IV SIMTEC
Simpósio de Tecnologia da Fatec Taquaritinga

Tecnologias Web para o Desenvolvimento Mobile Nativo
Web Technologies for Native Mobile Development

Vitor da Silva Cruz – vitor.dasilvacruz@gmail.coom

Erick Eduardo Petrucelli – erick.petrucelli@fatectq.edu.br

Faculdade de Tecnologia – Taquaritinga – São Paulo – Brasil

RESUMO

O mercado de aplicativos para dispositivos móveis enfrenta um dilema nos dias atuais: consideráveis divergências entre os principais sistemas operacionais móveis, Android e iOS, incluindo consideráveis divergências nas linguagens de programação e nos ambientes de desenvolvimento de cada plataforma. Uma abordagem crescente para amenizar isto é o desenvolvimento de aplicações Web que compilam como aplicativos nativos para ambas as plataformas. O presente trabalho teve por objetivo apresentar um estudo teórico sobre três tecnologias neste âmbito. Para tal, foram analisados artigos acadêmicos e livros sobre as tecnologias em questão, bem como as documentações oficiais disponibilizadas pelas equipes responsáveis por estas. O estudo de cada ferramenta foi divido em uma introdução sobre a ferramenta, as características gerais de uso durante o desenvolvimento, o funcionamento interno da comunicação com o sistema operacional móvel, as características de conectividade à Internet, os conhecimentos prévios necessários para adoção da ferramenta e uma reflexão sobre possíveis desvantagens. Ao final do estudo comparativo, observou-se que os desenvolvedores realmente podem usufruir das soluções para compilação de aplicativos móveis nativos a partir do desenvolvimento em plataforma Web, porém, a maturidade destas ferramentas ainda pode ser questionada.

Palavras-chave: Aplicativos Nativos, Desenvolvimento Móvel, Programação Web.

ABSTRACT

The market of mobile apps faces a dilemma nowadays: considerable differences between the major mobile operating systems, Android and iOS, including the considerable divergences about programming languages and development environments of each platform. A growing approach to mitigating it is the development of Web applications that can compile as native apps for both platforms. The objective of this paper was to present a theoretical study about three technologies in this field. For that, academic articles and books of the technologies were analyzed, as well as the official documentation provided by the teams responsible for them. The study of each tool was divided in an introduction about the tool, the general characteristics of use during the development, the internals details of the communication with the mobile operating system, the characteristics of Internet connectivity, the previous knowledge needed to adopt the tool and a reflection on possible disadvantages. At the end of the comparative study, it was observed that developers already can enjoy the solutions for compiling native mobile apps from Web platform development, however, the maturity of these tools can still be questioned.

Keywords: Native Apps, Mobile Development, Web Programming.

1. INTRODUÇÃO

"O uso dos *apps mobile* cresceu 58% em 2015 em relação ao ano anterior, sendo que 40% deles foram feitos por usuários antigos" (MOBILE TIME, 2016). Observando tal informação, é viável considerar que diversas empresas têm buscado se atualizar e oferecer aos seus clientes a comodidade de utilizar alguns – ou até mesmo todos – os seus serviços através do conforto de suas casas, ou a partir de qualquer outro local, através de um *smartphone* ou outro dispositivo móvel com conexão à Internet.

Por conta disto, pode-se entender que nos dias atuais ocorre uma "corrida" pelo desenvolvimento de aplicativos para sistemas operacionais móveis, sendo tal mercado dominado por duas plataformas: o Android, com cerca de 74% dos usuários, e o iOS, com cerca de 19% dos usuários (GLOBALSTATS, 2017). Para atender tal demanda, as empresas enfrentam o desafio de encontrar equipes de desenvolvimento flexíveis, com desenvolvedores habilitados ao menos em duas linguagens de programação e com conhecimentos aprofundados em diversas características específicas de cada plataforma. Estas divergências, além de dificultarem o desenvolvimento de aplicações compatíveis em ambas as plataformas, aumenta a probabilidade de erros. Também é de se destacar a provável elevação de custos para o desenvolvimento de duas soluções distintas, bem como o custo para manutenção e solução de *bugs* em bases de código diferentes, mas com um mesmo objetivo final.

