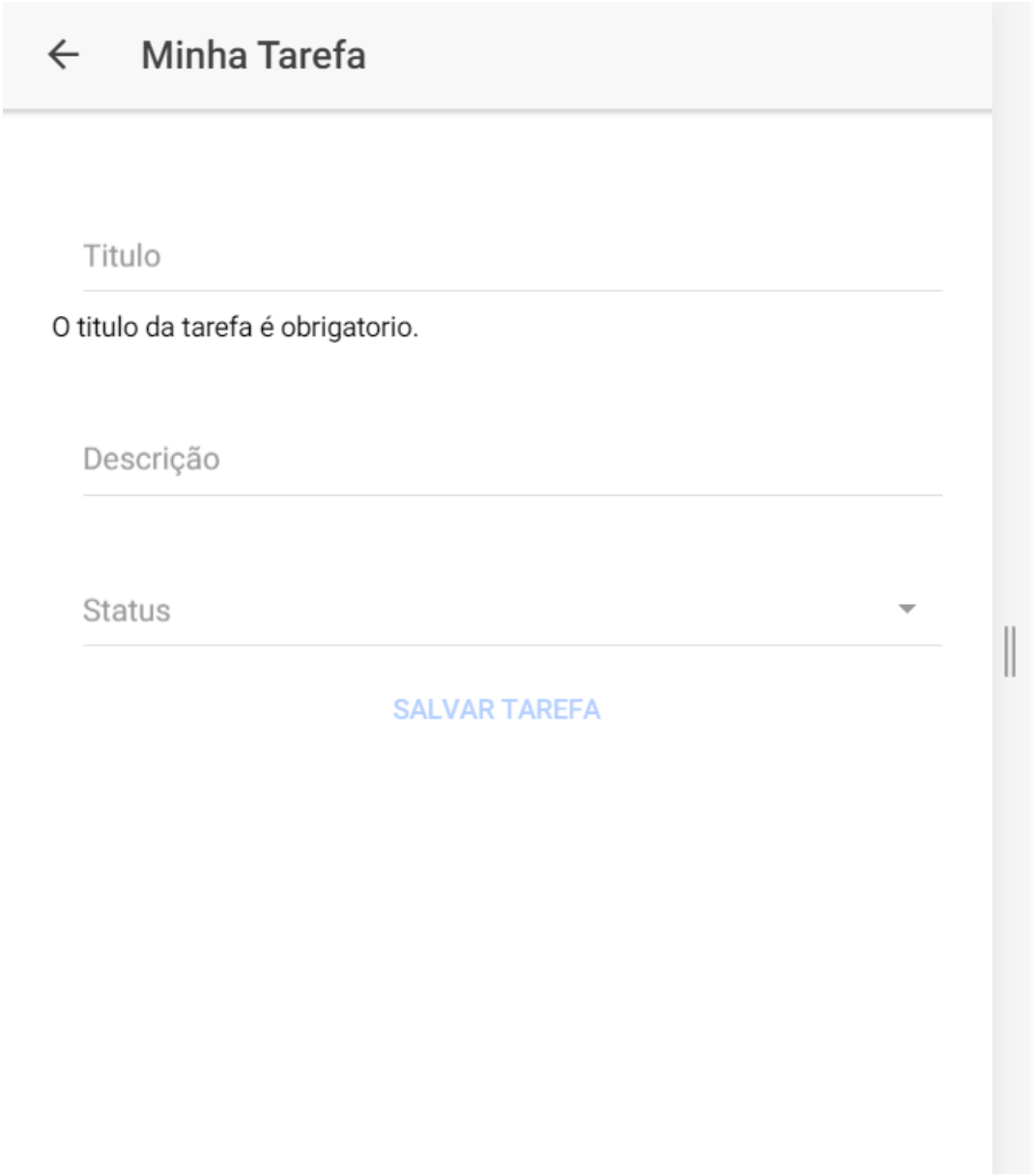


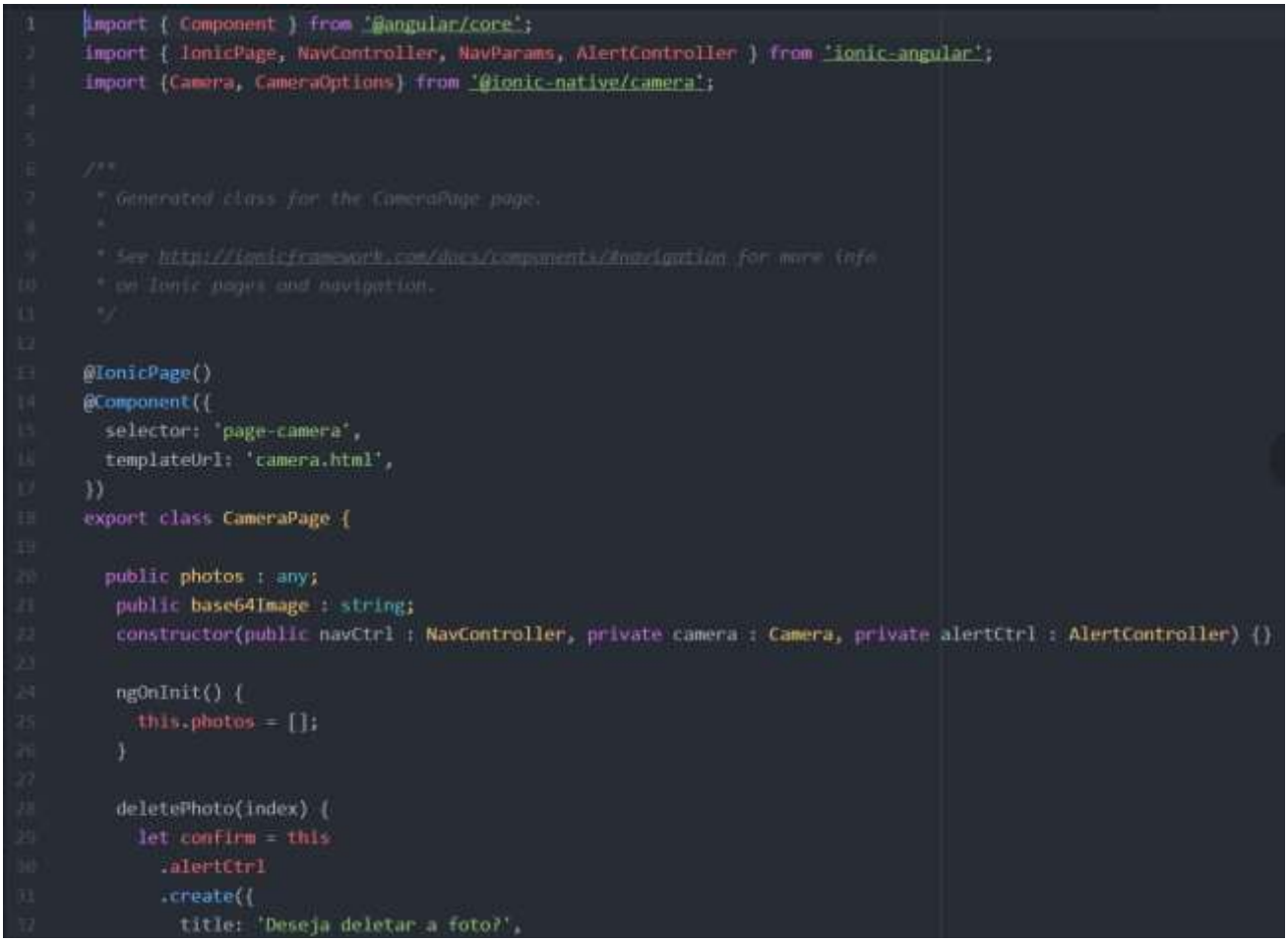
Figura 19 – Tela de criação das tarefas

E para que seja possível a demonstração da interação da aplicação com o hardware do dispositivo, foi criado também uma tela aonde o usuário pode tirar foto, para a criação da mesma foi utilizado o comando **ionic cordova plugin add ios android**, com isso a aplicação terá suporte para o Google Androido e para o Apple iOS.

