UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MURIEL RAMPINELLI MAZUCHETTI

ESTUDO DE FRAMEWORK MULTIPLATAFORMA APLICADO AO DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MOBILE HÍBRIDO PARA O CONTROLE FINANCEIRO PESSOAL

CRICIÚMA

2015

MURIEL RAMPINELLI MAZUCHETTI

ESTUDO DE FRAMEWORK MULTIPLATAFORMA APLICADO AO DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MOBILE HÍBRIDO PARA O CONTROLE FINANCEIRO PESSOAL

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Gilberto Vieira da Silva

CRICIÚMA

2015

MURIEL RAMPINELLI MAZUCHETTI

ESTUDO DE FRAMEWORK MULTIPLATAFORMA APLICADO AO DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MOBILE HÍBRIDO PARA O CONTROLE FINANCEIRO PESSOAL

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Dispositivos Móveis

Criciúma, 22 de Junho de 2015

BANCA EXAMINADORA

Prof. Gilberto Vieira da Silva - Esp. - UNESC - Orientador

Prof. Luciano Antunes - MSc. - UNESC

Prof. Fabricio Giordani - Esp. - UNESC

Dedico este trabalho aos meus pais, que sempre foram minha fonte de inspiração e motivação, estrutura da minha vida.

AGRADECIMENTOS

Primeiramente agradeço a Deus, que sempre me guiou e ajudou a trilhar o caminho correto para alcançar meus objetivos dando também a sabedoria necessária para concluir mais esta etapa de vida.

Aos meus pais Mario Luiz Mazuchetti e Adelaide Maria Rampinelli Mazuchetti que sempre me auxiliaram nos momentos difíceis e nunca me deixaram desistir. Meu pai sempre me inspirando e me ensinando que mesmo com as dificuldades é muito importante sorrir e nunca desistir dos objetivos por mais difíceis que sejam. A minha mãe que sempre me ajudou e passou muitas noites acordadas comigo auxiliando e sempre me mostrando que é valido lutar por algo que se deseja. A eles todo o meu amor e agradecimento especial.

Um agradecimento especial para o meu amigo Naidion Brovedan. Este grande irmão que demonstrou que amigos são para horas boas e ruins e que sempre me mostrou que quando se deseja algo na vida é necessário lutar por isto. Ele também não mediu esforços para me auxiliar quando necessário. A ele minha eterna gratidão.

Ao Guilherme Tramontin, este grande irmão que não é de sangue mais é quase. Sempre me mostrou e me deu esporros nas horas que tudo parecia perdido, nunca me deixou abandonar o projeto e sempre me incentivava a dar o meu melhor até o final. Muito obrigado por esta parceria que essa amizade seja eterna.

Um agradecimento também a Carla Macarini, está amiga que me fez sorrir em vários momentos e nem que fosse para brigar comigo, sempre me fez acreditar que eu era capaz de conseguir concluir mais uma etapa de vida. Muito obrigado.

Aos meus amigos Franco, Fernando, Kainã e Natan, que sempre acreditaram no meu potencial e sempre me disseram que seria fácil concluir essa fase uma vez que eles acreditavam em meu potencial. A eles toda a minha gratidão e que essa parceria continue pro resto das nossas vidas.

As minhas amigas Leticia e Renata que me aconselharam e me ajudaram a trilhar corretamente nesta etapa sempre acreditando que eu era capaz de alcançar meu objetivo. Muito obrigado.

Aos meus amigos em geral que me deram forças e auxilio de forma direta ou indireta, sou muito grato a vocês.

Um agradecimento a todos os professores do curso que nunca mediram esforços para nos guiar e passar seus conhecimentos. Em especial ao professor orientador Gilberto Vieira da Silva que me acompanhou nesta etapa e sempre me auxiliou da melhor maneira para que eu obtivesse sucesso.

“A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro.”

Albert Einstein

RESUMO

A tecnologia, principalmente à área de dispositivos móveis, teve um alto marco de evolução e tornou-se muito popular atualmente. Alguns aparelhos, como por exemplo, celulares, smartphones e tablets estão cada vez mais presentes no cotidiano dos indivíduos sendo que a capacidade de processamento destes está aumentando frequentemente. As plataformas móveis disponibilizam vários recursos que possibilitam que os desenvolvedores criem novas funcionalidades. Com a necessidade de uma mesma aplicação ser utilizada para mais de uma plataforma, originou-se, as aplicações multiplataformas. Esse novo conceito gerou benefícios a empresas e desenvolvedores, uma vez que não seria necessário reescrever o código fonte. Baseado nisso, foram sendo criadas ferramentas que auxiliassem no desenvolvimento, tais como os *frameworks* multiplataforma. Porém, percebeu-se que as aplicações não tinham a mesma performance que as nativas. Baseado nisto, o presente trabalho apresenta o desenvolvimento de um protótipo de aplicação híbrida para o controle financeiro, utilizando o *framework* Ionic. Para isso foi realizado um estudo sobre os *frameworks* multiplataforma, dando maior ênfase no *framework* Ionic, onde foi possível verificar os motivos pelo qual ele se sobressai perante os demais. Optou-se por uma aplicação para a área de finanças, pois utilizar recursos para controlar a movimentação financeira é um meio muito procurado pelos indivíduos.

Palavras-Chave: Dispositivos móveis, Aplicações híbridas, *Frameworks* multiplataforma, Controle de finanças pessoais.

ABSTRACT

Technology, especially the area of mobile devices, had a high mark of evolution and has become very popular today. Some devices, such as mobile phones, smartphones and tablets are increasingly present in daily life and the processing capabilities of these is increasing frequently. Mobile platforms provide several features that enable developers to create new features. With the need of the same application is used for more than one platform, originated from the multi-platform applications. This new concept has generated benefits to businesses and developers, since it would not be necessary to rewrite the source code. Based on this, they were created tools that would help in the development, such as cross-platform frameworks. However, it was noticed that the applications had not the same performance as native. Based on this, this paper presents the development of a hybrid prototype application for financial control, using the Ionic framework. For this was a study on the multi-platform frameworks, placing greater emphasis on Ionic framework, it was possible to verify the reasons for which he stands before the other. We chose an application to the area of finance, for use resources to control the financial transaction is a means much sought after by individuals.

Keywords: Mobile Devices, Hybrid Apps, Cross-platform *Frameworks*, Personal Financial Control.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	Application Program Interfaces
ART	Android Runtime
CLI	Command Line Ionic
CRUD	Create Ready Update Delete
MVC	Model View Controller
MVW	Model View Whatever
NDK	Native Development Toolkit
NPM	Node Package Manager
OHA	Open Handset Alliance
SDK	Software Development Kit
SSAS	Syntactically Awesome Stylesheets Sass
UI	User Interface

SUMÁRIO

1 INTRODUÇÃO

As plataformas móveis sempre tiverem uma visão ampla de crescimento e estão se tornando cada vez mais populares. O uso de celulares, tablets e smartphones conquistou grande parte das pessoas e nota-se que estão mais presentes no cotidiano dos indivíduos, sendo que a capacidade de processamento destes está sempre em expansão.

As mais populares são Android e iOS e foram as principais responsáveis pela maioria dos embarques de smartphones no terceiro trimestre de 2014. O Android lidera com 84% da quota do mercado contra 11,7% do iOS. Elas oferecem ainda uma vasta quantidade de benefícios, possibilitando que seja possível tirar total proveito de um aparelho portátil (TELECO, 2014).

Os aplicativos para dispositivos móveis estão em constante crescimento. A cada dia é possível perceber o surgimento de novidades nas lojas virtuais de smartphones. É comum que existam dúvidas na escolha de um meio adequado para desenvolver as aplicações móveis, umas vez que cada plataforma fornece uma ferramenta nativa para este proposito (CAVAZZA, 2011, tradução nossa).

Atualmente existem dois tipos de aplicações mobile, as nativas e as multiplataformas. As nativas são criadas exclusivamente para uma determinada plataforma utilizando linguagem de programação homologada pela mesma, já as multiplataformas podem ser executadas por qualquer plataforma mobile, além disto, esses tipos de aplicações são subdivididas em *webapps* que são desenvolvidas com linguagem *web* e executadas através do *browser* totalmente dependente da internet e as aplicações híbridas que contem a versatilidade das aplicações *web*, basicamente uma instancia do *browser*, que é executada compilada e interpretada como uma aplicação nativa (DEVMEDIA, 2015).

Com esse avanço tecnológico originaram-se os *frameworks* para a criação de aplicações híbridas. É possível citar como exemplo, *Sencha, PhoneGap, Titanium, Rhomobile, ParticleCode* e entre outros. O uso desses recursos acaba propiciando vantagens como, por exemplo, redução no tempo de desenvolvimento, maior custo benefício, boa experiência de usuário, facilidade de manutenção. Porém, de contraposto notava-se que os aplicativos híbridos eram relativamente lentos comparados aos nativos (CAVAZZA, 2011, tradução nossa).

Segundo Wilken (2015), com o surgimento do AngularJs, uma poderosa ferramenta JavaScript mantida pela Google, e pensando na performance das aplicações hibridas, em 2013 a empresa norte americana Drifty lançou o *framework* Ionic.

Ele é utilizado especialmente para projetar a interface do usuário. Desta forma, inclui-se todos os elementos visuais, como guias, botões, cabeçalhos de navegação, pop-ups entre outros (IONIC, 2015, tradução nossa).

Devido sua composição ser HTML, CSS e AngularJS, permite que suas aplicações tenham código fonte único, extinguindo, desta forma, a necessidade da utilização de um algoritmo para cada plataforma (WILKEN, 2015, tradução nossa).

Para ajudar no desenvolvimento das aplicações, o Ionic disponibiliza ainda uma ferramenta chamada Linha de Comando do Ionic do inglês *Command Line Ionic* (CLI), que gera os projetos iniciais, possibilita a pré-visualização, a criação e implantação do aplicativo (IONIC, 2015, tradução nossa).

No contexto atual, o seguinte trabalho pretende abordar um estudo acerca de *frameworks* multiplataforma *mobile*, realizando o desenvolvimento de uma aplicação híbrida para o controle financeiro pessoal utilizando o *framework* Ionic, expondo também os motivos pelo qual ele foi escolhido.

Optou-se um aplicativo para a área financeira, pois alguns especialistas garantem que buscar recursos para controlar as finanças é um meio utilizado por muitas pessoas que pretendem adquirir uma margem de tranquilidade financeira (ROCHA; SOUSA; TORRALVO, 2012).

1.1 OBJETIVO GERAL

Desenvolver um protótipo de aplicativo mobile híbrido para o controle financeiro pessoal utilizando o *framework* multiplataforma Ionic.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho consistem em:

a) verificar os *frameworks* para desenvolvimento multiplataforma;

b) compreender a diferença entre os tipos de aplicações móveis;

c) desenvolver um protótipo de um aplicativo híbrido;

d) estudar *framework* Ionic;

e) compreender a importância da utilização de softwares financeiros

1.3 JUSTIFICATIVA

Com o crescimento tecnológico, é cada vez mais comum que os smartphones estejam presentes no cotidiano das pessoas. As plataformas móveis acompanham frequentemente essa evolução e são as principais responsáveis pelo sucesso dos smartphones e tablets. É possível notar que a cada dia surgem novos aparelhos, e que cada vez mais os indivíduos acabam trocando seus dispositivos por outro mais moderno. Juntamente com os smartphones se obtém um alto índice de crescimento nas aplicações para dispositivos móveis (TELECO, 2014).

Diante deste crescimento, os desenvolvedores e as empresas buscam cada vez mais formas para reduzir manutenções e custos em suas aplicações uma vez que cada plataforma móvel possui uma linguagem nativa própria para o desenvolvimento, sendo assim não se tem um aproveitamento de código fonte obrigando os desenvolvedores a reescreverem o algoritmo das aplicações para que a mesma esteja disponível para mais de uma plataforma.

Atualmente existem as aplicações multiplataformas que são desenvolvidos uma única vez e pode ser utilizados para diversas plataformas. Devido ao constante crescimento dos aparelhos de smartphones é interessante que as empresas utilizem o desenvolvimento multiplataforma, uma vez que os usuários acabam familiarizando-se com as aplicações (WILKEN, 2015, tradução nossa).

A realização de um estudo baseado em *frameworks* multiplataforma *mobile*, se faz interessante, uma vez que estas ferramentas englobam novas tecnologias, como por exemplo, os novos conceitos de linguagem web que são HTML5, CSS3 e Javascript. Além deste ponto, cabe ressaltar que o custo benefício de desenvolver o código fonte uma única vez e promover a portabilidade entre plataformas é maior, contando ainda com a redução no tempo de desenvolvimento, e de manutenção, sendo que a linha de aprendizado torna-se exclusiva a uma única linguagem (ZIFLAJ, 2014, tradução nossa).

Um ponto importante a ser abordado, é que nos primórdios das aplicações híbridas, notava-se um queda no desempenho quando comparada as

aplicações nativas. Este fato levantou uma questão importante, que seria qual dos *framework* existentes é mais ideal para o desenvolvimento de uma aplicação tratando-se do fator performance.

De acordo com Wilken (2015), com o surgimento do AngularJS o conceito de interação existente entre o HTML e o Javascript, teve uma mudança significativa. Baseado nesta nova tecnologia, foi criado o *framework* Ionic, sua estruturação era semelhante aos demais, porém, seu diferencial perante os outros, falando em termos de tecnologia, é o Angularjs e seu CSS que utiliza a tecnologia *Syntactically Awesome Stylesheets Sass* (SASS), que é uma forma simplificada do fluxo do CSS, e o mesmo pode ser sobrescrito, tornando fácil personalização de interface das aplicações, possibilitando a criação de interfaces mais dinâmicas (IONIC, 2015, tradução nossa).

Com o intuito de desenvolver um protótipo de aplicação híbrida para o controle financeiro pessoal, pretende-se utilizar o *framework* Ionic, a fim de demonstrar o porque ele é viável para a criação de aplicações mobile híbridas.

A criação do protótipo de aplicativo para as finanças tem também o objetivo de contribuir para que as pessoas possam obter um planejamento sobre seus gastos uma vez que o smartphone é utilizado cada vez mais, agilizando assim, o controle financeiro

1.4 ESTRUTURA DO TRABALHO

O referente trabalho é composto por oito capítulos, sendo que o primeiro faz uma apresentação do tema proposto, seguido do objetivo geral, objetivos específicos, justificativa e estrutura do mesmo.

No segundo capitulo há são apresentadas as duas plataformas móveis mais populares atualmente.

O terceiro capítulo aborda a respeito das aplicações móveis, dando ênfase aos tipos existentes, destacando um subcapitulo referente aos frameworks multiplataforma.

O quarto capítulo representa o estudo baseado no *framework* Ionic, abordando as ferramentas que o compõe e seus recursos.

O quinto capítulo faz uma breve apresentação acerca dos motivos pela escolha de uma aplicação para finanças pessoais

O sexto capítulo representa os trabalhos correlatos ao estudo desenvolvido.

O sétimo capítulo demonstra toda a estrutura do trabalho que foi realizado para a criação do aplicativo, onde estão abordados os detalhes do desenvolvimento e onde foram realizados os resultados obtidos com o trabalho.

O oitavo e último capítulo representa a conclusão referente ao trabalho que foi realizado.

2 PLATAFORMAS MÓVEIS

O crescimento tecnológico atualmente é constante. Cada vez mais, é possível notar mudanças no mercado de eletrônicos, porém, pode-se ressaltar que a área de dispositivos móveis foi a que mais evoluiu. Atualmente, estes dispositivos possuem um sistema operacional, uma linguagem de programação próprio e ferramentas para o desenvolvimento (TELECO, 2014).

As plataformas móveis foram criadas, por um conjunto de tecnologias, que agregam vários recursos. Com isso, os smartphones passaram a utilizar plataformas para o funcionamento. Essas plataformas tem a responsabilidade em gerenciar todas as funções de hardware de um dispositivo móvel, como gps, câmera, agenda, lista telefônica, entre outros recursos. Cabe ressaltar ainda, que elas são a principal forma de interação entre usuário e aplicações (ALLEN; GRAUPERA; LUNDRIGAN, 2012).

Considerando a época de criação dos telefones móveis é possível classificá-los da seguinte forma: *feature phones* ou celulares, smartphones e tablets.

Os primeiros celulares possuíam poucos recursos comparados aos smartphones. Muitos destes não possuem nem acesso a câmera, muito menos a internet. Seu sistema operacional naturalmente era o *Sybiam* ou *J2ME*. Os aplicativos para celulares eram criados com Java (TELECO, 2014).

Logo após o surgimento dos smartphones, foi possível notar a diferença entre as tecnologias, uma vez que o smartphone mudou totalmente a percepção que se tinha a respeito da telefonia móvel, com as funcionalidades como, *bluetooth*, redes 4g, reconhecimento facial, reconhecimento biométrico entre outros (TELECO, 2014).

Os tablets foram criados logo em seguida dos smartphones, e tem basicamente as mesmas funcionalidades, porém, disponibilizam *displays* maiores e funcionam basicamente como um *palmtop* (TELECO, 2014)

De acordo com a Teleco (2014), as plataformas móveis acompanharam o crescimento dos smartphones, e estão cada vez mais completas. Entre as várias existentes destacam-se Android da Google, e o iOS da Apple. Além disto, foram as principais responsáveis pela grande maioria dos embarques de smartphones no terceiro trimestre de 2014. O Android lidera com 84% da quota do mercado contra 11,7% do iOS.

Cada uma delas possui suas próprias características que são responsáveis por torna-las interessantes, não apenas para os consumidores, mas também, para os desenvolvedores. Naturalmente os aparelhos mais fáceis de utilizar e com diversos funcionalidades e com maior custo-benefício são a principal busca realizada pelos consumidores. Os desenvolvedores, por outro lado, procuram por smartphones ou tables, onde a plataforma tem maior reconhecimento de mercado e que tenha uma boa documentação, baixo custo de desenvolvimento entre outros (ALLEN; GRAUPERA; LUNDRIGAN, 2012).

Juntamente com os dispositivos e as plataformas móveis, pode-se notar uma alta gama de aplicações sendo criadas e adicionadas nas *app stores*. Essas lojas virtuais são responsáveis distribuir e vender aplicações. Cada plataforma possui sua linguagem de desenvolvimento própria que permite a criação destas aplicações e consequentemente a publicação nas lojas virtuais.

2.1 PLATAFORMA ANDROID

Com o decorrer do tempo às plataformas moveis vem evoluindo constantemente. Um dos grandes destaques entre essas plataformas é o Android que se tornou muito popular devido à facilidade de uso e por estar presente na maioria dos smartphones e tablets (ALLEN; GRAUPERA; LUNDRIGAN, 2012).

Os especialistas Pereira e Silva (2009), afirmam que a plataforma Android é muito robusta, tendo destaque entre as demais já que é constituída de um sistema operacional, de um *middleware* que é o software responsável por fazer a comunicação entre o sistema operacional e as aplicações, facilitando o desenvolvimento e a integração das aplicações. Também é composta por uma interface de usuários e de vários aplicativos.