Com a utilização crescente da plataforma Web, bem como suas constantes inovações e amadurecimento, é natural que exista uma gama capacitada de desenvolvedores Web disponíveis no mercado. Desta maneira, a plataforma Web surge como uma alternativa concreta para driblar o problema exposto.

De acordo com Jones (2011, p. 5 e p. 6):

> Uma alternativa ao desenvolvimento de aplicativos nativos é criar, em vez disso, um aplicativo da Web. Isso envolve o uso de HTML, JavaScript e CSS mais familiar e fácil de aprender [...]. A distribuição para aplicativos da Web é através de um servidor Web, e não de uma loja de aplicativos [...]. A velocidade de execução provavelmente será mais lenta [...]. Também haverá acesso mais limitado ao hardware do dispositivo.

Sendo assim, soluções puramente Web – como a criação de *websites* que podem agir como aplicativos, os Progressive Web Apps (PWA) – podem enfrentar problemas de desempenho e sofrer de limitações ao acesso às funções dos dispositivos. Ressalta-se que o acesso a recursos nativos através de Application Programming Interfaces (API) do HTML5 e

do ECMAScript 2015+ (ES2015+) tem evoluído, mas ainda não se oferece o mesmo nível de acesso completo a todos os recursos de *hardware*.

Segundo Palmieri, Singh e Chicchetti (2012, p. 3):

> O PhoneGap é uma solução útil para a construção de aplicativos móveis usando linguagens de programação modernas da Web, como HTML, HTML5, CSS, CSS3 e JavaScript [...]. As aplicações PhoneGap são híbridas, o que significa que elas não são puramente nativas ou baseadas na Web. O significado de "não puramente nativo" vem da renderização de layout que é feita via Web View, em vez da linguagem nativa do sistema operacional [...].

Desta forma, aplicações híbridas também têm se apresentado como uma opção relevante e em crescimento no mercado de desenvolvimento, principalmente através de soluções como PhoneGap e Ionic, dentre outras similares. De fato, são apenas *websites* tradicionais que executam através do motor de navegador – um Web View do sistema operacional móvel – a partir de uma "casca" de aplicativo nativo. Embora com maior acesso aos recursos nativos e podendo ser colocado nas lojas de aplicativos de cada plataforma, não deixam de ser *sites* e, portanto, têm seu desempenho amarrado ao desempenho do motor de execução Web e, normalmente, apresentam desempenho inferior a aplicativos nativos reais.

Buscando contornar tais limitações, surgiram ferramentas que almejam permitir aos desenvolvedores Web utilizar seus conhecimentos prévios, mas ainda assim produzir aplicações realmente nativas.

De acordo com Ruben e Kris (2016, p. 197):

> [...] notamos que as soluções relativamente novas, React Native do Facebook e NativeScript da Telerik, estão ganhando muita atenção. Parte do seu sucesso decorre dos novos conceitos que elas introduziram [...]. Os novos conceitos fornecidos por essas soluções garantem uma competição saudável que impulsiona a melhoria e a inovação.

1.1. Metodologia

O presente estudo se baseou em revisão bibliográfica, tendo sido analisados artigos acadêmicos e livros sobre as tecnologias em questão. Visto que boa parte dos assuntos abordados são recentes, incluindo as próprias ferramentas que foram o foco do estudo, há carência de materiais já publicados, portanto as documentações oficiais disponibilizadas pelas equipes responsáveis representam parcela considerável do material consultado.

A escolha das ferramentas abordadas neste trabalho foi possível após um período de levantamento exploratório na Internet, em fóruns e comunidades de desenvolvimento *mobile*, quando foram identificadas as ferramentas que estavam sendo discutidos pelos profissionais.

1.2. Estrutura

Na introdução, apresentou-se a problematização da demanda por produção de apps móveis e o cenário da plataforma Web como plataforma de desenvolvimento móvel, bem como alguns problemas inerentes às soluções PWA e híbridas. Em seguida, foi apresentada a metodologia de condução do estudo e o presente tópico.