Também foram utilizados os comandos **ionic cordova plugin add cordova-plugin-camera**, para ter acesso ao plugin da câmera fornecido pelo Apache Cordova, e para

que a plataforma salve dentro da aplicação o comando utilizado foi **npm install --save @ionic-native/camera**.

Após a utilização destes comandos foi necessário só fazer a configuração dentro da página que a câmera será utilizada.



```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams, AlertController } from 'ionic-angular';
import {Camera, CameraOptions} from '@ionic-native/camera';


/**
 * Generated class for the CameraPage page.
 *
 * See http://ionicframework.com/docs/components/#navigation for more info
 * on Ionic pages and navigation.
 */

@IonicPage()
@Component({
  selector: 'page-camera',
  templateUrl: 'camera.html',
})
export class CameraPage {

  public photos : any;
   public base64Image : string;
   constructor(public navCtrl : NavController, private camera : Camera, private alertCtrl : AlertController) {}

   ngOnInit() {
     this.photos = [];
   }

   deletePhoto(index) {
     let confirm = this
       .alertCtrl
       .create({
         title: 'Deseja deletar a foto?',
```

Figura 20 – Tela de configuração back-end da câmera, Parte 1

```
      message: '',
      buttons: [
        {
          text: 'Não',
          handler: () => {
            console.log('Cancelar');
          }
        }, {
          text: 'Sim',
          handler: () => {
            console.log('Aceitar');
            this
              .photos
              .splice(index, 1);
            //return true;
          }
        }
      ]
    });
    confirm.present();
  }

  tirarFoto() {
    const options : CameraOptions = {
      quality: 50,
      destinationType: this.camera.DestinationType.FILE_URI,
      encodingType: this.camera.EncodingType.JPEG,
      mediaType: this.camera.MediaType.PICTURE
    }
    this
      .camera
      .getPicture(options)
```

Figura 21 – Tela de configuração back-end da câmera, Parte 2

```
            }
          }
        ]
      });
    confirm.present();
  }

  tirarFoto() {
    const options : CameraOptions = {
      quality: 50,
      destinationType: this.camera.DestinationType.FILE_URI,
      encodingType: this.camera.EncodingType.JPEG,
      mediaType: this.camera.MediaType.PICTURE
    }
    this
      .camera
      .getPicture(options)
      .then((imageData) => {
        this.base64Image = "file://" + imageData;
        this
          .photos
          .push(this.base64Image);
        this
          .photos
          .reverse();
      }, (err) => {
        console.log(err);
      });
  }

}
```

Figura 23 – Tela de configuração back-end da câmera, Parte 3

```
<ion-header>
<ion-navbar>
  <button menuToggle>
    <ion-icon name="menu" color= "primary"></ion-icon>
  </button>
</ion-navbar>
</ion-header>

<ion-content padding class="card-background-page">
  <button ion-button full (click)='tirarFoto()'><ion-icon name="camera"></ion-icon>  Tirar Foto</button>


  <ion-grid>
    <ion-row>
      <ion-col col-6 *ngFor="let photo of photos; let id = index">
        <ion-card class="releative">

          <ion-icon  ios="ios-add-circle" md="md-add-circle"class="deleteIcon" (click)="deletePhoto(id)"></ion-ico
         <img [src]="photo" *ngIf="photo" />
        </ion-card>
      </ion-col>
    </ion-row>
  </ion-grid>


</ion-content>
```

Figura 24 – Tela de configuração front-end da câmera.