Um dos recursos mais interessantes do Android é a possibilidade do desenvolvimento de aplicativos através do *Software Development Kit* (SDK). Essa plataforma teve início devido a uma parceria entre Google e *Open Handset Alliance* (OHA). A OHA é composta de várias empresas que estão envolvidas com tecnologia móvel, é possível citar (OPEN HANDSET ALLIANCE, 2014, tradução nossa):

a) empresas de software;
b) fabricantes de celulares;
c) empresa de semicondutores;

d) operadores de celular;

e) empresas de comercialização.

Além de ser uma plataforma interativa e possuir muita flexibilidade arquitetural dando possibilidade de integrar as aplicações nativas da plataforma com as aplicações criadas pelos desenvolvedores, o Android é baseado no sistema operacional Linux, sendo composto por um grupo de ferramentas que atuam em todas as fases do desenvolvimento de um projeto, desde a execução até a criação especifica de softwares. Sua base em Linux permite que o sistema operacional faça o gerenciamento de memória e cuide dos processos, permitindo que várias aplicações possam ser abertas ao mesmo tempo. Quando uma aplicação é executada a outra aplicação fica em segundo plano, ou seja, em espera (LECHETA, 2010) .

Um fator muito importante ressaltado por Pereira e Silva (2009), é que o sistema Android é constituído de uma plataforma de código aberto, possibilitando ao desenvolvedor utilizar qualquer recurso disponível no dispositivo móvel. O *open-source* ou código aberto tem uma grande contribuição para o aperfeiçoamento da plataforma Android já que permite aos desenvolvedores do mundo inteiro a criação de novas funcionalidades para a plataforma. A figura 1 ilustra a relação entre o Android e o dispositivo móvel ou hardware no qual ele é executado.

Figura 1 – Ilustração Android e hardware

Fonte: Pereira e Silva (2009)

O mercado global de aplicativos para celulares foi revolucionado pela plataforma Android, lembrando que a mesma foi a primeira de código fonte aberto. Ela cresceu em poucos anos de forma impressionante causando uma grande mudança no mercado mobile, e ganhou respeito diante de seus colegas de indústria (ABLESON et al, 2012).

Em meados de 2005, a Google realizou a compra da empresa Android Inc., que era responsável pelo desenvolvimento do sistema operacional dos dispositivos mobile com base em *Kernel Linux*. Baseado nesse sistema foi realizado o desenvolvimento do projeto Android pela OHA, sendo que seu código *open-source* é passível de alterações por todo e qualquer desenvolvedor de software. A partir desse período o Android tornou-se uma tecnologia de ponta que está sempre em constante crescimento (GARGENTA, 2011, tradução nossa; OPEN HANDSET ALLIANCE, 2014, tradução nossa).

2.2 PLATAFORMA IOS

A empresa *Apple* Inc. criou em 2007 o sistema operacional para seus dispositivos móveis chamada iOS. Conhecido anteriormente por iPhone OS teve seu nome alterado devido a outros dispositivos da *Apple* o adotarem como sistema operacional, como por exemplo, o *iPad, iPod Touch* e *AppleTV*. De forma diferencial, a *Apple* não licencia seu sistema operacional para que o mesmo seja instalado em outros dispositivos que não fazem parte de seu quadro de equipamentos (NEUBURG, 2014, tradução nossa).

De acordo com Lecheta (2012), quando foi criado, o iOS foi composto por uma tecnologia que tinha como principal característica sua *User Interface* (UI) ou interface com usuário em português, que possibilitava múltiplos toques (*multi-touch*), permitindo assim, que houvesse uma experiência diferenciada de combinações de gestos em tela. Ressalta também, que outro ponto forte e da plataforma é a forma com que é feita a integração com seu hardware. Devido a sua exclusividade para dispositivos Apple garante um grande desempenho em todos os quesitos (LECHETA, 2012).

Da mesma forma que o Android da Google, a plataforma da Apple ganha destaque entre as outras plataformas, justamente por estar entre as mais escolhidas por desenvolvedores e usuários. A preferência se inicia pelo lado do usuário, que

pode usufruir de mais de um milhão de aplicativos que estão disponíveis na *Apple Store* (LECHETA, 2012).

Existem várias ferramentas que auxiliam e possibilitam a criação de novas *apps*. Inicialmente ele dispõe de um SDK, que é um poderoso conjunto de ferramentas que são fornecidos pela própria Apple para facilitar o trabalho dos desenvolvedores. Além disto, ainda conta com várias aplicações, entre elas o *Xcode*, iOS *Simulator* e o *Instruments* (APPLE, 2015, tradução nossa). A figura 2 demonstra a arquitetura do sistema operacional iOS.

Figura 2 – Arquitetura do sistema operacional iOS

Fonte: Apple (2015)

3 APLICAÇÕES MÓVEIS

Da mesma maneira que as plataformas, os aplicativos para dispositivos móveis estão em constante crescimento. A cada dia é possível perceber o surgimento de novidades nas lojas virtuais de smartphones. É comum que muitos programadores tenham dúvida na escolha de um meio adequado para desenvolver suas aplicações móveis, já que cada plataforma fornece uma ferramenta nativa para o desenvolvimento (DEVMEDIA, 2015).

Diante deste crescimento, os desenvolvedores e as empresas buscam cada vez mais formas para reduzir manutenções e custos em suas aplicações uma vez que cada plataforma móvel possui uma linguagem de programação distinta, sendo assim não se tem um aproveitamento de código fonte obrigando os desenvolvedores a reescreverem o algoritmo das aplicações para que a mesma esteja disponível para mais de uma plataforma (DEVMEDIA, 2015)

Atualmente existem três tipos de aplicações, são elas nativas, *webapps* ou *mobile web sites* e, híbridas. Os subcapítulos a seguir irão descrever um pouco melhor cada uma delas.

A figura 3 faz um comparativo a respeito dos três tipos de aplicações existentes demonstrando a diferença entre elas:

Figura 3 – Comparativo entre *native apps, webapps* e aplicações híbridas

Fonte: Wilken (2015)

3.1 APLICAÇÕES NATIVAS

As aplicações móveis nativas são desenvolvidas com o intuito de serem executadas em uma plataforma específica, desta forma, são instaladas diretamente no sistema operacional dos dispositivos e seu funcionamento não necessita de acesso à rede, ou seja, de modo *off-line*. Cada aplicação nativa é desenvolvida em linguagem padrão para determinada plataforma, como por exemplo, Java para o Android da Google, e *Objective-C* e *Swifty* para iOS da *Apple*, nesse tipo de aplicação é possível contar com os recursos distintos disponibilizados pela própria plataforma, como por exemplo, acelerômetro, gps, câmera entre outros (DEVMEDIA, 2015).

Apesar do desempenho dessas aplicações ser muito alto, é necessário que o desenvolvedor tenha conhecimento da linguagem de programação que cada plataforma necessita para a criação de aplicações. Diante disto, o custo benefício de desenvolvimento torna-se menor, uma vez que, exista a necessidade do aplicativo ser executado em demais plataformas, dessa forma o código fonte deve ser escrito novamente. Perante isto, o tempo de aprendizado é maior, devido ao fato de que é necessário obter o conhecimento acerca da linguagem de programação de cada plataforma (DEVMEDIA, 2015).

3.2 WEBAPPS

Os *webapps* ou *mobile web sites*, são aplicativos que tem um bom funcionamento em um dispositivo móvel, mas são acessadas somente através do *browser*. A maioria destes aplicativos são sites acessados de forma ágil, com a exceção de ter um layout projetado para o formato das telas dos dispositivos móveis. Além disto, não permitem que a utilização de recursos nativos, como gps, câmera entre outros. Este tipo de aplicação possui algumas dependências, como por exemplo, o acesso a uma conexão de rede, caso contrário não é possível utiliza-lo (WILKEN, 2015, tradução nossa).

Diferentemente das aplicações nativas, não existe a necessidade de uma instalação, já que as mesmas são acessadas através do *browser*. Um dos grandes fatores que influencia no funcionamento dessas aplicações, é a conexão estável com a internet, uma vez que ela é a principal responsável por garantir velocidade e disponibilidade da aplicação (WILKEN, 2015, tradução nossa).

Um problema enfrentado pelos *webapps* era referente ao seu layout, que necessitava ser reestruturado uma vez que os smartphones e tablets possuem tamanhos distintos de tela, sendo assim era necessário que o desenvolvimento fosse muito focado nestes pequenos detalhes, porém, atualmente com frameworks de UI responsivos isto deixou de ser um dos grandes problemas (WILKEN, 2015, tradução nossa).

De acordo com Wilken (2015), este modelo de aplicações é multiplataforma, é composto por HTML, CSS e JavaScript, além disto conta também com algumas ferramentas *server-side,* como Java ou PHP.

O HTML é uma linguagem de programação baseada no conceito de Hypertext, que é uma forma não linear de organizar conteúdos por meio um conjunto de elementos denominados *tags*. Desta forma, esses elementos ligados formam uma grande rede de comunicação (W3S, 2015a, tradução nossa).

Por outro lado, o CSS é responsável por tornar a parte visual mais agradavel. Sua linguagem de folha de estilo em cascata é utilizada no intuito de formatar as informações escritas pelo HTML, deixando a página web com uma interface totalmente diferente da padrão (W3S, 2015b, tradução nossa).

JavaScript é uma das linguagens de programação mais populares no desenvolvimento de aplicações *web*. Possui suporte para a maioria dos navegadores, sendo também, responsável por criar qualquer tipo de página dinâmica (W3S, 2015c, tradução nossa).

3.3 APLICAÇÕES HÍBRIDAS

Com o vasto crescimento das aplicações para dispositivos móveis, é comum a busca de recurso que facilitem o desenvolvimento destas. A criação das aplicações chamadas híbridas está se tornando cada vez mais comum no cotidiano dos programadores e empresas.

Esse tipo de aplicação é chamado de multiplataforma. Com o próprio nome diz, permite que os *apps* sejam executados em diversas plataformas. As plataformas hibridas são formadas por uma combinação de *Web Apps* e aplicações nativas.

Diante disto, as aplicações híbridas contém a versatilidade das aplicações web, basicamente uma instancia do *browser*, que será executada dentro de uma

aplicação nativa. Sendo assim, podem utilizar os recursos de uma página *web* permitindo também o acesso às funções nativas do dispositivo. Além disto, é bem adequado para uma ampla gama de aplicações e ainda pode fornecer uma boa experiência do usuário. Quando um aplicativo híbrido é construído, ele será compilado, transformando a aplicação web em um aplicativo nativo (WILKEN,2015, tradução nossa).

Ao contrário de aplicações web ou *Web Apps*, que o usuário pode acessar, navegando através da barra de endereço, os aplicativos híbridos são normalmente instalados através de uma loja virtual e estão disponíveis através do menu principal plataforma. Isso significa que os usuários têm que seguir o mesmo procedimento de instalação de uma aplicação nativa (GOK; KHANNA, 2013, tradução nossa).

Inicialmente as aplicações híbridas foram escritas com uma combinação de HTML5, CSS, JavaScript, e com os SDKs específicos de cada plataforma, tais como Java para o Android, *Objective-C* e Swift para iOS, ou C # para Windows Phone. O pacote de uma aplicação híbrida geralmente inclui uma cópia integrada de todos os recursos web necessários, ou seja, HTML, JavaScript, CSS e imagens, para que a mesma seja carrega instantaneamente assim como um aplicativo nativo, sem esperar por um servidor web (GOK; KHANNA, 2013, tradução nossa).

Segundo Gok e Khanna (2013), com o avanço nos sistemas operacionais móveis e motores de processamento de JavaScript, um aplicativo híbrido rodando em dispositivos móveis razoavelmente modernos podem oferecer experiências de usuário altamente eficientes usando HTML, CSS, e JavaScript para a camada de interface do usuário. Desta forma, as aplicações hibridas podem trazer muitas vantagens aos desenvolvedores, como por exemplo, a criação de uma UI genérica de acordo com a regra de negócio, reduzindo assim 50% do tempo de desenvolvimento de uma aplicação.

Alguns benefícios que estão incluídos nas aplicações híbridas comparadas com as nativas são (GOK; KHANNA, 2013, tradução nossa):

a) redução no tempo de mercado: construindo uma aplicação híbrida é normalmente mais rápido e requer habilidades padrões altamente reutilizáveis. Ela não envolve uma curva de aprendizado tedioso quando comparado com linguagens de programação nativas;

b) ciclo de desenvolvimento multiplataforma barata: têm compatibilidade entre plataformas, reduzindo o código nativo necessário, resultando em linguagem reutilizável como HTML5, CSS e JavaScript que podem ser compartilhados e implantado em várias plataformas com ajuste mínimo;
c) grande gama de desenvolvedores: são construídos com tecnologia web, o que significa que qualquer desenvolvedor web possui as habilidades para a criação de uma aplicação híbrida;
d) baixo custo de manutenção: devido ao seu código fonte web, não é necessário reescreve-lo para a língua nativa de cada plataforma, reduzindo assim a manutenção;
e) processo de aprovação: a maioria das lojas virtuais tem um processo de aprovação das aplicações, que verifica se o mesmo é qualificado e seguro para ser disponibilizado para venda. Devido a não necessidade de atualizar o aplicativo através da loja virtual, seu processo de aprovação é feito apenas uma vez, sendo assim podem ser lançadas várias atualizações subsequentes, sem passar por este processo;
f) aplicações híbridas são o futuro: analisando o crescimento tecnológico dos sistemas operacionais móveis, é possível argumentar que a construção e implantação de aplicativos híbridos é estrategicamente a coisa certa a fazer.

De contraposto existem também algumas desvantagens em comparação com as aplicações nativas, de acordo com Gok e Khanna (2013), um desses fatores é a performance que é afetada devido ao fato de o JavaScript ser *single-thread*, o que significa que pode executar apenas uma operação por vez, porém, a solução é executa-lo em paralelo com uma *thread* nativa. É possível perceber também que não é em todas as plataformas que as aplicações funcionam corretamente sem a necessidade de alteração do código fonte, porém, é melhor que reescreve-lo por completo.

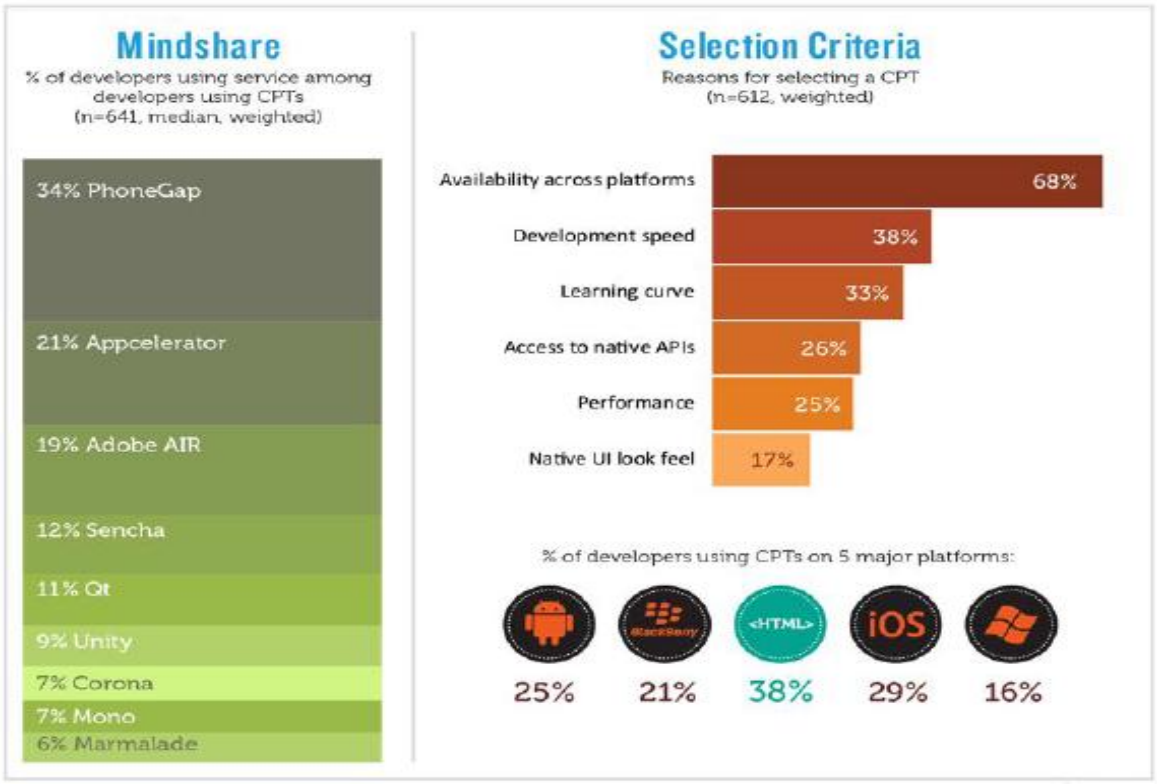
3.4 FRAMEWORKS MULTIPLATAFORMA

Com o crescimento das aplicações híbridas foram surgindo ferramentas que ajudassem no desenvolvimento, como por exemplo, *frameworks*. Atualmente existem várias ferramentas que realizam este trabalho, é possível citar, *Phonegap, KendoUI, Mobile Angular UI, Sencha Touch, AppCelerator Titanium* e entre outros (RAJ, 2014, tradução nossa).

Os primeiros *frameworks* foram *Phonegap* e *Cordova*, mesmo possuindo nomes diferentes ambos são a mesma coisa, a diferença entre eles é a forma de realização do build das aplicações, uma vez que o *Cordova* é a base que move o *Phonegap*. Foram responsáveis por revolucionar a forma de criação de aplicações híbridas, devido ao código fonte ser apenas *front-end*, sem a necessidade do uso da linguagem nativa (WILKEN, 2015, tradução nossa).

A figura 4 demonstra um comparativo entres os *frameworks* multiplataforma mais utilizados pelos desenvolvedores. Disponível através da pesquisa *Developer Economics 2013: The tools report* realizada pela Vision Mobile (2013), apresenta alguns *frameworks* que são escolhidos pelos desenvolvedores, e os cinco sistemas operacionais utilizados para a realização deste comparativo.

Figura 4 – Frameworks multiplataformas



Fonte: Vision Mobile (2013)

Os três frameworks mais utilizados em 2013, foram *Phonegap*, *Appcelerator* e *Adobe* AIR. Cada um possui uma característica que os diferencia, são elas:

a) *phonegap*: possibilita que sejam criados aplicações híbridas para diversas plataformas, entre elas iOS, Android, BlackBerry, Symbiam, Bada e Windows phone. Sua composição HTML, CSS e Javascript utiliza o conceito de webview nativo de cada plataforma, e executa a aplicação. Além disto, oferece alguns plug-ins para utilização de recursos nativos do dispositivo (PHONEGAP, 2015, tradução nossa);
b) *appcelerator titanium mobile*: além de ser um framework multiplataforma possui sua propria IDE, que auxilia os desenvolvedores na criação de *webapps* e aplicações híbridas. Tem suporte para as plataformas iOS, Android e Blackberry. Sua linguagem de programação é o Javascript e permite o uso de *server-side* como PHP, Python e Ruby (APPCELERATOR, 2015, tradução nossa);
c) *adobe air*: é um sistema de tempo de execução multiplataforma desenvolvido pela Adobe Systems para a construção de aplicativos de desktop e aplicações móveis, programado usando o Adobe Flash, ActionScript e, opcionalmente, Adobe Flex. O tempo de execução oferece suporte a aplicativos instaláveis em Windows, OS X e sistemas operacionais móveis como Android, iOS e BlackBerry Tablet OS (ADOBE, 2015, tradução nossa).