No Capítulo 2, apresenta-se a ferramenta React Native, com uma introdução, suas características gerais, como se comunica com o sistema operacional móvel, as características de conexão à Internet, os conhecimentos prévios necessários para adoção da ferramenta e uma reflexão sobre possíveis vantagens e desvantagens.

No Capítulo 3, apresenta-se a ferramenta NativeScript, seguindo a mesma estrutura utilizada no capítulo anterior.

No Capítulo 4, apresenta-se a ferramenta Weex, seguindo novamente a mesma estrutura utilizada no capítulo anterior.

O trabalho encerra-se com as considerações finais dos autores em relação ao conteúdo apresentado e, por fim, com as referências bibliográficas utilizadas.

2. REACT NATIVE

Smeets e Aerts (2016, p. 197) mostram a possibilidade de desenvolver de forma nativa usando React Native, extensão do projeto React, originalmente grafado como ReactJS.

O React Native é uma ferramenta desenvolvida pela empresa Facebook, a qual permite a criação de aplicações *mobile* nativas para Android e iOS, em uma abordagem multiplataforma baseada na linguagem JavaScript (WINTERFELDT, 2017).

Como citado, utiliza em seu núcleo o React, uma biblioteca JavaScript e se baseia na criação de interfaces através de JSX, uma extensão para o JavaScript que possibilita escrever sintaxe XML em meio aos códigos da linguagem, sem qualquer semântica ou delimitação de *tags*, possibilitando escrever a interface junto à lógica de programação.

2.1. Resumo sobre React

“React é um framework JavaScript. Foi originalmente criado por engenheiros no Facebook para resolver os desafios relacionados ao desenvolvimento de interfaces de usuário complexas” (GACKENHEIMER, 2015, p. 1).

React funciona com a ideia de criação de componentes modulares reativos através de *data binding* e pode ser usado para criação de aplicações de página-única (do inglês Single Page Applications, comumente grafado com a sigla SPA).

Segundo Salvaneschi, Margara e Tamburrelli (2015, p. 1), "a programação reativa tem sido cada vez mais investigada na comunidade de linguagens de programação e agora está ganhando interesse dos profissionais".

Conforme já citado, o React exige a utilização de JSX para desenvolvimento das interfaces. O JSX é uma extensão versátil que utiliza XML, tanto para interface nativa em dispositivos móveis, ou o Document Object Model (DOM) para a Web, em conjunto com o JavaScript para a lógica de programação e podendo incluir uma adaptação de CSS através do JavaScript, tudo em um único arquivo (HOLMES e BRAY, 2015).

2.2. Funcionamento interno do React Native

De acordo com Eisenman (2016, p. 17), "[...] React Native invoca as APIs de renderização nativas em Objective-C (para iOS) ou Java (para Android)".

React trabalha virtualizando a reatividade dos componentes do aplicativo, ou seja, guarda na memória a estrutura da interface sem alterações e, quando necessário, recupera a parte que será alterada para fazer a renderização mesclando com a alteração. Segundo Facebook (2017), as aplicações React Native são como aplicações React em todo o seu funcionamento, exceto pelo fato de que se invoca as APIs da linguagem nativa do dispositivo para renderizar os componentes, de acordo com o dispositivo em que está instalado. Isto se torna possível pela conexão que o React Native estabelece com a API dos elementos nativos.

Conforme detalha Eisenman (2016, p. 18):

> [...] o React Native realmente traduz sua marcação para elementos de UI reais e nativos, alavancando os meios existentes de renderização de visualizações em qualquer plataforma com a qual você esteja trabalhando. Além disso, React funciona separadamente do segmento principal da UI, para que seu aplicativo possa manter alto desempenho sem sacrificar a capacidade. O ciclo de atualização no React Native é o mesmo que no React: quando há adereços ou mudança de estado, React Native "re-renderiza" as visualizações. A principal diferença entre React Native e React no navegador é que React Native faz isso alavancando as bibliotecas UI de sua plataforma *host*, em vez de usar HTML e CSS [...]