6.1 Finalizando a Aplicação

Após a aplicação ser desenvolvida, para a realização de teste foi utilizado o Ionic View, uma aplicação desenvolvida pela própria Ionic, aonde os desenvolvedores conseguem distribuir sua aplicação, para que outros desenvolvedores ou até mesmo usuários consigam distribuir o seu projeto finalizado.

Para que a aplicação fique disponível no Ionic view, o desenvolvedor precisa fazer o login no Ionic, utilizando o comando ionic login, será pedido o e-mail e senha, feito o login, o desenvolvedor entra na pasta do projeto, ainda pelo terminal, digita o comando ionic upload e a ferramenta começará a fazer o upload da aplicação, que logo estará disponível.

Caso ocorra algum erro no upload, pode ser que a aplicação esteja sem o app_id, para resolver isso basta utilizar o comando ionic link, que a ferramenta irá criar um id para a aplicação desejada.

Esta app_id também serve para que o desenvolvedor passe para qualquer usuário que utilize o Ionic View e assim consiga ter acesso a sua aplicação.

Também é possível distribuir a aplicação de forma direta, como no caso do android, executando o comando ionic cordova build –release android a plataforma irá criar um arquivo na extensão .APK que assim poderá ser distribuído para qualquer dispositivo Android, para a versão do iOS, é necessário ter no computador o xcodebuild.

7 CONCLUSÃO

A plataforma Ionic, junto com o Apache Cordova, cumprem o seu objeto de desenvolvimento híbrido, e por serem ferramentas open source, ainda contam com uma comunidade, aonde é possível encontrar APIs e códigos prontos para serem utilizados de forma gratuita.

Então para o desenvolvimento de uma aplicação, no qual a intenção seja atingir um público maior, e que não se tenha problema financeiro para a execução do projeto, o Desenvolvimento Nativo, ainda continua sendo a melhor opção, pois como o desempenho será melhor, podendo ter diversos recursos sem comprometer a aplicação.

Outro ponto pouco explorado da plataforma, é a questão da integridade dos dados pessoais, dando pouca segurança para que grandes empresas comecem a utilizar a mesma.

O layout da aplicação mesmo com o HTML5 ainda continuam bastante limitados, aonde é possível ver um designer quase parecido com todas as aplicações criadas pela ferramenta.

Porém ainda continua sendo uma ótima solução para desenvolvedores que não possuem conhecimentos em diversas plataformas e que não possuem tempo e dinheiro para a especialização, e levando em consideração que utiliza o HTML5, que é uma linguagem bastante conhecida no meio do desenvolvimento.

O problema maior enfrentado nesta pesquisa foi a falta de material sobre cada plataforma, aonde grande parte da busca de material se esteve presente em apostilas, documentação estrangeiras.

São poucos sites com conteúdo sobre Ionic, e que apresentam uma inicialização rápida e pouco descritiva sobre o assunto, mesmo sendo uma maneira inovadora, e que pretende solucionar o principal problema encontrado por desenvolvedores, aonde diversas empresas fracassaram, ainda é um recurso pouco difundido no Brasil.

7.1 Trabalhos Futuros

Após esta pesquisa pretende-se buscar mais conhecido sobre o assunto, para que seja possível a ampliação dos recursos nativos utilizados na aplicação, e para que esta possa ser um exemplo das funcionalidades providas da plataforma, para que novos desenvolvedores que se interessem pelo assunto consigam aprender sobre o mesmo e ter um exemplo real.

Motivando ainda mais o uso deste recurso, comprovando a sua compatibilidade com diversos sistemas, sua interação com o usuário, interface e o hardware do dispositivo.

REFERÊNCIAS

GRILLO, Rafael, **Introdução ao Ionic Framework**. 2015, 26 de fevereiro, Disponível em: <http://tableless.com.br/introducao-ao-ionic-framework/> Acesso em Outubro, 2016.

PORTO, Tiago, **Aplicativos Híbridos com Ionic.** Disponível em: <http://www.tiagoporto.com/aplicativos-hibridos-com-ionic-voce-tambem-pode-comecar-a-desenvolver-agora> Acesso em Novembro, 2016.