A maioria dos *frameworks* multiplataforma utilizam linguagem *web*, ou seja, baseada em HTML, CSS e JavaScript. As aplicações híbridas acabaram tornando-se muito poderosas, pois a linguagem *web* está evoluindo constantemente e cada vez mais é possível notar novidades para o desenvolvimento *web* e *front-end*. Um grande exemplo, é o AngularJS mantido pela Google. Este *framework* revolucionou o mundo do JavaScript, alterando completamente sua interação com o HTML (DOCS ANGULAR, 2015, tradução nossa).

Segundo Wilken (2015), com o surgimento do AngularJS, e pensando na performance das aplicações hibridas, em 2013 foi lançado o framework Ionic. Sua

composição totalmente diferenciada permitiu uma grande evolução na performance das aplicações híbridas.

De acordo com Raj (2014), atualmente existe um *ranking* de *frameworks* que são considerados os mais poderosos para o desenvolvimento de aplicações híbridas.

No primeiro lugar do *ranking* está o *framework* Ionic. É um dos mais promissores *frameworks* HTML5 para aplicações móveis. Construído usando SASS, ele fornece muitos componentes de interface do usuário para ajudar a desenvolver aplicativos ricos e interativos. Ele usa toda dinamicidade do *framework* AngularJs, uma vez que é um dos principais por ter transformado o Ionic tão popular. Com a nova versão Angularjs, que tem o foco em mobile, com certeza irá ganhar ainda mais popularidade (RAJ, 2014, tradução nossa).

Em segundo lugar tem-se o *framework* Mobile Angular UI, que é um *framework* de interface de usuário, que é composto por bootstrap3 e angularjs, porém, diferente do Ionic, seu CSS é o mesmo utilizado para aplicações web (RAJ, 2014, tradução nossa).

Em terceiro lugar está o Intel XDK, é uma ferramenta para a criação de aplicação multiplataforma desenvolvida pela Intel. De fácil utilização, porém, depende de outros *frameworks* de Interface de usuário (RAJ, 2014, tradução nossa).

O quarto e o sétimo lugar são respectivamente os frameworks App Celerator Titanum e Phonegap, que já foram apresentados anteriormente.

Em quinto lugar está o *framework* Sencha Touch, diferentemente dos outros, ele proporcionar a criação de aplicações híbridas que adotam totalmente o layout conforme os componentes nativos de cada plataforma, dando a impressão que cada plataforma tem uma versão diferente de aplicação (RAJ, 2014, tradução nossa).

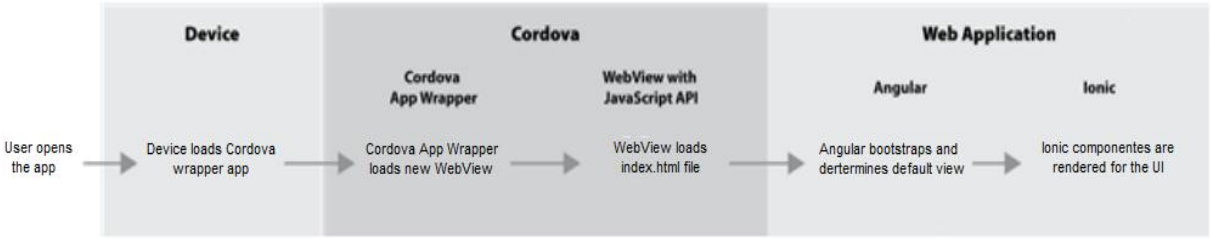
Em sexto lugar está o Kendo UI, que é um *framework* de interface totalmente dependente de jquery para realizar suas interações. Sendo assim desenvolvedores de jquery tem grande facilidade em utiliza-lo (RAJ, 2014, tradução nossa).

4 FRAMEWORK IONIC

O *framework* Ionic é uma combinação de tecnologias e utilitários projetados para tornar a facilitar e agilizar a construção aplicativos móveis híbridas com destaque visual, sendo estas tecnologias, HTML, CSS e JavaScript. Embora seja um *framework*, ele não é executado de maneira independente, já que é constituído por vários componentes, sendo eles, AngularJS como *framework* de aplicação web javascript, e Cordova que é responsável pela interpretação e construção do aplicativo para as plataformas móveis.

A figura 5 demonstra uma visão geral das diversas tecnologias utilizadas na composição do *framework* e como elas se comportam.

Figura 5 – Quadro de tecnologias utilizadas pelo framework Ionic, e como eles se encaixam



Fonte: Wilken (2015)

Analisando a figura 5, da esquerda para a direita, é possível perceber que o quadro inicial representa o dispositivo, possuindo este o sistema operacional Android ou iOS. A próxima camada é o Cordova, que é responsável por fazer a interpretação entre a plataforma móvel e o aplicativo. Ele realiza a criação de um aplicativo nativo que contém o que é chamado *WebView*, que é basicamente uma página web, onde a aplicação criada é executada e interpretada de forma interna (WILKEN, 2015, tradução nossa).

A última camada representa a aplicação *web* composta pelas tecnologias AngularJS e Ionic. Ela executa o AngularJS, que tem a principal finalidade de gerenciar a lógica e os dados do aplicativo *web*.

O AngularJS é um *framework* JavaScript, declarado muito poderoso dentre os desenvolvedores que o utilizam. É chamado de linguagem declarativa, já que ele adiciona propriedades que mudam o comportamento da linguagem e possibilita um interação dinâmica com os elementos do HTML através das chamadas diretivas (DOCS ANGULAR, 2015, tradução nossa).

Construído em cima do AngularJS, o framework Ionic, é utilizado especialmente para projetar a interface do usuário, desta forma, inclui-se todos os elementos visuais, como guias, botões, cabeçalhos de navegação, pop-ups entre outros. Seus componentes são criados por uma combinação de HTML, CSS e JavaScript (AngularJS) que se comportam como os controladores nativos da plataforma (PHAN, 2014, tradução nossa).

O núcleo do Ionic são os controladores de interface do usuário que não estão presentes no HTML, mas que são comuns para aplicativos móveis. No entanto, ele inclui ainda uma série de utilitários e recursos adicionais que ajudam a gerenciar seu aplicativo desde a criação até a pré-visualização para a implantação (WILKEN, 2015, tradução nossa).

Segundo Wilken (2015), o Ionic foi lançado em 2013, e desenvolvido inicialmente por Drifty. Ganhou popularidade rapidamente, sendo atualmente, uma das primeiras escolhas para a criação de aplicativos híbridos. Ainda ressalta que mais de 20 mil aplicativos feitos com Ionic são lançados mensalmente.

Para ajudar no desenvolvimento das aplicações o Ionic disponibiliza ainda uma ferramenta chamada Linha de Commando do Ionic do inglês *Command Line Ionic* (CLI), que gera os projetos iniciais, possibilitando também a pré-visualização da aplicação, a criação e implantação do aplicativo. O Ionic disponibiliza aos desenvolvedores uma biblioteca com vários ícones e tipos de fonte, para tornar a aplicação mais dinâmica (WILKEN, 2015, tradução nossa).

Os frameworks para desenvolvimento de aplicações hibridas não são novidades, atualmente, existem vários frameworks com esta utilidade, porém, o Ionic traz um novo e importante conjunto de melhorias para estes aplicativos. Recentemente, era possível notar que os aplicativos híbridos eram relativamente lentos e não tinham o mesmo desempenho que um aplicativo nativo. Tudo isso mudou com o surgimento do Ionic, ele permite a criação de aplicações de grande desempenho (WILKEN, 2015, tradução nossa).

Segundo Wilken (2015), embora seja uma ferramenta mais atual, o Ionic é um dos frameworks mais utilizados pelos desenvolvedores, devido a sua capacidade de proporcionar a criação de aplicativos de interface limpa, rápida criação e com a mesma performance de uma aplicação nativa. Os itens abaixo demonstram alguns motivos pelo qual o Ionic é uma das principais escolhas dos desenvolvedores (WILKEN, 2015, tradução nossa):

a) construção de aplicativos a plataforma web: usando HTML, CSS e Javascript é possível criar aplicações hibridas que se comportam como aplicações nativas;
b) construído com AngularJS: as aplicações criadas com Ionic permitem a utilização dos recursos adicionais do AngularJS;
c) utiliza de técnicas modernas: o framework foi projetado para trabalhar com os recursos de css3, permitindo o uso de animações nas aplicações;
d) poderosa ferramenta CLI: com esta ferramenta é possível gerenciar as tarefas de desenvolvimento rapidamente, tais como pré-visualização em um emulador, implantação em dispositivos móveis conectados e configuração de projetos iniciais;
e) ecossistema Ionic: fornece ainda um rico ecossistema que facilita o processo de desenvolvimento das aplicações;
f) permite o desenvolvimento através de web *browsers*: como é um framework *front-end*, é possível executar o ionic em web *browsers*, permitindo que sejam feitos debugs no código fonte facilitando o desenvolvimento do aplicativo.

Além de facilitar o trabalho dos desenvolvedores, este framework acabou se tornando uma boa opção para os gestores que querem garantir que o produto produzido em sua empresa seja feito com uma plataforma sólida, e que pode reduzir o trabalho e o custo de tempo dos desenvolvedores para a criação de aplicações. A baixo cita-se alguns motivos pelo o qual o Ionic é uma boa opção de escolha para os gestores utilizarem em sua empresa (PHAN, 2014, tradução nossa):

a) abrange todas as bases: é capaz de fornecer um conjunto completo de recursos necessários para a construção de um aplicativo hibrido completo;
b) *framework open-source*: muitos desenvolvedores tem preferência esse tipo de framework, porém, o ionic tem licenciamento permissivo para o uso de código fonte aberto, não prejudicando assim os direitos autorias do produto;

c) ionic tem uma equipe dedicada: escolher frameworks open-source gera dúvida, porém, o ionic tem uma equipe dedicada que pretende manter a plataforma na liderança;
d) desenvolvimento rápido das aplicações: em poucos minutos é possível gerar a base da aplicação e modifica-la para construir um aplicativo sólido, garantindo melhor gestão no tempo, custo e facilidade de manutenção;
e) ionic é divertido: apesar de ser subjetivo, o desenvolvimento com Ionic faz com que os desenvolvedores se divirtam criando os aplicativos, e acabam sendo estimulados com isso.

Contudo a elaboração de aplicações dinâmicas é cada vez mais comum. Além disto, seu CSS é responsivo possibilitando que as aplicações sejam redimensionadas de acordo com a tela do dispositivo que esta reproduzindo a aplicação sem a necessidade de fazer alterações. Ele ainda proporciona ao usuário final o mesmo desempenho de um aplicativo nativo e com um design que não é possível criar com as ferramentas nativas (WILKEN, 2015, tradução nossa).

Diante dos aspectos apresentados, os subcapítulos a seguir irão demonstrar um pouco sobre as tecnologias que compõem o *framework*, e qual a importância de cada uma delas para o funcionamento do Ionic.

4.1 NODEJS

Embora o NodeJS não seja uma das ferramentas responsáveis pela composição do framework Ionic, ele é uma das essências do mesmo. É no gerenciador de pacotes *Node Package Manager* (NPM) que ficam armazenados as bibliotecas e plataformas de desenvolvimento com JavaScript e entre elas estão o Cordova e o Ionic.

Essa plataforma foi criada sobre um motor JavaScript do *browser* Google Chrome para que fosse possível simplificar a construção de aplicações rápidas. Utilizando o modelo I/O que é direcionado ao uso de eventos não bloqueantes, tornando-o assim mais leve e eficiente, essa plataforma é uma ótima opção para aplicações que trabalham em tempo real com a transferência de dados através de dispositivos distribuídos. Basicamente o Node é um servidor de programas que roda

em JavaScript e utiliza a máquina V8 do Google que é um interpretador para o servidor (NODEBR, 2015).

O NPM do NodeJS é definida por dois fundamentos, é um repositório online responsável pela publicação de projetos de código fonte para o próprio Node, e também, serve como utilitário para linha de comando que faz a interação com este repositório online, ajudando na instalação de pacotes, gerenciamento de versões e dependências (NODEBR NPM, 2015).

4.2 CORDOVA

Apache Cordova trata-se de um conjunto de APIs de dispositivos que permite aos desenvolvedores de aplicativos móveis a possibilidade de acessar as funções do dispositivo nativo, como a câmera ou o acelerômetro do JavaScript por exemplo. Combinado com frameworks tais como jQuery Mobile, Dojo Mobile ou Sencha Touch, que permitem que um aplicativo para dispositivos moveis possam ser desenvolvidos apenas com linguagem web HTML, CSS e JavaScript (APACHE CORDOVA, 2015, tradução nossa).

Uma das grandes vantagens da utilização do Cordova é a possibilidade de construir uma aplicação móvel sem qualquer necessidade de utilização de código nativo (*Java, Objective-C*, etc). De contraposto a esta situação, é possível usufruir das tecnologias web para o desenvolvimento que estão hospedadas na própria aplicação localmente (APACHE CORDOVA, 2015, tradução nossa).

O que torna o Cordova uma poderosa ferramenta para o desenvolvimento de aplicações móveis é a sua composição de APIs JavaScript que tem base em padrão de desenvolvimento web, fazendo assim com que seja possível a portabilidade para demais plataformas sendo necessário apenas refazer a compilação do projeto para os demais dispositivos sem a necessidade de alteração de código fonte. Ainda é possível ressaltar que os aplicativos feitos em Cordova podem ser empacotados usando a plataforma SDK, e pode ser disponibilizado para a instalação na loja de aplicativos de cada dispositivo (APACHE CORDOVA, 2015, tradução nossa).

Seu conjunto de bibliotecas JavaScript uniforme podem ser utilizados para a construção do código fonte, e possuem suporte nativo para os dispositivos específicos. Além disto, o Cordova está disponível para as seguintes plataformas:

iOS, Android, BlackBerry, Windows Phone, Palm WebOS, Bada, e Symbian (APACHE CORDOVA, 2015, tradução nossa).

Além de ser uma das principais ferramentas que compõe o *framework* Ionic, é responsável por torná-lo uma ferramenta tão poderosa para o desenvolvimento de aplicações hibridas de alto nível. Sua estrutura totalmente em linguagem web permitiu que os criadores do Ionic apenas aprimorassem o uso do JavaScript para que as aplicações tivessem a mesma performance de uma aplicação nativa (WILKEN, 2015, tradução nossa).

4.3 ANGULARJS

O AngularJS é um *framework open-source* para *Javascript* que fornece aos seus desenvolvedores uma estrutura forte, permitindo que a produtividade e o desenvolvimento de aplicações para web sejam completas. O framework teve início em 2009, quando o funcionário da Google Misko Hevery, resolveu criar o projeto para facilitar o armazenamento de JSON online, que teria seria preço de acordo com a quantidade de megabytes solicitados, porém, ao longo do tempo desistiu da ideia e resolveu abandonar o projeto e distribuir o Angular como uma plataforma de código aberto. Posteriormente a Google Inc. começou a apoia-lo e em junho de 2012 foi realizada o lançamento da primeira versão do AngularJS oficialmente (DOCS ANGULAR, 2015, tradução nossa).

Este framework tem uma grande composição que torna a sua utilização cada vez mais frequente entre os desenvolvedores de páginas web. Sua principal funcionalidade é ser utilizado como extensão dentro de uma página HTML, além de possibilitar a fácil utilização do Ajax. Basicamente o AngularJS é uma linguagem chamada declarativa, já que utiliza a adição de propriedades que mudam o comportamento da linguagem e possibilita uma interação dinâmica com os elementos do HTML (DOCS ANGULAR, 2015, tradução nossa).

Uma forma muito inovadora de trabalhar com o HTML está presente no AngularJS, pois o mesmo trabalhar com as chamadas diretivas que são responsáveis por facilitar o manuseio dos elementos do HTML. Estas diretivas dão poder a estre framework e o diferencia dos demais, já que ele praticamente diz ao Browser como devem ser exibidas as páginas web alterando o padrão de comportamento do HTML. As Diretivas permitem ao desenvolvedor especificar tags

HTML personalizadas e reusáveis, que personalizam o comportamento de certos elementos. Abaixo estão ilustradas as principais diretivas e para que servem (DOCS ANGULAR, 2015, tradução nossa):

a) *ng-app*: define o elemento como elemento raiz da aplicação;
b) *ng-bind*: modifica o texto de um elemento HTML automaticamente, de acordo com o seu resultado, vindo das regras de negócio;
c) *ng-model*: proporciona a ligação da *view* com o escopo, criando uma ligação bidirecional;
d) *ng-class*: carrega dinamicamente os atributos da classe;
e) *ng-controller*: determina um *controller* JavaScript;
f) *ng-repeat*: interação com cada elemento de um *array;*
g) *ng-show*: mostra o elemento HTML;
h) *ng-hide*: esconde o elemento HTML;
i) *ng-switch*: permite instanciar um *template* em uma lista de opções, de acordo com o valor resultante da expressão;
j) *ng-view*: utilizada para manipulação de rotas;
k) *ng-if*: declaração de condição.

Outra característica muito importante do AngularJS é a utilização do padrão *Model-View-Controller* (MVC). Recentemente foi criado um novo padrão de para o AngularJS que é *Model-View-Whatever* (MVW) que ressalta que independente do padrão de projeto utilizado os conceitos de *model, view* e *controller* deve ser sempre seguidos para que cada funcionalidade seja separada corretamente dentro de uma aplicação (DOCS ANGULAR, 2015, tradução nossa).

Dentre os componentes do Ionic o AngularJS é o responsável por fazer com que a performance dos aplicativos seja de alto nível. Sua logística de trabalho baseada no uso de diretivas acaba propiciando a estrutura da aplicação web ganho significativo em performance (IONIC, 2015, tradução nossa). Além disso, vem crescendo e se destacando entre os *frameworks* JavaScript. Ao obter-se o Cordova como base para a criação de aplicações hibridas e alterar o JavaScript simples para AngularJS, o Ionic adquiriu um grande ganho com performance fazendo com que seu uso torne-se cada vez mais frequente.

4.4 RECURSOS DO IONIC

O grande diferencial do *framework* está em seus recursos. Da mesma maneira que o Twitter Bootstrap revolucionou as aplicações web, o Ionic alterou o conceito sobre o desenvolvimento de aplicativos móveis híbridos. Ele cuida de toda a aparência e das interações entre telas a fim criar uma aplicação atraente (IONIC, 2015, tradução nossa).

Iniciando pelos componentes de interface, possui um CSS único, gerado a partir de SASS, que é uma forma simplificada do fluxo do CSS, e o mesmo pode ser sobrescrito, tornando fácil personalização de interface das aplicações. Além disso o mesmo é independente da camada JavaScript, concedendo maior arbítrio na criação das aplicações. Possui um sistema de grids que possibilita a criação de layout com uma quantidade ilimitada de colunas, possibilitando também o alinhamento vertical dentro destas (IONIC, 2015, tradução nossa).