2.3. Conexão das aplicações com a Internet

Frequentemente aplicações móveis se utilizam – ou mesmo dependem – de conexão com a Internet, seja para receber, seja para enviar recursos, de e para um servidor ou URL

remotos. Nesta vertente, é oferecido por padrão a possibilidade de utilização do *fetch*, uma API ainda experimental nos navegadores Web (MDN, 2017), mas implementada conforme a especificação no React Native. Ainda assim, é permitido permitindo importar outras bibliotecas para a tarefa, como *frisbee* e *axios*, se desejado.

React Native também oferece suporte a WebSockets, um protocolo de comunicação *real time* que fornece canais de comunicação *full-duplex* em uma única conexão. Por fim, também há a possibilidade de se trabalhar com aplicações *off-line first*, onde o aplicativo pode oferecer certas funções mesmo estando desconectado e, caso volte a ser conectado em algum momento, possa fazer as ações de sincronia que forem necessárias (FACEBOOK, 2017).

2.4. Conhecimentos prévios necessários

Por ser uma ferramenta de desenvolvimento baseada na plataforma Web, não é surpresa que um bom entendimento de JavaScript é primordial, bem como um entendimento ao menos razoável de HTML e CSS.

Porém, para desenvolver plenamente com React Native, um conjunto maior de conhecimentos é necessário. O conhecimento da própria biblioteca React é fundamental, bem como as especificidades de utilizar JSX, principalmente por suas divergências em relação à adaptação do CSS e o entendimento dos componentes visuais disponíveis no React Native, diferentes do que se pode estar acostumado ao montar interfaces com HTML. Também se recomenda um conhecimento básico no ecossistema de ferramentas de desenvolvimento JavaScript, principalmente Node e NPM (MASIELLO e FRIEDMANN, 2017).

2.5. Vantagens

Como principal vantagem está o fato de se utilizar a plataforma Web para criação de aplicativos móveis nativos, sua premissa básica.

Além disso, permite testar facilmente a aplicação durante o desenvolvimento, através de um *app* chamado Expo, capaz de atualizar em tempo real a aplicação no dispositivo móvel a cada modificação no código. Também há um editor *online* para testes de desenvolvimento, permitindo visualizar a interface proposta no próprio navegador (FACEBOOK, 2017).

Outra característica comumente elencada como vantagem é o fato de ter uma grande empresa responsável pelo projeto, no caso o Facebook. Isto não apenas facilita integrações

com outras soluções da empresa, como *login* integrado com sua rede social, como também garante grande visibilidade ao projeto e, portanto, muitas pessoas e empresas demandando profissionais com estes conhecimentos.

2.6. Desvantagens

Segundo Velasco (2016, p. 28):

> O maior risco que vem com o uso do React Native é o seu baixo nível de maturidade. Como um projeto jovem, React Native ainda contém alguns problemas e falta de otimizações. No entanto, esse risco é algo compensado pelo fato de o Facebook usar em produção para seus próprios aplicativos. A comunidade também está contribuindo para o projeto, e as atualizações são lançadas a cada duas semanas [...]. Outra desvantagem importante é o que acontece quando você encontra uma limitação nas APIs nativas suportadas. Se você estiver confortável em JavaScript, a curva de aprendizado de linguagens de programação como Java ou Objective-C, é muito íngreme e pode potencialmente atrasar um projeto.

Mesmo que não necessite escrever códigos completamente distintos ao necessitar acessar uma API nativa que o React Native não ofereça suporte, o desenvolvedor pode precisar implementar códigos nativos.

Um ponto controverso que eventualmente pode ser apontado como desvantagem está na exigência de utilização do JSX, o qual pode se apresentar como uma questão muito disruptiva para parte dos desenvolvedores acostumados à plataforma Web e à sua tradicional separação de responsabilidades entre HTML, CSS e JavaScript.

Paradoxalmente, outra possível desvantagem advém justamente do fato do Facebook estar por traz do projeto, o que, apesar dos pontos positivos, pode significar que os rumos sejam definidos apenas por interesses puramente comerciais de uma única organização, em detrimento das reais necessidades da comunidade de desenvolvedores.