WAGNER, Francis, **Ionic Framework – Part 2.** Disponível em: < http://netcoders.com.br/blog/ionic-framework-parte-2/>, Acesso em Outubro, 2016.

MONTEIRO, Jane Dirce Alves, **Desenvolvimento de Aplicações Multi-plataformas para Dispositivos Móveis.** 2006. Dissertação – Universidade de São Paulo, USP, São Carlos, 2006.

RAVULAVARU, Arvind, **Learning Ionic** – **Build real-time and hybrid mobile applications with Ionic**: Packt Publishing Ltd, Livery Place, 35 Livery Street, Birmingham B3 2PB, UK, 2015.

LIMA, Victor, **Nativo x Híbrido – A Discussão FINAL (PARTE 1)**, 2016, 03 de março. Disponível em: < http://www.concretesolutions.com.br/2016/03/16/nativo-x-hibrido/> Acesso em Novembro, 2016.

AMBROS, Luisa, **Diferença entre Aplicativos Nativos, Híbridos e Mobile Web Apps,** 2013, 24 de setembro. Disponível em: < http://www.luisaambros.com/blog/diferenca-entre-aplicativos-nativos-hibridos-e-mobile-web-apps/> Acesso em Novembro, 2016.

ROCHA, Adriano Mendonça; NETO, Roberto Mendes Finzi, **Introdução à Arquitetura Apple iOS**, 2011, Artigo, Departamento de Ciência da Computação – Universidade Federal de Goiás (UFG).

BORDIM, Maycon Viana, **Introdução a Arquitetura Android**, 2012, Disponível em: < http://sites.setrem.com.br/stin/2012/anais/Maycon.pdf >, Acesso em Fevereiro, 2017.

RUBAI, Eduardo Laguna; BONETTI, Tiago Piperno, **Desenvolvimento Web Dentro dos Paradigmas do HTML5 e CSS3**, 2015, Disponível em: < http://web.unipar.br/~seinpar/2015/_include/artigos/Eduardo_Laguna_Rubai.pdf >, Acesso em Fevereiro, 2017.

VASCONCELLOS, Luis, **Apps Híbridas com Cordova e Ionic #1**, 2015, Disponível em: < http://luisvasconcellos.com/2015/04/06/apps-hibridas-com-cordova-e-ionic.html >, Acesso em Fevereiro, 2017.

Mobile HTML5 de Estelle Weyl O'Reilly, Copyright 2014 Estelle Weyl, 978-1-449-31141-4

KAMADA, Terumi P. B., ISHIDA, Celso Yoshikazu, GOMES, Márcio Luiz, CARPEJANI, Jayson, **Análise das Plataformas de Desenvolvimento Mobile aplicados na Área Educacional, usando Android e Windows Phone**, 2012,

Disponível em: < http://seer.ufrgs.br/renote/article/viewFile/30916/19896 >, Acesso em Março, 2017.

CRIS, Robison, **Desenvolvimento Android utilizando a IDE Android Studio,** Disponível em: < http://www.devmedia.com.br/desenvolvimento-android-utilizando-a-ide-android-studio/33872 >, Acesso em Março, 2017.

BRAWERMAN, **Alessandro, Xcode iOS: Introdução ao desenvolvimento para iOS**, Disponível em: < http://www.devmedia.com.br/xcode-ios-introducao-ao-desenvolvimento-para-ios/29987 >, Acesso em Março, 2017.

Documentation – Apache Cordova, Disponível em: **<** https://cordova.apache.org/docs/en/latest/ >, Acesso em: Janeiro, 2017.

GUGIK, Gabriel, **A história dos computadores e da computação**, 2009, Disponível em: < https://www.tecmundo.com.br/tecnologia-da-informacao/1697-a-historia-dos-computadores-e-da-computacao.htm >, Acesso em: Janeiro, 2017.