De modo geral, a parte visual do framework inclui as seguintes opções (IONIC, 2015, tradução nossa):

a) *headers*: assim como nas páginas web os *headers* ou cabeçalhos, são regiões fixadas no topo da tela das aplicações e contém vários estilos de cores;
b) *footers*: os rodapés são regiões na parte inferior de uma tela que podem conter vários tipos de conteúdo, possui os mesmo estilos de cores que os *headers*;
c) *content*: a área de conteúdo no Ionic é a janela de exibição de rolagem do seu aplicativo. Enquanto os cabeçalhos e rodapés estão fixados à parte superior e inferior, respectivamente, a área de conteúdo vai preencher o espaço disponível restante;
d) *buttons*: o botão é a essencial de qualquer aplicação móvel, já que ele é responsável por realizar as ações, porém, visualmente falando os botões do Ionic possuem vários estilos de cores, e tamanhos e podem ser utilizados de forma dinâmica dentro das aplicações;
e) *list*: a exibição de lista é um componente muito versátil e poderoso. Exibições de lista suporta vários modos de interação, tais como a edição, arrastar para reordenar e puxar para atualizar;

f) *cards*: cartões tornaram-se amplamente utilizado nos últimos anos. Eles são uma ótima maneira de conter e organizar a informação, ao mesmo tempo, criam expectativas previsíveis para o utilizador;
g) *forms*: da mesma maneira que nas páginas web, os formulários possuem vários tipos de input exclusivamente desenhados com o css do Ionic;
h) *toggle*: as *toggle* ou alternâncias, são uma ferramenta que funcionam semelhantes ao *checkbox*, porém, são mais sensíveis ao *touchscreen*. O CSS ainda permite que cada *toggle* possua uma cor especifica;
i) *checkbox*: as caixas de seleção funcionam exatamente igual aos do HTML comum;
j) *radio buttons*: diferente dos radio *buttons* padrões, é possível fazer com que o item selecionado seja marcado com um ícone especifico;
k) *range*: os intervalos podem ser temáticos e utilizar qualquer cor Ionic padrão, além de possuir vários outros elementos, como um item de lista ou de cartão.;
l) *tabs*: são uma região horizontal de botões ou links que permitem uma experiência de navegação consistente entre as telas. Ele pode conter qualquer combinação de texto e ícones, é um método popular para permitir a navegação móvel.

Além dos itens citados acima o Ionic ainda vem alguns utilitários opcionais para o design das telas, como cores, ícones e preenchimento automático de layout (IONIC, 2015, tradução nossa).

Contudo, o Ionic ainda possui uma gama de componentes que executam as ações básicas de uma aplicação todas escritas com AngularJS, o que agiliza muito e garante muito a performance das ações feitas com JavaScript simples.

Muitos componentes CSS precisam do Javascript a fim de produzir os efeitos. O Ionic segue o padrão MVC, garantindo assim boa organização, leitura e entendimento do código fonte.

Seu CLI, é o uma dos principais fatores por torna-lo genial. De acordo com seus criadores, essa ferramenta proporciona a facilidade de muitas ações, bem como a geração de projetos, certificado de segurança das aplicações, criação do

build para várias plataformas, geração de ícones e *splashs* entre outra principais funcionalidades (IONIC, 2015, tradução nossa).

Além disto, como seu *framework* javascript é o angular, é possível utilizar também, dos benefícios do mesmo, tais como criação de novas diretivas, criação de serviços, controle de rotas e muitas outras funcionalidades que tornam o código fonte da aplicação mais limpo, facilitando muito mais as questões de manutenção. A forma que ele trabalha com *collections*, facilita muito a manipulação de dados via lista e *arrays*. Além do mais sua interação com o HTML, torna mais fácil a obtenção de valores retornáveis de um campo independentemente do tipo (DOCS ANGULAR, 2015, tradução nossa).

Diante do estudo levantado acerca dos frameworks multiplataforma, o capítulo a seguir traz uma breve descrição sobre os motivos pelo qual escolheu-se o desenvolvimento para a área de finanças pessoais.

5 CONTROLE FINANCEIRO PESSOAL

Com o intuito de instruir as pessoas em utilizar melhor o seu dinheiro, foi criada a denominação finanças pessoais que por muitos economistas é considerada uma ciência. Para Cherobim e Espejo (2010), por exemplo, ela faz o estudo e aplica os conceitos financeiros para a tomada de decisões de um indivíduo ou família. Porém Sandroni (2008) acrescenta que as finanças pessoais fazem um breve estudo sobre as melhores formas da utilização de crédito no mercado financeiro, realizando aplicação vantajosas e criando novos meios de obtenção de renda pessoal.

Os conceitos financeiros sempre tiveram suma importância para que houvesse um desenvolvimento social e intelectual da humanidade. Com o intuito de orientar melhor os indivíduos esses conceitos utilizam técnicas especificas que permitem a gestão financeira pessoal.

Atualmente é comum que as pessoas façam o controle das finanças para obterem equilíbrio financeiro. É possível utilizar das técnicas disponibilizadas por especialista e economistas, com o intuito de controlar os mínimos detalhes a respeito do dinheiro.

Criar um orçamento que se encaixe dentro da renda salarial de cada cidadão ou cidadã não é uma tarefa considerada simples e poucos tem domínio sobre como realiza-lo. Quanto melhor for o controle, a organização e o planejamento financeiro, mais eficientes serão as escolhas de consumo, disciplinando o uso da renda de um determinado indivíduo (CERBASI, 2012).

Luquet (2000) afirma que a chave para reter o medo desnecessário que as pessoas tem quando o assunto é dinheiro, é o controle financeiro, uma vez que ao organizar suas finanças com critérios bem definidos é possível descobrir maiores recursos e benefícios que podem ser aplicados com a renda.

Segundo Frankenberg (1999) o planejamento financeiro pessoal pretende estabelecer uma estratégia que seja precisa, deliberada e dirigida, onde o acumulo de bens e valores tem o propósito de formar o patrimônio de um indivíduo ou família. Ele afirma também que o planejamento de finanças pessoais é árduo pois existem inúmeros imprevistos e incertezas na vida.

Cerbasi (2004), por exemplo, instrui os leitores de seus livros, a focarem nas despesas que são pequenas e que as pessoas naturalmente não costumam a

anotar. Sugere ainda a utilização de recursos tecnológicos como por exemplo, planilhas e softwares para a gestão financeira.

A busca de recursos para controlar as finanças é um meio utilizado por muitas pessoas que pretendem adquirir uma margem de tranquilidade financeira. Segundo Torralvo, Sousa e Rocha (2012) utilizar recursos tecnológicos para controlar as finanças lhe dá vantagens, já que as informações são mais exatas e acessíveis facilitando a organização de despesas e receitas de forma precisa e direta, além de proporcionar um histórico completo sobre as finanças, possibilitando que os indivíduos estejam sempre atualizados e garantindo que não gastem mais do que podem adquirir.

Os meios tecnológicos para controle financeiro proporcionam conforto, usabilidade e segurança para os usuários, garantindo fácil acesso as projeções de receitas, despesas de forma mais precisa, uma vez que os indivíduos estão sempre utilizando smartphones com maior frequência, sendo possível fazer o controle a qualquer momento (CERBASI, 2012).

6 TRABALHOS CORRELATOS

Atualmente é cada vez mais comum que o número de projetos de pesquisas referente a área de tecnologia venham se expandindo, já que a tecnologia está em constante crescimento. Diante deste crescimento tecnológico podemos destacar a área de dispositivos móveis e recursos para a web, devido à grande quantidade de funcionalidades disponível para a criação de novos recursos.

Este capitulo abordara alguns projetos de pesquisas relacionados a o trabalho que está sendo proposto.

6.1 ESTUDO DE FRAMEWORKS MULTIPLATAFORMAS PARA O DESENVOLVIMENTO DE APLICAÇÕES MOBILE HÍBRIDAS

Em 2014, os acadêmicos Bruno Batista Boniati e Ezequiel Douglas Prezotto, do curso de Tecnologia em Sistemas para Internet da Universidade Federal de Santa Maria, realizou um estudo sobre frameworks multiplataformas para utilização no desenvolvimento de aplicações móveis híbridas através de um estudo de caso.

Com o crescimento das tecnologias móveis, notou-se um aumento na diversidade de equipamentos e dispositivos. Da mesma forma, a quantidade de plataformas e as linguagens de programação para aplicação móveis também receberam este impacto. Neste interim deu-se origem as aplicações híbridas que permitem que uma única aplicação seja reproduzida em várias plataformas (BONIATI; PREZOTTO, 2014).

Baseado em frameworks para o desenvolvimentos de aplicações híbridas Boniati e Prezotto (2014), demonstram algumas características e pontos fortes e fracos acerca dos principais frameworks criando uma aplicação para o controle de aulas e de presença de alunos. Este software permite o cadastro das aulas e dos alunos que fazem parte desta, sendo possível bater uma foto do aluno e controlar se ele esteve presente ou ausente.

De acordo com Boniati e Prezotto (2014), para a criação deste protótipo de aplicativo híbrido, foi utilizado o framework Apache Cordova e a mesma foi desenvolvida através da ferramenta *SublimeText*. Seu teste foi realizado no

dispositivo Android, e simulados nos emuladores do FirefoxOS, WindowsPhone e BlackBerry.

Através do conhecimento obtido, foi possível notar de um dispositivo para o outro as interfaces sofrem pequenas mudanças entre as plataformas e que cada uma interpreta os componentes da aplicação de forma distinta (BONIATI; PREZOTTO, 2014).

6.2 DESENVOLVIMENTO DE CROSS-PLATFORM MOBILE APPS UTILIZANDO O TITANIUM MOBILE

A elaboração de um projeto de pesquisa com o intuito de desenvolver uma aplicação multiplataforma com o framework Titanium Mobile, foi realizada no ano de 2012, pelo acadêmico Adriel Almeida Café, do curso de Sistemas de Informação da Faculdade Zacarias de Góes do estado da Bahia.

Com base no framework Titanum Mobile foi realizado um levantamento dos pontos positivos e negativos acerca deste framework criando uma aplicação híbrida aplicada através de um estudo de caso (CAFÉ, 2012).

Segundo Café (2012), Titanium Mobile é um framework de código fonte aberto, ou seja, open-source, e foi fundado em 2006 pela empresa Appcelerator. Ele possui uma IDE própria chamada Titanium Studio, permitindo a criação de webapps e aplicações híbridas. Além disto, suas aplicações são feitas com HTML5, CSS3 e Javascript, suportando também linguagens *server-side* PHP, Ruby e Python.

O estudo de caso demonstrará o funcionamento do Titanium Mobile, e como através das tecnologias web, o código fonte pode ser reutilizado permitindo a portabilidade entre as plataformas Android e iOS.

Será rescrito um aplicativo nativo da Google Play chamado DicInfo e replicado para estas plataformas utilizando linguagem web (CAFÉ, 2012).

A aplicação criada utilizará o mesmo banco de dados da aplicação nativa. Contará com recursos comuns a ambas as plataformas como por exemplo *SQLite* e o uso do *ListView*, que é responsável por apresentar o conteúdo. A interface do usuário não foi alterada, sendo assim ela utilizara a interface padrão de cada plataforma (CAFÉ, 2012).

De acordo com Café (2012), o uso do framework possibilitou que fosse desenvolvida uma aplicação que pudesse ser executada em mais de uma plataforma, através de um único código fonte reutilizável e simples.

6.3 ESTUDO DA TECNOLOGIA PHONEGAP/CORDOVA E A APLICAÇÃO EM UM ESTUDO DE CASO

Em 2014, o acadêmico Férlon M. Piran, pelo Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandensse do campus Passo Fundo, do curso de Sistemas de Informação para Internet fez um estudo baseado na tecnologia Phonegap que é um framework para o desenvolvimento de aplicações móveis híbridas. Para este estudo ela foi aplicada para a criação de um mural de mensagens e recados.

Está ferramenta permite a criação de aplicações que pode ser executada em qualquer plataforma móvel, sem a necessidade de reescrever o código fonte para cada sistema operacional. Utiliza linguagem web para o desenvolvimento, sendo assim não existe a necessidade do programador conhecer a linguagem nativa de cada plataforma (PIRAN, 2014).

O Phonegap é um framework open-source, criado para o desenvolvimento de aplicações multiplataformas utilizando tecnologias como HTML5, CSS e Javascript. Além disto, sua tecnologia container possibilita que as aplicações sejam transformadas em nativas permitindo o uso de recursos nativos, sendo assim, são instaladas normalmente como qualquer outra aplicação (PIRAN, 2014).

De acordo com Piran (2014), este framework conta ainda com uma ferramenta responsável por criar o build das aplicações chamada Adobe Phonegap Build. Ela facilita a compilação das aplicações e faz a portabilidade para várias plataformas.

O estudo de caso criado, o usuário poderá visualizar os recados e mensagens que estão listado ordenados por data e hora, possibilitando ainda o filtro por categorias, permitindo que o recebedor apague os recados. As mensagens tem um formulário na seguinte composição, título, categoria, mensagem com número máximo de 150 caracteres, e-mail, data e hora (PIRAN, 2014).

Com esse estudo foi possível perceber que o framework Phonegap proporciona o desenvolvimento de aplicações multiplataformas concedendo aos

desenvolvedores maior tranquilidade, e facilitando o trabalho para manutenções em código fonte, além disto, não é necessário que se tenha conhecimento acerca de todas as linguagens de programação nativas de cada plataforma.

6.4 DESENVOLVIMENTO HÍBRIDO VERSUS DESENVOLVIMENTO NATIVO DE APLICATIVOS MÓVEIS

A estudante Mariana Ribeiro Mendes, do curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Minas Gerais do Campus Bambuí realizou, no ano de 2014, um estudo comparativo entre o desenvolvimento de aplicações híbridas e nativas apresentando as diferenças entre ambas e os principais motivos pelo qual as empresas estão adotando a criação de aplicações híbridas e os aspectos pelo qual as aplicações nativas não são viáveis para determinados ambientes.

Segundo Mendes (2014), as aplicações nativas tem uma certa dependência de conhecimento especifico, já que cada plataforma de desenvolvimento tem uma linguagem diferente. Ressalta ainda que com o grande crescimento das plataformas móveis, é cada vez mais inviável a criação de uma aplicação nativa de médio ou grande porte, já que o custo benefício para mantê-la para várias plataformas é custoso tanto em tempo de desenvolvimento quanto em manutenção.

Por outro lado o aplicativo híbrido é basicamente uma instancia do browser do Sistema Operacional, que implementa uma classe que é capaz de exibir o conteúdo web. E diferentemente do aplicativo nativo, ele não possui funcionalidades do dispositivo, sendo necessário a utilização de algum framework, como por exemplo o Phonegap, para que seja feita a compilação e o build para que transformará a aplicação web em uma aplicação nativa para que a mesma possa ser instalada e executada, dando acesso aos recursos nativos do dispositivo (MENDES, 2014).

De acordo com Mendes (2014), é vantajoso para as empresas a criação de aplicações híbridas, devido ao fato de não ser necessário ter que reescrever várias vezes o código fonte para diferentes plataformas, além disto, esse tipo de aplicação facilita as manutenções e tempo de desenvolvimento de um determinado aplicativo.

7 PROTOTIPO DE APLICAÇÃO HÍBRIDA PARA O CONTROLE FINANCEIRO UTILIZANDO O FRAMEWORK MULTIPLATAFORMA IONIC

Com base no conhecimento adquirido durante o levantamento bibliográfico o referente trabalho tem como propósito desenvolver um protótipo de aplicativo híbrido utilizando o framework Ionic. Para isso foi necessário a preparação de um ambiente adequado para o desenvolvimento e alguns recursos específicos.

O aplicativo é direcionado para a área de gestão financeira e poderá ser utilizado em dispositivos móveis que possuem as plataformas Android e iOS. Além disto, foram utilizados *plugins* para o banco de dados e para a exibição de uma janela de confirmação para a exclusão dos itens, respectivamente, *SQLite* e *Dialogs*.

7.1 METODOLOGIA

Para providenciar este projeto de pesquisa, inicialmente foi realizado o levantamento bibliográfico necessário para elaboração do referencial teórico, em livros, artigos e publicações, por meio de pesquisas na internet e *ebooks*.

Após o levantamento, deu-se início a criação do referencial teórico, abordando os principais elementos acerca da elaboração de uma aplicação híbrida utilizando frameworks multiplataforma, neste caso, o Ionic.

Foram abordados alguns aspectos, como plataformas móveis Android e iOS, aplicações móveis nativas, *webapps* e híbridas com maior ênfase. Além disto, comentou-se a respeito de frameworks multiplataformas e os mais utilizados pelos desenvolvedores. Por fim, um capitulo acerca do Framework Ionic, sua composição e seus recursos e outro sobre aplicações móveis para o controle financeiro.

A próxima etapa foi o desenvolvimento da aplicação. Inicialmente criou-se uma aplicação de teste para compreender e estudar o funcionamento do framework. Em seguida, iniciou-se a modelagem dos dados para a criação de banco. Foi realizado também um levantamento a respeito dos requisitos e das ferramentas extras que seriam utilizadas, como por exemplo plugins. Diante disto, a aplicação para o controle financeiro começou a ser desenvolvida.

Os testes foram realizados durante o processo de desenvolvimento, uma vez que as compilação e execuções eram efetuadas diretamente no dispositivo

móvel com a plataforma Android. Somente foram feitos testes no emulador do iOS quando a aplicação foi finalizada.

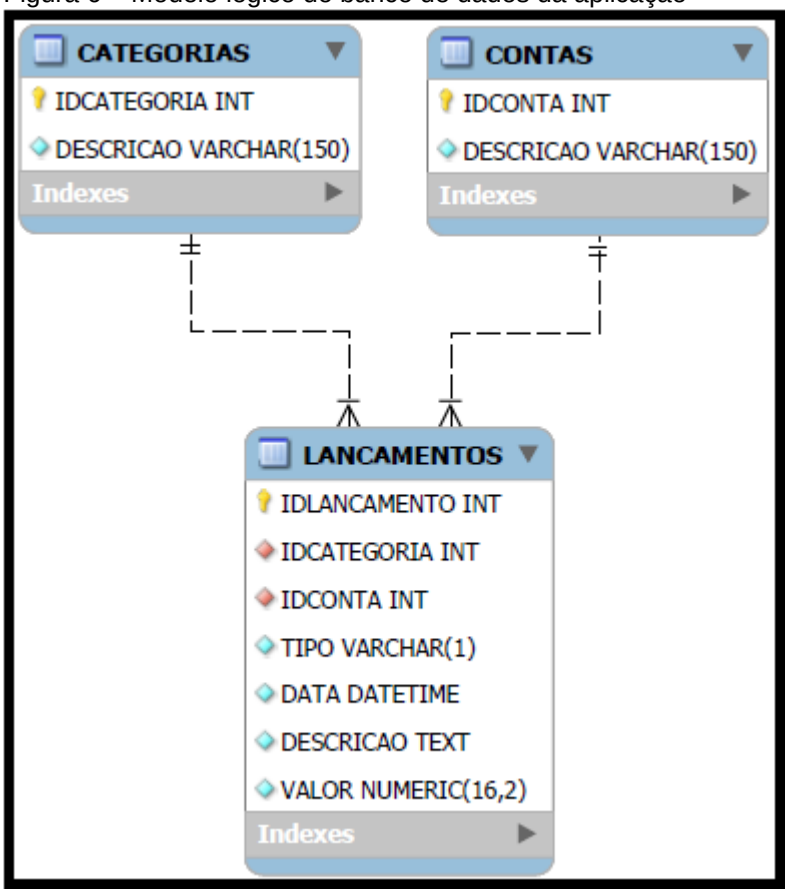
7.1.1 Modelagem dos Dados

Para o desenvolvimento da aplicação foi necessário a realização de um levantamento acerca da estrutura dos softwares móveis para o controle financeiro e quais as principais funcionalidades.