3. NATIVESCRIPT

Segundo Šmitalová (2017, p. 11):

> O NativeScript é um *framework* de código aberto disponibilizado sob a licença Apache 2. Foi criado e é suportado pela Telerik. Esta ferramenta de desenvolvimento permite a criação de aplicativos iOS e Android nativos, o Windows Mobile está planejado para ser adicionado no futuro. O código base pode ser escrito em Angular 2, TypeScript ou JavaScript moderno. NativeScript torna o código específico para cada plataforma com o mecanismo de renderização nativo. O resultado é uma interface de usuário com desempenho e experiência do usuário nativos.

É uma solução que também permite desenvolvimento de aplicações de forma nativa, baseando-se em XML para a estruturação da interface e CSS para sua estilização. A estruturação do projeto ocorre conforme os padrões do *framework* Angular. É relevante destacar que, no momento de elaboração deste trabalho, segundo Telerik (2017), uma versão de NativeScript com suporte ao *framework* Vue está em desenvolvimento, mas ainda não publicada oficialmente. O restante deste capítulo manterá seu foco na versão baseada no Angular, por ser a solução oficialmente suportada pelo NativeScript no momento. Este trabalho ainda apresentará um pouco mais sobre Vue ao abordar a ferramenta Weex.

3.1. Resumo sobre Angular e TypeScript

Angular é um *framework client-side* que possibilita a criação de SPA, tendo sido desenvolvido e disponibilizado pela empresa Google. É responsável por interfaces dinâmicas que possibilitam manipulação no virtual do DOM (ALMEIDA, 2015).

TypeScript é uma linguagem de *script* fortemente tipada, criada pela Microsoft. Exige a definição de tipos de dados, é puramente orientada a objetos com herança baseada em classes e, ao final do desenvolvimento, é convertida para JavaScript tradicional em um processo denominado “transpilação” (BIERMAN, ABADI e TORGERSEN, 2014).

Com o uso do Angular no NativeScript, possibilita criar componentes reativos, qualquer alteração o mesmo é renderizado com a nova alteração tornando o contanto com o cliente mais dinâmico, além da capacidade de reutilização quando necessário.

3.2. Funcionamento interno do NativeScript

NativeScript oferece total acesso às APIs nativas através de JavaScript puro, ou através de TypeScript, seja com o Angular ou com o Vue (TELERIK, 2017).

Este acesso direto é proporcionado através de uma ponte JavaScript, a qual permite aos desenvolvedores criar suas próprias extensões, sem utilização de linguagem nativa (ŠMITALOVÁ 2017). Ainda segundo este autor, isto só é possível graças à implementação de motores de execução JavaScript existente dentro das próprias plataformas móveis, o motor V8 no Android (GOOGLE, 2017) e o motor JavaScript Core no iOS (APPLE, 2017).

E este acesso possibilita o desenvolvimento de uma interface diferente, ou ter acesso a câmera, e entre outros que o aparelho oferece, utilizando no código somente XML e JavaScript, e o V8 ou JavaScript Core é quem torna tal acesso possível de maneira nativa.

3.3. Conexão das aplicações com a Internet

Assim como apresentado para o React Native, segundo Telerik (2017), o desenvolvedor tem à sua disposição para realizar requisições remotas a API *fetch*, mas também pode se utilizar de qualquer outro módulo desenvolvido em Node, inclusive para outros tipos de conexão, como Stream ou WebSockets, contanto que tais módulos não precisem estar sendo executados por navegadores Web para funcionar, ou seja, é necessário que as ferramentas possam realizar requisições a partir do JavaScript do Node.

3.4. Conhecimentos prévios necessários

O NativeScript é mais amplo nesta questão, pois permite usar XML, CSS e JavaScript, mas também permite usar Angular e TypeScript. E, assim que for oficialmente lançado, também permitirá usar o Vue. Assim como outras ferramentas, necessita de conhecimento básico em Node e NPM (TELERIK, 2017). É interessante observar que suportar diversas linguagens e *frameworks* não é algo que deve aumentar a curva de aprendizado, pelo contrário, deve permitir que desenvolvedores habituados a qualquer uma das opções possa imergir no desenvolvimento nativo com mais velocidade.

Telerik (2017) aponta que, para criar um pacote instalável da aplicação, ainda assim é necessário ter os SDKs (*software development kits*) dos sistemas operacionais alvo, mas não há necessidade de conhecimentos prévios com os SDKs, pois o NativeScript oferece guias para iniciantes e para usuários avançados, além de oferecer instalador para Windows que configura automaticamente as variáveis de ambiente do Android Studio e do Java para um ambiente de desenvolvimento Android sem complicações.

3.5. Vantagens

Assim como citado para o React Native, a principal vantagem do NativeScript também é habilitar o desenvolvedor Web a criar aplicativos móveis nativos multiplataforma.

De acordo com Winterfeldt (2017, p. 33):

> A estrutura de código aberto do NativeScript [...] permite o desenvolvimento de aplicativos móveis nativos para iOS e Android com uma única base de código. Para isso, XML, JavaScript ou TypeScript, CSS e Angular são usados. O NativeScript oferece a vantagem em relação a outros *frameworks* gratuitos de que as classes e métodos nativos de sistemas operacionais móveis podem ser usados no código de aplicativo baseado em JavaScript. Portanto, o NativeScript pode ser estendido por suas próprias funcionalidades, que são baseadas em APIs nativas, sem necessidade de conhecer a sintaxe das linguagens de programação específicas da plataforma.

Conforme observado, uma grande vantagem do NativeScript está em sua flexibilidade, oferecendo além de acesso puro por JavaScript às APIs e à interface, o desenvolvimento de extensões próprias e uso do Angular (e em breve do Vue) no lugar do JavaScript puro.

É de se mencionar que também possibilita ao desenvolvedor acompanhar a aplicação enquanto a cria, através de emulador ou mesmo através de cabo USB (TELERIK, 2017).

3.6. Desvantagens

O *framework* Angular tem passado por evoluções confusas nos últimos anos e, ainda que com a força do Google em sua retaguarda, pode-se considerar uma desvantagem do NativeScript se prender a ele. Como o desenvolvimento de forma produtiva com Vue no NativeScript ainda não está oficialmente completo, tal desvantagem ainda persiste neste momento, mas pode se tornar irrelevante em breve.

Outra possível desvantagem, advém paradoxalmente do fato de o acesso às APIs nativas ser feito através dos motores JavaScript das plataformas. Embora uma grande vantagem para simplificar o processo de desenvolvimento de operações nativas quando necessário, pode significar um impacto negativo em desempenho, uma vez que os motores JavaScript nem sempre entregam o mesmo desempenho de chamadas realmente nativas.

Embora com uma empresa por trás, a Telerik não chega perto do tamanho e da força do nome Facebook, portanto o NativeScript não chega a ser tão conhecido e não possui uma comunidade tão grande quanto o React Native, o que também pode ser considerado uma desvantagem por alguns desenvolvedores.

4. WEEX

Alibaba (2017) aponta que o Weex é um *framework* para a construção de aplicações multiplataforma, desenvolvido pela empresa chinesa Alibaba, o maior *e-commerce* varejista

do mundo e número de negócios. Atualmente mantido pela Apache Incubadora, tem também forte apoio da comunidade Vue, por tê-lo adotado como núcleo para construção de interfaces.

4.1. Resumo sobre Vue

Segundo Schmitz e Georgii (2017, p. 2):

> [...] o Vue (pronuncia-se "*view*"), um *framework* baseado em componentes reativos, usado especialmente para criar interfaces Web, na maioria das vezes Single Page Applications [...] com somente um arquivo HTML. Vue.js foi concebido para ser simples, reativo, baseado em componentes, compacto e expansível.

Este *framework open-source* com licença MIT foi inicialmente criado pelo desenvolvedor Evan You, após sair da Google onde trabalhava com o Angular. Após uma boa e crescente aceitação pelos desenvolvedores, transformou-se em um projeto da comunidade para a comunidade, contando atualmente com desenvolvedores de várias partes do mundo e mantendo-se totalmente independente dos interesses comerciais de uma única empresa.