Modelar os dados da aplicação facilita no desenvolvimento, uma vez que através dela é possível analisar e obter uma maior percepção de como funcionara a ligação entre os cadastros, diminuindo os erros de programação. Sendo assim, torna-se um requisito que deve ser considerado importante.

Para a criação da modelagem, foi utilizada a ferramenta MySQL Workbench, que permite que, através do modelo lógico, sejam gerados os *scripts* para a criação do banco de dados (MYSQL WORKBENCH, 2015, tradução nossa), sendo assim, a figura 6, mostra o modelo lógico do banco de dados:

Figura 6 – Modelo lógico do banco de dados da aplicação



Fonte: Do autor.

Conforme demonstrado na figura 8, optou-se por desenvolver uma tabela única que receba os lançamentos, onde o campo *tipo* recebe o tipo do lançamento, sendo ele despesa ou receita.

7.1.2 Ferramentas para o Desenvolvimento

Após a modelagem dos dados, foram levantadas as ferramentas que seriam utilizadas para a criação da aplicação. Inicialmente foi instalado uma das principais ferramentas, o NodeJS. Ele contém os pacotes com a instalação do Ionic e está disponível gratuitamente através do endereço: https://nodejs.org/. Após isso, utilizou-se a seguinte expressão no *prompt* de comando do Windows:

Figura 7 – Instalação do Ionic

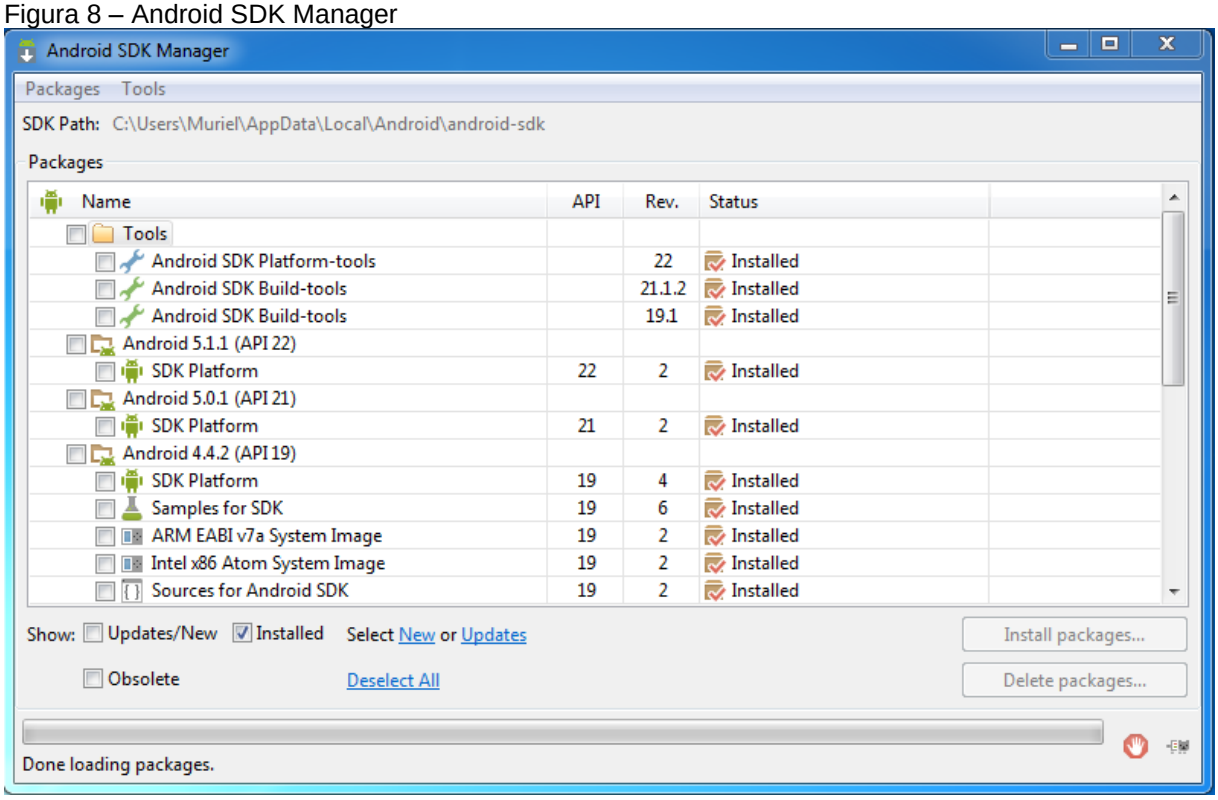
```
$ npm install -g cordova ionic
```

Fonte: Ionic (2015)

Esse comando é responsável por instalar o framework Ionic, dando acesso a todo o CLI e seus recursos em geral.

As próximas ferramentas foram o Apache ANT e o editor de texto SublimeText2 que serviu como IDE. Juntamente com essas foi instalado o Java SDK disponível em: https://developer.android.com/sdk/, ele gerencia todas as APIs que são instaladas através do SDK Manager. Nele estão presentes novas versões do sistema operacional Android. Esse procedimento torna-se muito importante, uma vez que o Ionic necessita de uma versão de API para gerar o *build* das aplicações. Para este projeto foi utilizada a API 19. A figura 8 demonstra o SDK Manager e suas versões:

Figura 8 – Android SDK Manager



Fonte: Do autor.

O ambiente ainda contou com mais três ferramentas essenciais, o Git que é uma ferramenta para o gerenciamento do controle de versão de projetos via *prompt* de comando (GIT, 2015, tradução nossa). Ele ainda é utilizado para realizar o download de alguns plugins que estão armazenados no github.

Outra ferramenta muito importante é o gerenciador de utilitários chamado Bower, que é dependente do NodeJs e do Git (BOWER, 2015, tradução nossa). Além disto, permite a instalação de plugins ngCordova, que funcionam com uma combinação de Cordova e AngularJS (NG CORDOVA, 2015, tradução nossa).

Por fim para o gerenciamento do projeto foi utilizado a ferramenta *Source Tree*. Ela permite que se obtenha um historio completo das versões salvas no Git e possibilita que sejam salvas as novas alterações.

7.1.3 Desenvolvimento da Aplicação

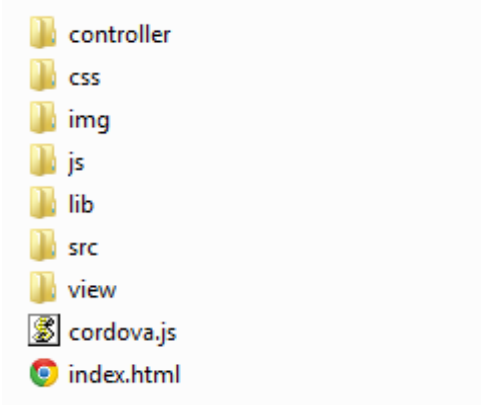
Posterior a conclusão da configuração de ambiente, deu-se início ao desenvolvimento do protótipo de aplicação híbrida para o controle financeiro. Para isso foram utilizados os comandos de criação de projetos do CLI do ionic, que são representados por um linha de comando, possibilitando ainda a escolha de três tipos

diferentes de layout iniciais. Optou-se então a escolha do layout em branco, para que o mesmo fosse modelado de acordo com a necessidade.

Sendo assim o projeto contém uma estruturação de pastas onde estão armazenadas várias informação e também em uma pasta separada com o código fonte da aplicação. Os demais repositórios são utilizados apenas para fins de compilação e interpretação para o *build* do *app* bem como os *plugins* utilizados.

A principal pasta do projeto é a *www*. Nela está contido todo o código fonte da aplicação.

Figura 9 – Estrutura da pasta www



Fonte: Do autor.

Na pasta *js* estão todos os arquivos *javascript* que fazem parte do funcionamento e configurações do sistema. O arquivo principal da aplicação é o *app.js*, pois nele, além de atribuir o nome da aplicação, injetar os recursos necessários para o funcionamento do Ionic e os demais serviços criados, é iniciado a variável global *app* e atribuído a ela a instância da aplicação, de modo que todos os *controllers* e *routes*, serão criados a partir dela. Embora o Angular permita modularizar em arquivos distribuídos, foi optado por iniciar e criar o banco de dados neste. A figura 10 retrata o código desse arquivo.

Figura 10 – Arquivo javascript *app.js*

```
var db = null;
var app = angular.module('upmoney', ['ionic','ngCordova','chart.js','upmoney.services']);

app.run(function($ionicPlatform,$cordovaSQLite,$cordovaSplashscreen) {
  $ionicPlatform.ready(function() {

    setTimeout(function() {
        $cordovaSplashscreen.hide();
    }, 5000);

    if(window.cordova && window.cordova.plugins.Keyboard) {
        cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
    //Abre o banco e cria as tabelas
    db = $cordovaSQLite.openDB("upmoney.db");
    $cordovaSQLite.execute(db, "CREATE TABLE IF NOT EXISTS CATEGORIAS(IDCATEGORIA INTEGER PRIMARY KEY AUTOINCREMENT,DESCRICAO VARCHAR(150) NOT NULL);");
    $cordovaSQLite.execute(db, "CREATE TABLE IF NOT EXISTS CONTAS(IDCONTA INTEGER PRIMARY KEY AUTOINCREMENT,DESCRICAO VARCHAR(150) NOT NULL);");
    var commandSQL = "CREATE TABLE IF NOT EXISTS LANCAMENTOS("+
                     "      IDLANCAMENTO INTEGER PRIMARY KEY AUTOINCREMENT,"+
                     "      IDCATEGORIA INTEGER NOT NULL,"+
                     "      IDCONTA INTEGER NOT NULL,"+
                     "      TIPO VARCHAR(1) NOT NULL,"+ /*R - RECEITAS/D - DESPESAS*/
                     "      DATA DATETIME NOT NULL,"+
                     "      DESCRICAO TEXT NOT NULL,"+
                     "      CONSOLIDADO VARCHAR(1) DEFAULT 'N',"+ /*S - SIM/N - NÃO*/
                     "      VALOR NUMERIC(16,2) NOT NULL,"+
                     "      FOREIGN KEY(IDCATEGORIA) REFERENCES CATEGORIAS(IDCATEGORIA),"+
                     "      FOREIGN KEY(IDCONTA) REFERENCES CONTAS(IDCONTA));";
    $cordovaSQLite.execute(db, commandSQL);
    $cordovaSQLite.execute(db, "CREATE TABLE IF NOT EXISTS PARAMETROS(EXIGESENHA VARCHAR(1) DEFAULT 'S',SENHA TEXT);");
  });
});
```

Fonte: Do autor.

Seguindo as boas práticas de programação e buscando sempre minimizar ao máximo a repetição de códigos, para criar as telas foi optado por usar um *template abstract* que contém os menus. Dentro deste arquivo, o *menu.tpl.html*, foi adicionado, além dos códigos responsáveis pela geração do mesmo, uma diretiva do framework chamada de *ion-nav-view*, conforme pode ser visto na figura 16. Ela é responsável por embutir parte de código html sem a necessidade de reescrever todo o código do menu em cada uma das telas do aplicativo.

Figura 11 – *Template abstract* do menu

```
<ion-side-menus enable-menu-with-back-views="false">
    <ion-side-menu-content>
        <ion-nav-bar class="bar-positive">
            <ion-nav-back-button>
            </ion-nav-back-button>

            <ion-nav-buttons side="left">
                <button class="button button-icon button-clear ion-navicon" menu-toggle="left">
                </button>
            </ion-nav-buttons>
        </ion-nav-bar>

        <ion-nav-view name="menuContent"></ion-nav-view>
    </ion-side-menu-content>

    <ion-side-menu side="left">
        <ion-header-bar class="bar-stable">
            <h1 class="title">Menu</h1>
        </ion-header-bar>
        <ion-content>
            <ul class="list">
                <a href="#/main/home" class="item" menu-close><i class="icon ion-ios-home"></i> Home</a>
                <a href="#/main/listacontas" class="item" menu-close><i class="icon ion-ios-filing"></i> Contas</a>
                <a href="#/main/listacategorias" class="item" menu-close><i class="icon ion-ios-pricetags"></i> Categorias</a>
                <a href="#/main/listalancamentos" class="item" menu-close><i class="icon ion-ios-compose"></i> Lançamentos</a>
                <a href="#/main/listagraficos" class="item" menu-close><i class="icon ion-arrow-graph-up-right"></i> Gráficos</a>
            </ul>
        </ion-content>
    </ion-side-menu>
</ion-side-menus>
```

Fonte: Do autor.

Em seguida foram criados os *templates* responsáveis pelas telas de listagem, cadastro, gráficos, entre outra. Seguindo o padrão MVC, as mesmas foram adicionadas na pasta *view*. Para que elas funcionassem dentro do menu, conforme mencionado acima, todo o código está dentro de uma diretiva chamada de *ion-view*, conforme pode ser observado na figura 12.

Figura 12 – *Template* usando o *ion-view*

```
<ion-view view-title="Contas">
    <ion-nav-buttons side="right">
        <botao-adicionar url="adicionaconta"></botao-adicionar>
    </ion-nav-buttons>
    <ion-content ng-controller="contaController">
        <div class="list" ng-show="contas.length">
            <div class="item item-button-right" ng-repeat="conta in contas">
                {{conta.DESCRICAO}}
                <button class="button button-assertive" ng-click="deletarConta(conta)">
                    <i class="icon ion-ios-trash-outline"></i>
                </button>
            </div>
        </div>
        <label ng-hide="contas.length > 0">Nenhuma conta cadastrada!</label><BR />
    </ion-content>
</ion-view>
```

Fonte: Do autor.

Outra técnica utilizada para não reescrever códigos, foi a criação de uma diretiva do Angular. Ela permite que sejam criadas *tags* personalizadas a partir de uma *template*, que pode ser codificado dentro do próprio arquivo *js* onde está sendo criada a diretiva ou em um arquivo externo. Para o protótipo, foi criado a diretiva *botaoAdicionar*. Ela é responsável por generalizar todo código *html* contido no botão de adicionar que está presente em todas as telas de cadastro. Por convenção do Angular, o uso da *tag* no código sempre será separada por traço. Na figura 13 é possível observar o código onde é criado a diretiva e que o nome dela, de fato, não possui traços. Já na figura 12, mostrada acima, observa-se que ela está sendo chamada como *botao-adicionar*. Outro fato muito relevante na criação das diretivas, é a possibilidade de receber atributos. Nesse caso está recebendo o atributo *url*, onde na chamada passamos qual a *url* que deve ser direcionada ao clicar no botão.

Figura 13 – Criação de uma diretiva

```
app.directive('botaoAdicionar',  function() {
    return {
        restrict: 'E',
        scope:{
            url:'@'
        },
        template: '<a href="#/main/{{url}}"><i class="button button-icon button-clear ion-ios-plus-outline"></i></a>'
    }
});
```

Fonte: Do autor.

Depois de ter desenvolvido a parte visual, iniciou-se a codificação dos *controllers.* Para manter o padrão, estes foram adicionados dentro do diretório *controller.* Como o próprio nome sugere, eles são os controladores que fazem a interface entre a *view* e o *model.* Neles, estão contidas todas as funções que comandam as rotinas referentes ao processo pelo qual são responsáveis. Como a estrutura é baseada no Angular, a criação dele usa o comando *controller*, onde deve ser informado o nome. Na função executada por eles, sempre deve ser informado a variável global *$scope*. Nessa variável estão contidos todos os objetos criados dentro do aplicativo. Ele é responsável pelo *databinding* entre *view* e *controller*, pelo uso das funções dentro da *view*, enfim, é o cerne da aplicação. Também devem ser informados os *plug-ins* que serão utilizados e, caso exista, os módulos criados. A figura 14 demonstra a estrutura de um *controller.*

Figura 14 – Estrutura de um *controller*

```
app.controller('contaController', function($scope,$ionicPlatform,$cordovaDialogs,$state,Conta) {
    $scope.contas = [];
    $scope.contas = null;

    $scope.getContas = function(){
        Conta.all().then(function(contas){
            $scope.contas = contas;
        });
    }

    $scope.inserirConta = function(conta){
        Conta.add(conta);
        $state.go('main.listacontas');
    }

    $scope.deletarConta = function(conta) {
        $cordovaDialogs.confirm('Apagar '+conta.DESCRICAO+'?', 'Confirmação', ['Cancelar','Confirmar'])
            .then(function(buttonIndex) {
                if(buttonIndex == 2){
                    Conta.remove(conta);
                    $scope.getContas();
                }
            });
    }

    $ionicPlatform.ready(function() {
        $scope.getContas();
    });
});
```

Fonte: Do autor.

Para deixar o código mais limpo e menos trabalhoso para manutenções futuras, foi criado um módulo de manipulação de dados genérico e, estendendo este, um para cada cadastro. Nos parâmetros informados na construção do *controller* demonstrado na figura acima, é possível verificar que é passado o parâmetro *Conta.* Ele contém todas as funções para consultar, inserir, excluir e deletar contas. Na figura 15 é demonstrado como ficou a estrutura desse módulo.

Figura 15 – Estrutura do arquivo CRUD de contas

```
.factory('Conta', function(DB) {
    var self = this;

    self.all = function() {
        return DB.query('SELECT * FROM CONTAS')
            .then(function(result){
                return DB.fetchAll(result);
            });
    };

    self.getById = function(id) {
        return DB.query('SELECT * FROM CONTAS WHERE IDCONTA = ?', [id])
            .then(function(result){
                return DB.fetch(result);
            });
    };

    self.add = function(conta) {
        var parameters = [conta.IDCONTA, conta.DESCRICAO];
        return DB.query("INSERT INTO CONTAS (IDCONTA, DESCRICAO) VALUES (?,?)", parameters);
    }

    self.remove = function(conta) {
        var parameters = [conta.IDCONTA];
        return DB.query("DELETE FROM CONTAS WHERE IDCONTA = (?)", parameters);
    }

    self.update = function(contaorig, contaalter) {
        var parameters = [contaorig.IDCONTA, contaalter.DESCRICAO, contaorig.IDCONTA];
        return DB.query("UPDATE CONTAS SET IDCONTA = (?), DESCRICAO = (?) WHERE IDCONTA = (?)", parameters);
    }

    return self;
})
```

Fonte: Do autor.

Fica claro identificar que os códigos SQLs são parametrizados e genéricos, de modo que basta passar a entidade conta. Também é possível identificar o parâmetro DB sendo passado na função. Nele é que está contido a interface com o banco de dados. A função *query* executa um comando e as funções *fetch* e *fetchAll* transformam o resultado em uma lista *JSON*. Na figura 16 é possível identificar a estrutura do *DB*.