Uma característica marcante de seu funcionamento, o *two-way data binding*, muda positivamente a maneira como as alterações de estado das aplicações são tratadas, dinamicamente se atualizando a interface controlada por DOM virtual conforme os dados são atualizados. Os dados são sincronizados a cada evento de entrada por padrão, sem que nada precise ser feito por parte do desenvolvedor (KYRIAKIDIS, MANIATIS e YOU, 2017).

4.2. Funcionamento interno do Weex

O núcleo das aplicações Weex para as APIs do sistema operacional são as linguagens nativas destes, ao mesmo tempo em que a ferramenta fornece muitos módulos e ferramentas nativas para que os desenvolvedores possam usá-las, até mesmo extensões para a interface nativa. Este instrumento não só possibilita, a partir do mesmo código, criar aplicações para Android e iOS, mas até mesmo para a Web. Caso o usuário não queira instalar o aplicativo, o desenvolvedor pode oferecer a possibilidade de acessar uma página na Internet, desenvolvida com o mesmo código-fonte – evidentemente, não oferecerá o mesmo desempenho mas garantirá a mesma experiência de usuário. A aplicação é processada pelo núcleo Vue através do motor JavaScript do aparelho do cliente, o qual gerencia a renderização nativa e invoca as APIs nativas para interação com o usuário (ALIBABA, 2017).

4.3. Conexão das aplicações com a Internet

Alibaba (2017) demonstra que o Weex provê módulos próprios para conexões com a Internet do tipo WebSockets, Stream e até mesmo acrescentar uma Web View dentro do aplicativo. Segundo eles, os módulos próprios podem oferecer o melhor desempenho possível, ao invés de depender de módulos JavaScript isonômicos como as outras soluções.

Segundo Fette e Melnikov (2011, p. 4), “o protocolo WebSockets permite a comunicação bidirecional entre um cliente que executa código não confiável em um ambiente controlado para um *host* remoto que tenha ativado as comunicações desse código”, portanto este cenário pode atender todas as necessidades de comunicação de uma aplicação moderna, sem necessidade de existência de módulos específicos para requisições remotas, como *fetch*.

4.4. Conhecimentos prévios necessários

Assim como citado anteriormente, o desenvolvimento com Weex necessita de conhecimentos sólidos na plataforma Web, ou seja, HTML, CSS e JavaScript. Por se posicionar sobre o núcleo do Vue, o conhecimento deste também é primordial. Como o Vue é relativamente simples e os conceitos básicos da plataforma Web não são desprezados por ele, a curva de aprendizado pode ser muito rápida. As *tags* usadas para a criação das interfaces, assim como as especificações CSS suportadas e as formas de acesso às APIs nativas também são conhecimentos importantes e estão na documentação (ALIBABA, 2017).

Assim como nas outras ferramentas, necessário ter conhecimentos prévios com Node e NPM. E para empacotar tudo em um instalável, é necessário conhecer basicamente como configurar e usar os SDKs de cada dispositivo.

4.6. Vantagens

Assim como exposto nas ferramentas anteriores, o Weex também se desponta por proporcionar a possibilidade de utilização da plataforma Web como ecossistema para o desenvolvimento de aplicativos móveis nativos.

Segundo Alibaba (2017), existe um mercado de extensões criado por desenvolvedores da ferramenta, vantajoso por abrir mais possibilidades à aplicação, por ter a sintaxe Web, código único e aplicação é multiplataforma entre Web, Android e iOS.

Por fim, ressalta-se que a comunidade Vue tem se mostrado muito aberta à colaboração, visivelmente apaixonada pelo *framework* e seu ecossistema, sendo este outro ponto positivo relevante quanto à adoção do Weex.

4.5. Desvantagens

Por ainda ser um projeto em desenvolvimento, não é aconselhável para aplicações críticas. Além disso, por ser uma empresa chinesa o principal mantenedor, boa parte da comunidade são de chineses. Atualmente a documentação em inglês recebe menos informações e tem sido alvo de muitas reclamações por desenvolvedores ocidentais que tentam utilizar a ferramenta e não conseguem evoluir (ALVES, 2017).