Figura 16 – Estrutura do arquivo DB

```
angular.module('upmoney.services',['ionic','ngCordova'])
.factory('DB', function($cordovaSQLite, $q, $ionicPlatform) {
    var self = this;
    //Executa os comandos
    self.query = function (query, parameters) {
        parameters = parameters || [];
        var q = $q.defer();

        $ionicPlatform.ready(function () {
            $cordovaSQLite.execute(db, query, parameters)
                .then(function (result) {
                    q.resolve(result);
                }, function (error) {
                    console.warn('Um erro foi encontrado');
                    console.warn(error);
                    q.reject(error);
                });
        });
        return q.promise;
    }
    //Busca todos os registros
    self.fetchAll = function(result) {
        var output = [];
        for (var i = 0; i < result.rows.length; i++) {
            output.push(result.rows.item(i));
        }
        return output;
    }
    //Busca o primeiro registro
    self.fetch = function(result) {
        var output = null;
        output = angular.copy(result.rows.item(0));
        return output;
    }

    return self;
})
```

Fonte: Do autor.

Também é possível identificar que ele está atribuído a um módulo, cujo nome é *upmoney.service*, que foi injetado na criação do *app*, conforme visualizado na figura 10. O Angular permite separar cada *factory* em um módulo distinto. Como seriam muito módulos para injetar, decidiu-se criar as responsáveis pelos cadastros dentro deste. Por isso que nos *controllers* é passado apenas o parâmetro Conta, pois o módulo à que ela pertence já foi injetado. Observa-se também nessa figura, que foram passados dois *plugins* do Ionic. O *$cordovaSQLite,* responsável pela integração e manipulação de dados no banco *SQLite*, e o *$ionicPlatform*, que possuí ligação com a plataforma em que está sendo executado. Nesse caso, estamos usando para saber quando a função está sendo lida.

Chegando ao final do desenvolvido, foi criado o arquivo *route.js*, que defini as rotas dos *links* executados. Para isso, é usado o comando *config* do Angular e usado os *plugins $stateProvider* e *$urlRouterProvider.* O primeiro é

responsável por ligar a *view* com seu respectivo *controller* e lança-los de acordo com a *url* determinada. Já o *$urlRouterProvider*, é usado para os casos onde a *url* que chega não é reconhecida, sendo como uma rota *default.* A figura 17 demonstra uma parte das configurações de rota do aplicativo.

Figura 17 – Configurações de rota

```
app.config(function($stateProvider, $urlRouterProvider) {
    $stateProvider
        .state('main', {
            url: "/main",
            abstract: true,
            templateUrl: "view/menu.tpl.html"
        })
        .state('main.home', {
            url: "/home",
            views: {
                'menuContent' :{
                    templateUrl: "view/home.tpl.html",
                    controller: "homeController"
                }
            }
        })
        .state('main.adicionaconta', {
            //cache: false,
            url: "/adicionaconta",
            views: {
                'menuContent' :{
                    templateUrl: "view/adicionaConta.tpl.html",
                    controller: "contaController"
                }
            }
        })

    $urlRouterProvider.otherwise("/main/home");
})
```

Fonte: Do autor.

No primeiro *state*, é possível observar o parâmetro *abstract: true*. Ele define que essa *view* será abstrata e todas as demais serão executadas dentro desta. Já nos demais, foi indicado um caminho - *main.home*, uma url - */home*, e no parâmetro *views* foi informado que o arquivo contido no *templateUrl* e *controller* farão parte do *menuContent.*

Após isso, foi necessário realizar a adição da plataforma, bem como o *build* do aplicativo, através do CLI do ionic, para que fosse possível verificar o funcionamento e realizar os testes.

7.1.4 Funcionamento da Aplicação

O protótipo de aplicação é composta por um conjunto de telas com finalidades especificas. Existem telas para a visualização de informações, listagem de dados cadastrados, e as telas de cadastramento. Além disto, utiliza o conceito de menus, então todas as telas são acessadas através dele.

Seguindo uma ordem, inicialmente, tem-se a página inicial, que é compostas por dois quadros com informações, um com o saldo das contas e o outro com o total de despesas e receitas. Sua principal funcionalidade é demonstrar ao usuário uma visão geral de todos os gastos e ganhos que ele obteve até o momento.

A tela de listagem de contas, serve para demonstrar todas as contas cadastradas. Além disto, a lista permite que sejam excluídas as informações, e em seu cabeçalho dá acesso ao cadastramento de novas contas. Essa tela de cadastro possui apenas um campo de descrição.

Assim como a funcionalidade das contas, a tela de listagem das categorias faz a exibição de todas as categorias, que representem algo com que o usuário realizou um gasto ou recebeu uma receita. Possibilita também a exclusão dos itens de lista e a inserção de novas informações. Exemplos padrões de categorias são: alimentação, estudos, vestuário, salário e entre outros.

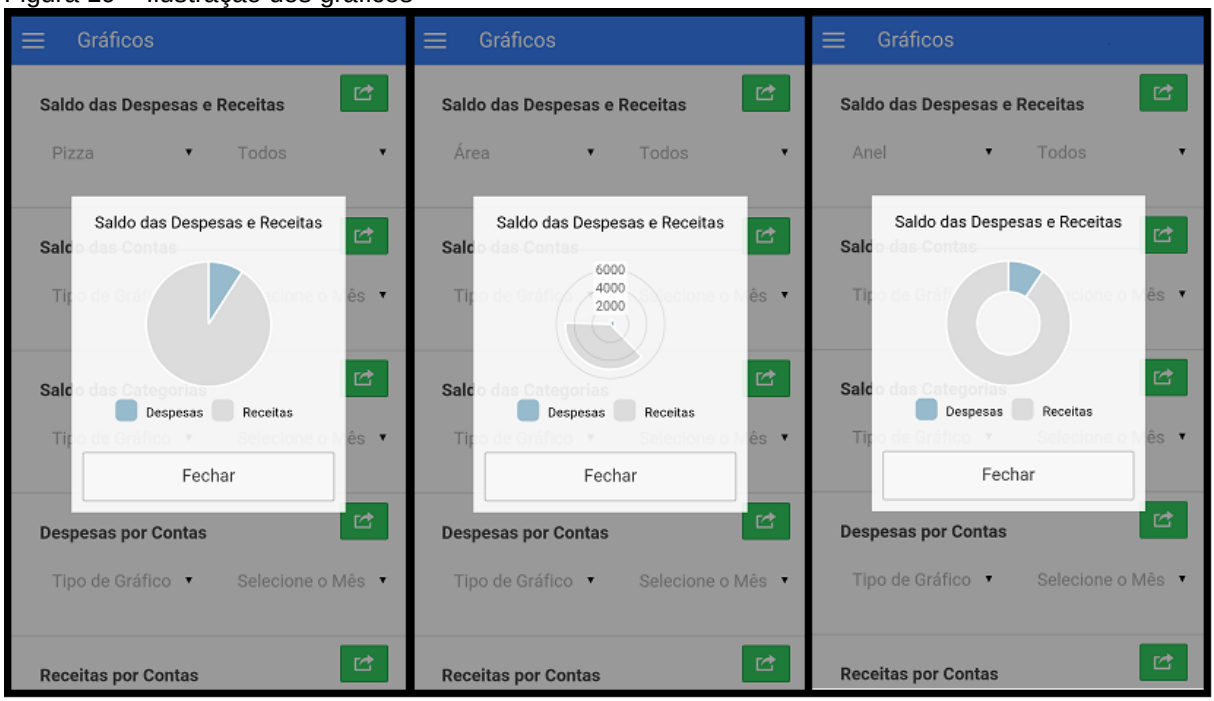
A principal tela da aplicação que é responsável pela movimentação das informações é a tela de lançamentos, que simplifica o cadastramento de receitas e despesas. Assim como as contas e categorias, essa tela faz a listagem dos lançamentos que foram cadastrados, porém, exibe além da descrição, se é uma despesa ou receita e o valor, permitindo também a exclusão e inserção de novos lançamentos. Seu cadastro possui os campos valor, descrição, data, tipo de lançamento, categoria e conta.

Figura 18 – Telas de listagem de informações

Fonte: Do autor.

A última tela é a de gráficos, que possui uma listagem com sete gráficos específicos. São eles: saldo das despesas e receitas, saldo das contas, saldo das categorias, despesas por contas, receitas por contas, despesas por categorias, receitas por categorias. É possível, exibir os gráficos com três formatos, pizza, anel e por área, respectivamente, bem como selecionar o mês para qual se deseja visualizar a informação. Ao emitir é exibido um popup no centro da tela com o gráfico.

Figura 19 – Ilustração dos gráficos

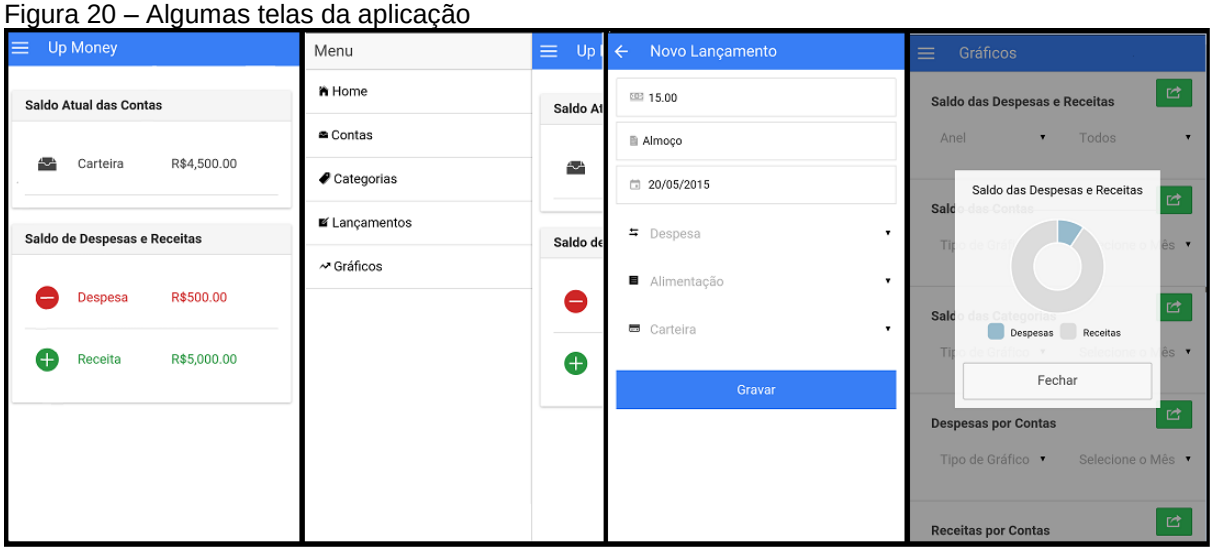


Fonte: Do autor.

7.1.5 Testes da Aplicação

Após concluída a fase de desenvolvimento do protótipo, foi realizada a etapa de testes. Para tal procedimento, inicialmente, foi averiguado se os menus estavam dando acesso aos cadastros correspondentes a seus nomes. Logo após foram cadastradas as contas, categorias e lançamentos para verificar se o funcionamento cadastral estava correto. Em seguida com as informações cadastradas foram executados testes com base nas listas com o intuito de analisar o processo de atualização e exclusão de itens. Imediatamente, realizou-se a emissão dos gráficos a fim de que o usuário possa ter uma breve amostra acerca de suas informações. A figura 20 demonstra algumas telas da aplicação:

Figura 20 – Algumas telas da aplicação

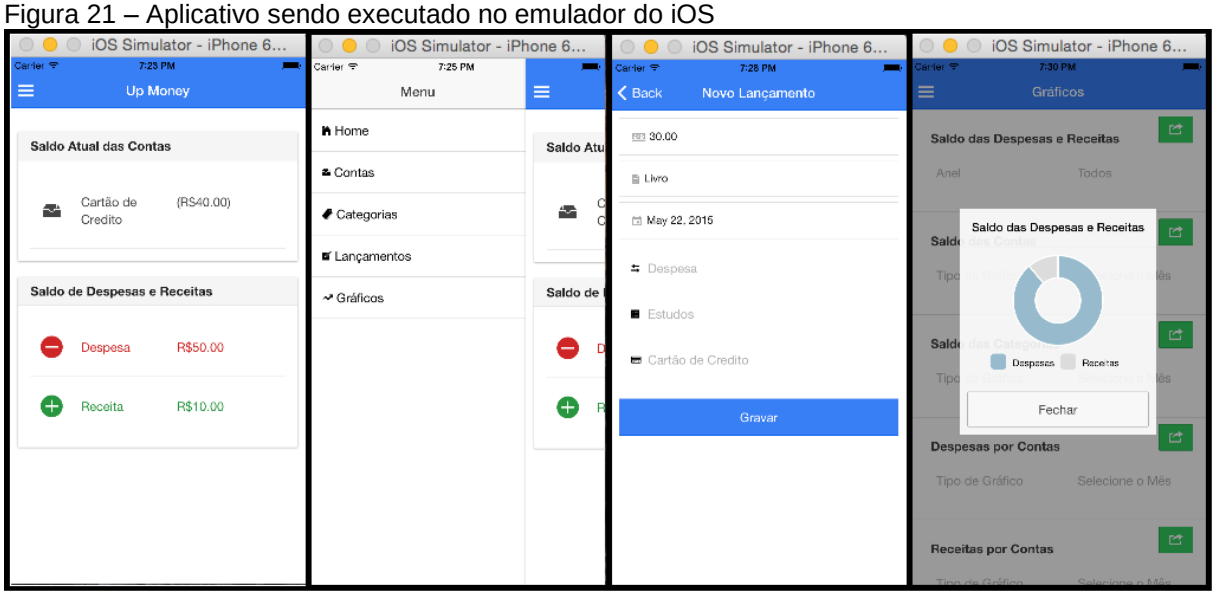


Fonte: Do autor.

No contexto atual, pretendeu-se fazer a execução dos testes em outra plataforma a fim de validar o conceito da aplicação híbrida, e verificar se as funcionalidades teriam o mesmo resultado apresentado na outra plataforma. Para este procedimento, foi utilizada a ferramenta *VMWare Player* 7 versão para uso individual, para criar uma máquina virtual com o sistema operacional *Mac OS X Yosemite* 10.10, uma vez que para a realização do *build* para o iOS é necessário o SDK da Apple que funciona exclusivamente em seu sistema operacional.

Através disto o projeto foi copiado para a máquina virtual e utilizou-se os mesmo procedimentos que os da plataforma Android. Adicionando a plataforma do iOS, realizando o build e executando através do emulador.

Figura 21 – Aplicativo sendo executado no emulador do iOS



Fonte: Do autor.

7.2 RESULTADOS OBTIDOS

Com a realização deste estudo a respeito de *frameworks* multiplataforma *mobile*, foi possível a criação de um protótipo de aplicação híbrida para o controle financeiro, utilizando o framework Ionic.

Os resultados obtidos com essa aplicação são satisfatórios, e algumas funcionalidades podem comprovar que este framework apresenta um diferencial diante dos outros, uma vez que seus recursos, facilitaram a linha de aprendizado e o desenvolvimento da aplicação.

O fator relevante deste estudo, está no fato do *AngularJs*, não proporcionar somente um melhor desempenho das aplicação, mas também, pelo fato de permitir que a funções e otimização de código fonte seja ampla, onde através de seus injetores, permite a criação de serviços, como por exemplo, para as funções de CRUD (*Create Ready Update Delete*) do banco de dados conforme mencionado no item 7.1.3.

Além deste fator, as diretivas que o framework possui, reduzem o tempo, e o escopo de desenvolvimento do layout da aplicação, uma vez que não é necessário reescrever todos os cabeçalhos, menus e informações que se repetem

dentro de um *template*. Outra observação a ser feita, é que seu CSS, possibilitou a criação de telas mais atraentes, e graças a documentação do framework não se obteve dificuldade para a criação do layout.

Importante ressaltar que o uso dos *plugins* para a plataforma são compatíveis para ambas, tanto para o Android quanto para o iOS, sendo assim os gráficos tiveram o mesmo funcionamento entre ambas as plataformas, demonstrando que não são apenas funcionalidades cadastrais que obtém um bom funcionamento.

Outra grande diferencial levantado, foi que devido ao CLI, a criação de versões para outras plataformas é reduzida a uma única linha de comando, uma vez que o *framework* não gera a portabilidade entre plataformas, mais sim aproveita o código fonte escrito em linguagem web, e o compila e faz a interpretação do *webview* que é criado dentro da aplicação. Diante deste motivo e graças a interação entre *AngularJs* e o *HTML*, as aplicações criadas possuem um desempenho diferenciado dos demais *frameworks*.

Um diferencial proporcionado ainda pelo *framework*, é que seu CLI, possui um comando que gera o código de segurança das aplicações para que as mesma possam ser publicadas nas lojas virtuais. Ainda pra ajudar em questão de performance existe um comando que proporciona que a arquivo para instalação final da aplicação seja otimizado, melhora ainda mais as questões de desempenho.

Sendo assim, atingiu-se os objetivos propostos, demonstrando alguns benefícios do desenvolvimento de uma aplicação *mobile* multiplataforma, com o desenvolvendo um protótipo de aplicação móvel híbrida para o controle financeiro pessoal utilizando o *framework* Ionic

8 CONCLUSÃO

Com a constante expansão tecnológica, é possível perceber o crescimento dos dispositivos móveis, aplicações e meios para o desenvolvimento de novas funcionalidades. Os celulares, tablets e smartphones estão se tornando cada vez mais uma peça chave na vida das pessoas. Oferece uma vasta quantidade de vantagens e benefícios de fácil acesso aos usuários. Isso faz com que os desenvolvedores cada vez mais, busquem por aperfeiçoar seu conhecimento perante a este avanço.

Perante a esta situação, foram surgindo novas ferramentas que auxiliam o desenvolvimento de aplicações, e que reduzissem o custo de desenvolvimento e o número de manutenções, bem como a necessidade de reescrever o código fonte, uma vez que, se tinha necessidade que a aplicação fosse replicada para outra plataforma.

Diante disto, optou-se por realizar o desenvolvimento de uma aplicação mobile híbrida para o controle financeiro utilizando um *framework* multiplataforma especifico. Conclui-se que com a vasta quantidade de frameworks existentes atualmente, é realmente difícil escolher um *framework* para o desenvolvimento de aplicações *mobile* híbridas. Sendo assim com pesquisas já realizadas e com base nas funcionalidades dos *frameworks* optou-se por utilizar o Ionic.

Ao final deste projeto de pesquisa os objetivos foram alcançados, o que possibilitou um maior conhecimento sobre *frameworks* multiplataformas e aplicações híbridas e principalmente acerca do Ionic. Foi constatado ainda, que o *framework* Ionic tem uma vasta quantidade de recursos que possibilita o desenvolvimento de uma aplicação dinâmica, e de forma mais ágil que as demais, sendo que a linha de aprendizado adquirida, pode ser utilizada também, para a criação de aplicações web, uma vez que esses *frameworks* utilizam essa tecnologia.

Durante a pesquisa encontraram-se algumas dificuldades, porém, a principal foi na fundamentação teórica a respeito dos *frameworks* multiplataformas e referente ao Ionic, uma vez que este *framework* é muito novo, e existe pouca documentação para compreender sua criação e sua composição.

Este estudo abre caminho para trabalhos futuros, tais como a implantação de um protótipo de aplicação híbrida integrado a um *webservice,* utilizando por exemplo, o *framework* Ionic, fazendo com que a aplicação funcione off-

line, porém, ao conectar-se com a rede as informações são enviadas ao servidor automaticamente. Desta forma seria possível desenvolver em conjunto uma aplicação *web* que receberia as informações enviadas ao *webservice*. Outra opção seria realizar um estudo de como utilizar um framework *multiplataforma* para a criação de testes automatizados para aplicações mobile, uma vez que não seria necessário reescrever o código de automação para outra plataforma. Como terceira opção seria possível realizar um comparativo entre o desempenho nativo versus híbrido a respeito de jogos, recriando um jogo já existente e vendo como o mesmo iria se comportar em várias plataformas.

REFERÊNCIAS

ABLESON. W. Frank et tal. **Android em Ação.** 3. Rio de Janeiro: Campus, 2012. 656 p.

ALLEN, Sarah; GRAUPERA, Vidal; LUNDRIGAN, Lee. **Desenvolvimento Profissional Multiplataforma para Smartphone, Iphone, Android, Windows Mobile e Blackberry.** Rio de Janeiro: Alta Books, 2012, 280p.

ADOBE. **Adobe AIR Documentation**. Disponível em: <http://www.adobe.com/devnet/air/documentation.html> Acesso em: 20 maio 2015

APACHE CORDOVA. **Cordova.** Disponível em: <https://cordova.apache.org/> Acesso em 12 abr. 2015.

APPCELERATOR. **Appcelerator Platform Documentation**. Disponível em: <https://docs.appcelerator.com/> Acesso em: 20 maio 2015

APPLE. **iOSDev Center**. 2015. Disponível em: <https://developer.apple.com/devcenter/ios/>. Acesso em: 18 maio 2015.

BOWER. **Bower.** Disponível em: <http://bower.io/>. Acesso em: 14 maio 2015.

CAFÉ, Adriel Almeida. **Desenvolvimento de Cross-Platform Mobile Apps Utilizando o Titanium Mobile**. 2012. 81 f. Trabalho de Conclusão de Curso (Bacharel em Sistema de Inforação) – Faculdade Zacarias de Góes, Bahia.

CAVAZZA, Fred. **Mobile Web App vs. Native App? It's Complicated.** Forbes, set. 2011, Disponível em: < http://www.forbes.com/sites/pega/2015/06/03/disruptors-wanted-how-successful-companies-view-employees-and-customers/. Acesso em: 20 nov. 2014

CERBASI, Gustavo. **Casais inteligentes enriquecem juntos:** finanças pessoais. São Paulo: Gente, 2012. 163 p.

CERBASI, Gustavo. **Casais inteligentes enriquecem juntos:** finanças para casais. São Paulo: Gente, 2004. 192 p.

CHEROBIM A. P. M. S; ESPEJO M.M.S.B. **Finanças pessoais:** conhecer para enriquecer. São Paulo: Atlas, 2010. 147 p.

DEVELOPER ANDROID. **API Guide.** Disponível em:<http://developer.android.com/guide/components/index.html>. Acesso em: 10 nov. 2014.

DEVMEDIA. **Aplicações Móveis**: Nativas ou Web?. Disponível em: <http://www.devmedia.com.br/aplicacoes-moveis-nativas-ou-web/30392> Acesso em: 5 maio 2015.

DOCS ANGULAR. **API Guide.** Disponível em:<https://docs.angularjs.org/guide/>. Acesso em: 11 abr. 2015.

FRANKENBERG, L**. Seu futuro financeiro**. 8. ed. Rio de Janeiro: Campus, 1999

GARGENTA, Marko. **Learning Android.** Sebastopol**:** O'reilly, 2011. 266 p.

GOK, Nizametin; KHANNA, Nitin. **Building Hybrid Apps with Java and JavaScript**. Sebastopol,CA: O'Reilly, 2013. 156p.

HALFELD, Mauro. **Investimentos:** como administrar melhor seu dinheiro. São Paulo: Fundamento educacional, 2001. 142 p.

IONIC. **Ionic Documentation.** Disponivél em: <http://ionicframework.com/docs/> Acesso em: 5 maio 2015

LECHETA, Ricardo R., **Google Android:** aprenda a criar aplicações para dispositivos móveis com Android SDK. 2. São Paulo: Novatec, 2010.

LECHETA, Ricardo R. **Desenvolvendo para iPhone e iPad**. 2. São Paulo: Novatec Editora, 2012. 749 p.

LUQUET, G. H. **Guia valor econômico de finanças pessoais**. São Paulo: Editora Globo, 2000. 144 p.

MENDES, Mariana Ribeiro; GARBAZZA, Itagildo Edmar; TERRA, Daniela Costa. Desenvolvimento híbrido versus desenvolvimento nativo de aplicativos móveis. In: VII Semana de Ciência e Tecnologia do IFMG campus Bambuí VII Jornada Científica e I Mostra de Extensão, 2014, Minas Gerais, **Tópico temático**. São Paulo: Novatec, 2014. p. 25-30

MYSQL WORKBENCH. **My SQL Products.** Disponível em:< https://www.mysql.com/products/workbench/>. Acesso em: 14 maio 2015.

NEUBURG, Matt. **iOS7Programming Fundamentals**. Sebastopol,CA: O'Reilly, 2014. 422 p.

NG CORDOVA. **Cordova with the power of AngularJS.** Disponível em:<http://ngcordova.com/>. Acesso em: 14 maio 2015.

NODEBR. **O que é node js.** Disponivel em : < http://nodebr.com/o-que-e-node-js/ >. Acesso em: 11 abr. 2015.

NODEBR NPM. **O que é a npm do nodejs.** Disponível em : < http://nodebr.com/o-que-e-a-npm-do-nodejs/>. Acesso em: 11 abr. 2015.

OPEN HANDSET ALLIANCE. **Android.** Disponível em: <http://www.openhandsetalliance.com/android_overview.html>. Acesso em: 10 out. 2014.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android para desenvolvedores**. Rio de Janeiro: Brasport, 2009.

PHAN, Hoc. **Full Stack Mobile App with Ionic Framework**. Kindle Edition, 2014. 96p.

PHONEGAP. **Phonegap Documentation**. Disponível em: <http://docs.phonegap.com/> Acesso em: 20 maio 2015

PREZZOTO, Ezequiel Douglas; BONIATI, Bruno Batista. Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas. In: Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação, 2014, Rio Grande do Sul, **Anais do EATI**. São Paulo: Novatec, 2014. p. 72-79

PUGLIESI, Nataly. **6 apps para cuidas das finanças pessoais.** Você S/A, jul. 2013, Disponível em:<http://exame.abril.com.br/revista-voce-sa/edicoes/181/noticias/planilha-de-bolso/> Acesso em: 20 nov. 2014.

RAJ, Jay. **The Top 7 Hybrid Mobile App Frameworks.** Sidepoint, nov. 2014, Disponivel em: < http://www.sitepoint.com/top-7-hybrid-mobile-app-frameworks/>

ROCHA, Ricardo Humberto; SOUSA, Almir Ferreira de; TORRALVO, Caio Gragata. **Planejamento Financeiro Pessoal e Gestão do Patrimônio.** São Paulo: Atlas, 2012. 168 p.

SANDRONI, P. **Dicionário de administração e finanças.** Rio de Janeiro: Record, 2008. 527 p.

TELECO. **Mercado mundial de smartphones.** Disponível em:<http://www.teleco.com.br/sist_operacional.asp>. Acesso em: 6 nov. 2014.

VISION MOBILE. **Developer Economics 2013**: The tools report. Disponível em: <http://www.visionmobile.com/product/developer-economics-2013-the-tools-report/> Acesso em: 20 maio 2015

W3S. **W3School Online.** Disponível em: <http://www.w3schools.com/html/default.asp> Acesso em: 5 maio 2015a.

W3S. **W3School Online.** Disponível em: <http://www.w3schools.com/css/default.asp> Acesso em: 5 maio 2015b.

W3S. **W3School Online.** Disponível em: <http://www.w3schools.com/js/default.asp> Acesso em: 5 maio 2015c.

WILKEN, Jeremy. **Ionic in Action.** Estados Unidos: Manning Early Access Program, 2015. 325 p.

ZIFLAJ, Aldo. **Native App vs Hybrid App Development.** SidePoint, aug. 2014, Disponível em:< http://www.sitepoint.com/native-vs-hybrid-app-development/ Acesso em: 8 maio 2015.

APÊNDICE(S)

APÊNDICE A – ARTIGO CIENTÍFICO

Estudo de Framework Multiplataforma Aplicado ao Desenvolvimento de um Protótipo de Aplicativo Mobile Híbrido para o Controle Financeiro Pessoal

Muriel R. Mazuchetti[1], Gilberto Vieira da Silva[2]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC) Av. Universitária, 1105 - Bairro Universitário – Criciúma – SC – Brasil

[2]Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC) Av. Universitária, 1105 - Bairro Universitário – Criciúma – SC – Brasil

muriel.rampinelli@gmail.com, gilbertovieirasilva@hotmail.com

Abstract. *Mobile platforms provide several features that enable developers to create new features. With the need of the same application is used for more than one platform, originated from the multi-platform applications. For the development of these applications are used frameworks composed of web language. Based on this, this paper presents the development of a hybrid mobile application prototype for financial control, using the Ionic framework. It chose an application to the area of finance, because experts say that using technological resources ensures a complete and accurate historical information*

Resumo. *As plataformas móveis disponibilizam vários recursos que possibilitam que os desenvolvedores criem novas funcionalidades. Com a necessidade de uma mesma aplicação ser utilizada para mais de uma plataforma, originou-se, as aplicações multiplataformas. Para o desenvolvimento dessas aplicações são utilizados frameworks composto por linguagem web. Baseado nisto, o presente trabalho apresenta o desenvolvimento de um protótipo de aplicação mobile híbrida para o controle financeiro, utilizando o framework Ionic. Optou-se por uma aplicação para a área de finanças, pois os especialistas afirmam que utilizar recursos tecnológicos garante um histórico completo e exato das informações.*

1. Introdução

As plataformas móveis sempre tiverem uma visão ampla de crescimento e estão se tornando cada vez mais populares. O uso de celulares, tablets e smartphones conquistou grande parte das pessoas e nota-se que estão mais presentes no cotidiano dos indivíduos, sendo que a capacidade de processamento destes está sempre em expansão.

As mais populares são Android e iOS e foram as principais responsáveis pela maioria dos embarques de smartphones no terceiro trimestre de 2014. O Android lidera com 84% da quota do mercado contra 11,7% do iOS. Elas oferecem ainda uma vasta quantidade de benefícios, possibilitando que seja possível tirar total proveito de um aparelho portátil (TELECO, 2014).

Os aplicativos para dispositivos móveis estão em constante crescimento. A cada dia é possível perceber o surgimento de novidades nas lojas virtuais de smartphones. É comum que existam dúvidas na escolha de um meio adequado para desenvolver as aplicações móveis, umas vez que cada plataforma fornece uma ferramenta nativa para este proposito (CAVAZZA, 2011, tradução nossa).

Atualmente existem dois tipos de aplicações mobile, as nativas e as multiplataformas. As nativas são criadas exclusivamente para uma determinada plataforma utilizando linguagem de programação homologada pela mesma, já as multiplataformas podem ser executadas por qualquer plataforma mobile, além disto, esses tipos de aplicações são subdivididas em *webapps* que são desenvolvidas com linguagem *web* e executadas através do *browser* totalmente dependente da internet e as aplicações híbridas que contem a versatilidade das aplicações *web*, basicamente uma instancia do *browser*, que é executada compilada e interpretada como uma aplicação nativa (DEVMEDIA, 2015).

Com esse avanço tecnológico originaram-se os *frameworks* para a criação de aplicações híbridas. É possível citar como exemplo, *Sencha, PhoneGap, Titanium, Rhomobile, ParticleCode* e entre outros. O uso desses recursos acaba propiciando vantagens como, por exemplo, redução no tempo de desenvolvimento, maior custo benefício, boa experiência de usuário, facilidade de manutenção. Porém, de contraposto notava-se que os aplicativos híbridos eram relativamente lentos comparados aos nativos (CAVAZZA, 2011, tradução nossa).

Segundo Wilken (2015), com o surgimento do AngularJs, uma poderosa ferramenta JavaScript mantida pela Google, e pensando na performance das aplicações hibridas, em 2013 a empresa norte americana Drifty lançou o *framework* Ionic.

Ele é utilizado especialmente para projetar a interface do usuário. Desta forma, inclui-se todos os elementos visuais, como guias, botões, cabeçalhos de navegação, pop-ups entre outros (IONIC, 2015, tradução nossa).

Devido sua composição ser HTML, CSS e AngularJS, permite que suas aplicações tenham código fonte único, extinguindo, desta forma, a necessidade da utilização de um algoritmo para cada plataforma (WILKEN, 2015, tradução nossa).

No contexto atual, o seguinte artigo pretende abordar um estudo acerca de *frameworks* multiplataforma *mobile*, realizando o desenvolvimento de uma aplicação híbrida para o controle financeiro pessoal utilizando o *framework* Ionic, expondo também os motivos pelo qual ele foi escolhido.

Optou-se um aplicativo para a área financeira, pois alguns especialistas garantem que buscar recursos para controlar as finanças é um meio utilizado por muitas pessoas que pretendem adquirir uma margem de tranquilidade financeira (ROCHA; SOUSA; TORRALVO, 2012).

2. Plataformas Móveis

O crescimento tecnológico atualmente é constante. Cada vez mais, é possível notar mudanças no mercado de eletrônicos, porém, pode-se ressaltar que a área de dispositivos móveis foi a que mais evoluiu. Atualmente, estes dispositivos possuem um sistema operacional, uma linguagem de programação próprio e ferramentas para o desenvolvimento (TELECO, 2014).

As plataformas móveis foram criadas, por um conjunto de tecnologias, que agregam vários recursos. Com isso, os smartphones passaram a utilizar plataformas para o funcionamento. Essas plataformas tem a responsabilidade em gerenciar todas as funções de hardware de um dispositivo móvel, como gps, câmera, agenda, lista telefônica, entre outros recursos. Cabe ressaltar ainda, que elas são a principal forma de interação entre usuário e aplicações (ALLEN; GRAUPERA; LUNDRIGAN, 2012).

Logo após o surgimento dos smartphones, foi possível notar a diferença entre as tecnologias, uma vez que o smartphone mudou totalmente a percepção que se tinha a respeito da telefonia móvel, com as funcionalidades como, *bluetooth*, redes 4g, reconhecimento facial, reconhecimento biométrico entre outros (TELECO, 2014).

De acordo com a Teleco (2014), as plataformas móveis acompanharam o crescimento dos smartphones, e estão cada vez mais completas. Entre as várias existentes destacam-se Android da Google, e o iOS da Apple. Além disto, foram as principais responsáveis pela grande maioria dos embarques de smartphones no terceiro trimestre de 2014. O Android lidera com 84% da quota do mercado contra 11,7% do iOS.

Cada uma delas possui suas próprias características que são responsáveis por torna-las interessantes, não apenas para os consumidores, mas também, para os desenvolvedores. Naturalmente os aparelhos mais fáceis de utilizar e com diversos funcionalidades e com maior custo-benefício são a principal busca realizada pelos consumidores. Os desenvolvedores, por outro lado, procuram por smartphones ou tables, onde a plataforma tem maior reconhecimento de mercado e que tenha uma boa documentação, baixo custo de desenvolvimento entre outros (ALLEN; GRAUPERA; LUNDRIGAN, 2012).

Juntamente com os dispositivos e as plataformas móveis, pode-se notar uma alta gama de aplicações sendo criadas e adicionadas nas *app stores*. Essas lojas virtuais são responsáveis distribuir e vender aplicações. Cada plataforma possui sua linguagem de desenvolvimento própria que permite a criação destas aplicações e consequentemente a publicação nas lojas virtuais.

3. Aplicações Móveis

Da mesma maneira que as plataformas, os aplicativos para dispositivos móveis estão em constante crescimento. A cada dia é possível perceber o surgimento de novidades nas lojas virtuais de smartphones. É comum que muitos programadores tenham dúvida na escolha de um meio adequado para desenvolver suas aplicações móveis, já que cada plataforma fornece uma ferramenta nativa para o desenvolvimento (DEVMEDIA, 2015).

Diante deste crescimento, os desenvolvedores e as empresas buscam cada vez mais formas para reduzir manutenções e custos em suas aplicações uma vez que cada plataforma móvel possui uma linguagem de programação distinta, sendo assim não se tem um aproveitamento de código fonte obrigando os desenvolvedores a reescreverem o algoritmo das aplicações para que a mesma esteja disponível para mais de uma plataforma (DEVMEDIA, 2015)

Atualmente existem três tipos de aplicações, são elas nativas, *webapps* ou *mobile web sites* e, híbridas. Os subcapítulos a seguir irão descrever um pouco melhor cada uma delas.

Figure 1. Comparativo entre tipos de aplicação

3.1 APLICAÇÕES NATIVAS

As aplicações móveis nativas são desenvolvidas com o intuito de serem executadas em uma plataforma específica, desta forma, são instaladas diretamente no sistema operacional dos dispositivos e seu funcionamento não necessita de acesso à rede, ou seja, de modo *off-line*. Cada aplicação nativa é desenvolvida em linguagem padrão para determinada plataforma, como por exemplo, Java para o Android da Google, e *Objective-C* e *Swifty* para iOS da *Apple*, nesse tipo de aplicação é possível contar com os recursos distintos disponibilizados pela própria plataforma, como por exemplo, acelerômetro, gps, câmera entre outros (DEVMEDIA, 2015).

Apesar do desempenho dessas aplicações ser muito alto, é necessário que o desenvolvedor tenha conhecimento da linguagem de programação que cada plataforma necessita para a criação de aplicações. Diante disto, o custo benefício de desenvolvimento torna-se menor, uma vez que, exista a necessidade do aplicativo ser executado em demais plataformas, dessa forma o código fonte deve ser escrito novamente. Perante isto, o tempo de aprendizado é maior, devido ao fato de que é necessário obter o conhecimento acerca da linguagem de programação de cada plataforma (DEVMEDIA, 2015).

3.2 WEBAPPS

Os *webapps* ou *mobile web sites*, são aplicativos que tem um bom funcionamento em um dispositivo móvel, mas são acessadas somente através do *browser*. A maioria destes aplicativos são sites acessados de forma ágil, com a exceção de ter um layout projetado para o formato das telas dos dispositivos móveis. Além disto, não permitem que a utilização de recursos nativos, como gps, câmera entre outros. Este tipo de aplicação possui algumas dependências, como por exemplo, o acesso a uma conexão de rede, caso contrário não é possível utiliza-lo (WILKEN, 2015, tradução nossa).

Diferentemente das aplicações nativas, não existe a necessidade de uma instalação, já que as mesmas são acessadas através do *browser*. Um dos grandes fatores que influencia no funcionamento dessas aplicações, é a conexão estável com a internet, uma vez que ela é a principal responsável por garantir velocidade e disponibilidade da aplicação (WILKEN, 2015, tradução nossa).

De acordo com Wilken (2015), este modelo de aplicações é multiplataforma, é composto por HTML, CSS e JavaScript, além disto conta também com algumas ferramentas *server-side*, como Java ou PHP.

3.3 APLICAÇÕES HÍBRIDAS

Esse tipo de aplicação é chamado de multiplataforma. Como o próprio nome diz, permite que os *apps* sejam executados em diversas plataformas. As aplicações hibridas são formadas por uma combinação de *Web Apps* e aplicações nativas.

Diante disto, as aplicações híbridas contém a versatilidade das aplicações web, basicamente uma instancia do *browser*, que será executada dentro de uma aplicação nativa. Sendo assim, podem utilizar os recursos de uma página *web* permitindo também o acesso às funções nativas do dispositivo. Além disto, é bem adequado para uma ampla gama de aplicações e ainda pode fornecer uma boa experiência do usuário. Quando um aplicativo híbrido é construído, ele será compilado, transformando a aplicação web em um aplicativo nativo (WILKEN,2015, tradução nossa).

Ao contrário de aplicações web ou *WebApps*, que o usuário pode acessar, navegando através da barra de endereço, os aplicativos híbridos são normalmente instalados através de uma loja virtual e estão disponíveis através do menu principal plataforma. Isso significa que os usuários têm que seguir o mesmo procedimento de instalação de uma aplicação nativa (GOK; KHANNA, 2013, tradução nossa).

Segundo Gok e Khanna (2013), com o avanço nos sistemas operacionais móveis e motores de processamento de JavaScript, um aplicativo híbrido rodando em dispositivos móveis razoavelmente modernos podem oferecer experiências de usuário altamente eficientes usando HTML, CSS, e JavaScript para a camada de interface do usuário. Desta forma, as aplicações hibridas podem trazer muitas vantagens aos desenvolvedores, como por exemplo, a criação de uma UI genérica de acordo com a regra de negócio, reduzindo assim 50% do tempo de desenvolvimento de uma aplicação.

3.4 FRAMEWORKS MULTIPLATAFORMA

Com o crescimento das aplicações híbridas foram surgindo ferramentas que ajudassem no desenvolvimento, como por exemplo, *frameworks*. Atualmente existem várias ferramentas que realizam este trabalho, é possível citar, *Phonegap, KendoUI, Mobile Angular UI, Sencha Touch, AppCelerator Titanium* e entre outros (RAJ, 2014, tradução nossa).

Os primeiros *frameworks* foram *Phonegap* e *Cordova*, mesmo possuindo nomes diferentes ambos são a mesma coisa, a diferença entre eles é a forma de realização do build das aplicações, uma vez que o *Cordova* é a base que move o *Phonegap*. Foram responsáveis por revolucionar a forma de criação de aplicações híbridas, devido ao código fonte ser apenas *front-end*, sem a necessidade do uso da linguagem nativa (WILKEN, 2015, tradução nossa).

A maioria dos *frameworks* multiplataforma utilizam linguagem *web*, ou seja, baseada em HTML, CSS e JavaScript. As aplicações híbridas acabaram tornando-se muito poderosas, pois a linguagem *web* está evoluindo constantemente e cada vez mais é possível notar novidades para o desenvolvimento *web* e *front-end*. Um grande exemplo, é o AngularJS mantido pela Google. Este *framework* revolucionou o mundo do JavaScript, alterando completamente sua interação com o HTML (DOCS ANGULAR, 2015, tradução nossa).

Segundo Wilken (2015), com o surgimento do AngularJS, e pensando na performance das aplicações hibridas, em 2013 foi lançado o framework Ionic. Sua composição totalmente diferenciada permitiu uma grande evolução na performance das aplicações híbridas.

4. Framework Ionic

O *framework* Ionic é uma combinação de tecnologias e utilitários projetados para tornar a facilitar e agilizar a construção aplicativos móveis híbridas com destaque visual, sendo estas tecnologias, HTML, CSS e JavaScript. Embora seja um *framework*, ele não é executado de maneira independente, já que é constituído por vários componentes, sendo eles, AngularJS como *framework* de aplicação web javascript, e Cordova que é responsável pela interpretação e construção do aplicativo para as plataformas móveis.

O AngularJS é um *framework* JavaScript, declarado muito poderoso dentre os desenvolvedores que o utilizam. É chamado de linguagem declarativa, já que ele adiciona propriedades que mudam o comportamento da linguagem e possibilita um interação dinâmica com os elementos do HTML através das chamadas diretivas (DOCS ANGULAR, 2015, tradução nossa).

Construído em cima do AngularJS, o framework Ionic, é utilizado especialmente para projetar a interface do usuário, desta forma, inclui-se todos os elementos visuais, como guias, botões, cabeçalhos de navegação, pop-ups entre outros. Seus componentes são criados por uma combinação de HTML, CSS e JavaScript (AngularJS) que se comportam como os controladores nativos da plataforma (PHAN, 2014, tradução nossa).

O núcleo do Ionic são os controladores de interface do usuário que não estão presentes no HTML, mas que são comuns para aplicativos móveis. No entanto, ele inclui ainda uma série de utilitários e recursos adicionais que ajudam a gerenciar seu aplicativo desde a criação até a pré-visualização para a implantação (WILKEN, 2015, tradução nossa).

Diante do estudo levantado acerca dos frameworks multiplataforma, o capítulo a seguir traz uma breve descrição sobre os motivos pelo qual escolheu-se o desenvolvimento para a área de finanças pessoais.

5. Controle Financeiro Pessoal

Os conceitos financeiros sempre tiveram suma importância para que houvesse um desenvolvimento social e intelectual da humanidade. Com o intuito de orientar melhor os indivíduos esses conceitos utilizam técnicas especificas que permitem a gestão financeira pessoal.

Cerbasi (2004), por exemplo, instrui os leitores de seus livros, a focarem nas despesas que são pequenas e que as pessoas naturalmente não costumam a anotar. Sugere ainda a utilização de recursos tecnológicos como por exemplo, planilhas e softwares para a gestão financeira.

A busca de recursos para controlar as finanças é um meio utilizado por muitas pessoas que pretendem adquirir uma margem de tranquilidade financeira.

Segundo Torralvo, Sousa e Rocha (2012) utilizar recursos tecnológicos para controlar as finanças lhe dá vantagens, já que as informações são mais exatas e acessíveis facilitando a organização de despesas e receitas de forma precisa e direta, além de proporcionar um histórico completo sobre as finanças, possibilitando que os indivíduos estejam sempre atualizados e garantindo que não gastem mais do que podem adquirir.

6. O protótipo de aplicação híbrida

Com base no conhecimento adquirido durante o levantamento bibliográfico o referente trabalho tem como propósito desenvolver um protótipo de aplicativo híbrido utilizando o framework Ionic. Para isso foi necessário a preparação de um ambiente adequado para o desenvolvimento e alguns recursos específicos.

O aplicativo é direcionado para a área de gestão financeira e poderá ser utilizado em dispositivos móveis que possuem as plataformas Android e iOS. Além disto, foram utilizados *plugins* para o banco de dados e para a exibição de uma janela de confirmação para a exclusão dos itens, respectivamente, *SQLite* e *Dialogs*.

Para o desenvolvimento da aplicação foi necessário a realização de um levantamento acerca da estrutura dos softwares móveis para o controle financeiro e quais as principais funcionalidades.

Modelar os dados da aplicação facilita no desenvolvimento, uma vez que através dela é possível analisar e obter uma maior percepção de como funcionara a ligação entre os cadastros, diminuindo os erros de programação. Sendo assim, torna-se um requisito que deve ser considerado importante. Para a criação da modelagem, foi utilizada a ferramenta MySQL Workbench,

Para criar a aplicação foram utilizados os comandos de criação de projetos do CLI do Ionic, que são representados por um linha de comando, possibilitando ainda a escolha de três tipos diferentes de layout iniciais. Optou-se então a escolha do layout em branco, para que o mesmo fosse modelado de acordo com a necessidade.

Sendo assim o projeto contém uma estruturação de pastas onde estão armazenadas várias informação e também em uma pasta separada com o código fonte da aplicação. Os demais repositórios são utilizados apenas para fins de compilação e interpretação para o *build* do *app* bem como os *plugins* utilizados.

Na pasta *js* estão todos os arquivos *javascript* que fazem parte do funcionamento e configurações do sistema. O arquivo principal da aplicação é o *app.js*, pois nele, além de atribuir o nome da aplicação, injetar os recursos necessários para o funcionamento do Ionic e os demais serviços criados, é iniciado a variável global *app* e atribuído a ela a instância da aplicação, de modo que todos os *controllers* e *routes*, serão criados a partir dela. Embora o Angular permita modularizar em arquivos distribuídos, foi optado por iniciar e criar o banco de dados neste

Seguindo as boas práticas de programação e buscando sempre minimizar ao máximo a repetição de códigos, para criar as telas foi optado por usar um *template abstract* que contém os menus. Dentro deste arquivo, o *menu.tpl.html*, foi adicionado, além dos códigos responsáveis pela geração do mesmo, uma diretiva do framework chamada de *ion-nav-view*. Ela é responsável por embutir parte de código html sem a necessidade de reescrever todo o código do menu em cada uma das telas do aplicativo.

Depois de ter desenvolvido a parte visual, iniciou-se a codificação dos *controllers*. Para manter o padrão, estes foram adicionados dentro do diretório *controller*. Como o próprio nome sugere, eles são os controladores que fazem a interface entre a *view* e o *model*. Neles, estão contidas todas as funções que comandam as rotinas referentes ao processo pelo qual são responsáveis. Como a estrutura é baseada no Angular, a criação dele usa o comando *controller*, onde deve ser informado o nome. Na função executada por eles, sempre deve ser

informado a variável global *$scope*. Nessa variável estão contidos todos os objetos criados dentro do aplicativo. Ele é responsável pelo *databinding* entre *view* e *controller*, pelo uso das funções dentro da *view*, enfim, é o cerne da aplicação. Também devem ser informados os *plug-ins* que serão utilizados e, caso exista, os módulos criados.

Chegando ao final do desenvolvido, foi criado o arquivo *route.js*, que defini as rotas dos *links* executados. Para isso, é usado o comando *config* do Angular e usado os *plugins* *$stateProvider* e *$urlRouterProvider*. O primeiro é responsável por ligar a *view* com seu respectivo *controller* e lança-los de acordo com a *url* determinada. Já o *$urlRouterProvider*, é usado para os casos onde a *url* que chega não é reconhecida, sendo como uma rota *default*.

Após isso, foi necessário realizar a adição da plataforma, bem como o *build* do aplicativo, através do CLI do ionic, para que fosse possível verificar o funcionamento e realizar os testes.

6.1 FUNCIONALIDADES DA APLICAÇÃO

O protótipo de aplicação é composta por um conjunto de telas com finalidades especificas. Existem telas para a visualização de informações, listagem de dados cadastrados, e as telas de cadastramento. Além disto, utiliza o conceito de menus, então todas as telas são acessadas através dele.

Seguindo uma ordem, inicialmente, tem-se a página inicial, que é compostas por dois quadros com informações, um com o saldo das contas e o outro com o total de despesas e receitas. A aplicação conta com uma listagem de contas, categorias, lançamentos e gráficos utilizando o conceito de lista.

A principal tela da aplicação que é responsável pela movimentação das informações é a tela de lançamentos, que simplifica o cadastramento de receitas e despesas. Seu cadastro possui os campos valor, descrição, data, tipo de lançamento, categoria e conta.

Além disto existe outra funcionalidade que é a exibição dos gráficos que permite a verificação das informações cadastradas no aplicativo.

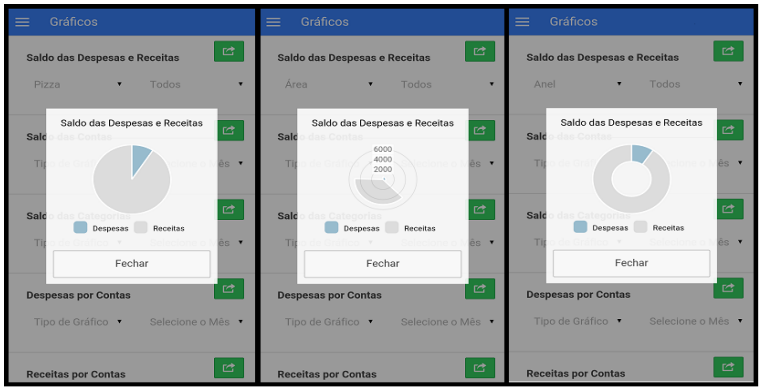


Figura 2. Ilustração dos gráficos da aplicação.

6.2 TESTES DA APLICAÇÃO

Após concluída a fase de desenvolvimento do protótipo, foi realizada a etapa de testes. Para tal procedimento, inicialmente, foi averiguado se os menus estavam dando acesso aos cadastros correspondentes a seus nomes. Logo após foram cadastradas as contas, categorias e lançamentos para verificar se o funcionamento cadastral estava correto. Em seguida com as informações cadastradas foram executados testes com base nas listas com o intuito de analisar

o processo de atualização e exclusão de itens. Imediatamente, realizou-se a emissão dos gráficos a fim de que o usuário possa ter uma breve amostra acerca de suas informações.

No contexto atual, pretendeu-se fazer a execução dos testes em outra plataforma a fim de validar o conceito da aplicação híbrida, e verificar se as funcionalidades teriam o mesmo resultado apresentado na outra plataforma. Para este procedimento, foi utilizada a ferramenta *VMWare Player* 7 versão para uso individual, para criar uma máquina virtual com o sistema operacional *Mac OS X Yosemite* 10.10, uma vez que para a realização do *build* para o iOS é necessário o SDK da Apple que funciona exclusivamente em seu sistema operacional.

Através disto o projeto foi copiado para a máquina virtual e utilizou-se os mesmo procedimentos que os da plataforma Android. Adicionando a plataforma do iOS, realizando o build e executando através do emulador.

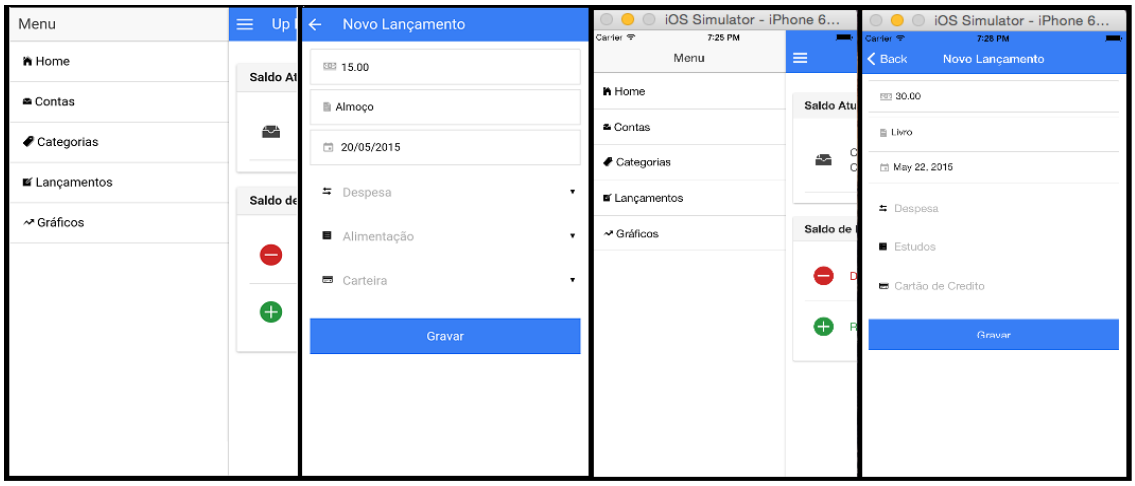


Figura 3. Comparativo entre as telas em ambas plataformas

7. Conclusão

Com a constante expansão tecnológica, é possível perceber o crescimento dos dispositivos móveis, aplicações e meios para o desenvolvimento de novas funcionalidades. Os celulares, tablets e smartphones estão se tornando cada vez mais uma peça chave na vida das pessoas. Oferece uma vasta quantidade de vantagens e benefícios de fácil acesso aos usuários. Isso faz com que os desenvolvedores cada vez mais, busquem por aperfeiçoar seu conhecimento perante a este avanço.

Perante a esta situação, foram surgindo novas ferramentas que auxiliam o desenvolvimento de aplicações, e que reduzissem o custo de desenvolvimento e o número de manutenções, bem como a necessidade de reescrever o código fonte, uma vez que, se tinha necessidade que a aplicação fosse replicada para outra plataforma.

Diante disto, optou-se por realizar o desenvolvimento de uma aplicação mobile híbrida para o controle financeiro utilizando um *framework* multiplataforma especifico. Conclui-se que com a vasta quantidade de frameworks existentes atualmente, é realmente difícil escolher um *framework* para o desenvolvimento de aplicações *mobile* híbridas. Sendo assim com pesquisas já realizadas e com base nas funcionalidades dos *frameworks* optou-se por utilizar o Ionic.

Foi possível adquirir com este artigo um maior conhecimento sobre *frameworks* multiplataformas e aplicações híbridas e principalmente acerca do Ionic. Foi constatado ainda,

que o *framework* Ionic tem uma vasta quantidade de recursos que possibilita o desenvolvimento de uma aplicação dinâmica, e de forma mais ágil que as demais, sendo que a linha de aprendizado adquirida, pode ser utilizada também, para a criação de aplicações web, uma vez que esses *frameworks* utilizam essa tecnologia.

References

Allen, Sarah; Graupera, Vidal; Lundrigan, Lee. Desenvolvimento Profissional Multiplataforma para Smartphone, Iphone, Android, Windows Mobile e Blackberry. Rio de Janeiro: Alta Books, 2012, 280p.

Cavazza, Fred. Mobile Web App vs. Native App? It's Complicated. Forbes, set. 2011, Disponível em: < http://www.forbes.com/sites/pega/2015/06/03/disruptors-wanted-how-successful-companies-view-employees-and-customers/. Acesso em: 20 nov. 2014

Cerbasi, Gustavo. Casais inteligentes enriquecem juntos: finanças para casais. São Paulo: Gente, 2004. 192 p.

Devmedia. Aplicações Móveis: Nativas ou Web?. Disponível em: <http://www.devmedia.com.br/aplicacoes-moveis-nativas-ou-web/30392> Acesso em: 5 maio 2015.

Docs Angular. API Guide. Disponível em:<https://docs.angularjs.org/guide/>. Acesso em: 11 abr. 2015.

Gok, Nizametin; Khanna, Nitin. Building Hybrid Apps with Java and JavaScript. Sebastopol,CA: O'Reilly, 2013. 156p.

Ionic. Ionic Documentation. Disponivél em: <http://ionicframework.com/docs/> Acesso em: 5 maio 2015

Phan, Hoc. Full Stack Mobile App with Ionic Framework. Kindle Edition, 2014. 96p.

Raj, Jay. The Top 7 Hybrid Mobile App Frameworks. Sidepoint, nov. 2014, Disponível em: < http://www.sitepoint.com/top-7-hybrid-mobile-app-frameworks/>

Rocha, Ricardo Humberto; Sousa, Almir Ferreira de; Torralvo, Caio Gragata. Planejamento Financeiro Pessoal e Gestão do Patrimônio. São Paulo: Atlas, 2012. 168 p.

Teleco. Mercado mundial de smartphones. Disponível em:<http://www.teleco.com.br/sist_operacional.asp>. Acesso em: 6 nov. 2014.

Wilken, Jeremy. Ionic in Action. Estados Unidos: Manning Early Access Program, 2015. 325p.