5. CONSIDERAÇÕES FINAIS

Com base no que foi apresentado durante este estudo, denota-se que as ferramentas analisadas efetivamente habilitam o desenvolvedor Web a criar aplicativos móveis nativos, se aproveitando de boa parte de seus conhecimentos prévios e geralmente evitando a necessidade de aprendizado específico de duas plataformas móveis distintas.

Observa-se também a viabilidade de criação de uma única base de código, para a criação de aplicativos para ambas as plataformas móveis. Indo um pouco além, dependendo de como a arquitetura da aplicação for estruturada e como divergências nos padrões visuais das plataformas forem controladas, uma base de código universal se torna possível não apenas para os aplicativos móveis, mas também para a versão Web.

Todas estas vantagens podem representar grande avanço para os negócios, pois as empresas não necessariamente precisariam de diversas equipes de desenvolvimento para criação de seus aplicativos em diferentes linguagens.

Em contrapartida, as tecnologias apresentadas são novas, inclusive estando ainda em franco desenvolvimento. Portanto, há um fator de risco inerente à adoção de qualquer uma delas, inclusive sobre a incerteza de continuidade dos projetos para o longo prazo.

Conclui-se que os avanços tecnológicos nesta área são relevantes e promissores, porém no estado atual se apresentam mais como bom material para estudos, mas ainda deve-se manter a cautela antes de adotar em projetos reais para ambientes em produção.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ALIBABA, **Weex Documentation**. 2017. Disponível em: <http://weex-project.io>. Acesso em: 15 set. 2017.

ALMEIDA, F. **Mean Full Stack JavaScript para aplicações Web**: com MongoDB, Express, Angular e Node. Casa do Código, 2015.

ALVES, T. **Native Apps with Vue.js: Weex or NativeScript?**. 2017. Disponível em: <https://hackernoon.com/native-apps-with-vue-js-weex-or-nativescript-8d8f0bac041d>. Acesso em: 3 out. 2017.

APPLE, **JavaScriptCore**. 2017. Disponível em: <https://developer.apple.com/documentation/JavaScriptCore>. Acesso em: 15 set. 2017.

BIERMAN, G.; ABADI, M.; TORGERSEN, M. **Understanding TypeScript**. 2014. CiterSeerX. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=3F4F1592A7CD386FCA61DAA500392447?doi=10.1.1.650.1531&rep=rep1&type=pdf>. Acesso em: 03 out. 2017.

EISENMAN, B. **Learning React Native**: Building Native Mobile Apps with JavaScript. 1. ed. California: O'Reilly Media, 2016.

FACEBOOK. **React Native Docs**. 2017. Disponível em: <https://facebook.github.io/react-native/docs>. Acesso em: 3 out. 2017.

GACKENHEIMER, C. **Introduction to React**. 1. ed. New York: Apress Media, 2015.

GLOBALSTATS. **StatCounter**: Mobile Operating System Market Share Worldwide. set. 2017. Disponível em: <http://gs.statcounter.com/os-market-share/mobile/worldwide>. Acesso em: 2 out. 2017.

GOOGLE. **V8 JavaScript Engine**. 2017. Disponível em: <https://android.googlesource.com/platform/external/v8>. Acesso em: 15 set. 2017.

HOLMES, E.; BRAY, T. **Getting Started with React Native**. 1. ed. Birmingham: Packt Publishing Ltd, 2015.

JONES, R. G. **Emerging Technologies**: Mobile Apps for Language Learning. Virginia Commonwealth University. Disponível em: <http://scholarspace.manoa.hawaii.edu/bitstream/10125/44244/1/15_02_emerging.pdf>. Acesso em: 3 out. 2017.

KRYIAKIDIS, A.; MANIATIS, K.; YOU, E. **The Majesty of Vue.js 2**. 1. ed. British Columbia, Canada: Leanpub, 2016.

MASIELLO, E.; FRIEDMANN, J. **Mastering React Native**. 1. ed. Birmingham: Packt Publishing Ltd, 2017.

MDN. **Fetch API**. Mozilla Developer Network, set. 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API>. Acesso em: 5 out. 2017.

MOBILE TIME. **Uso dos Aplicativos Móveis Cresceu 58% em 2015**. jan. 2016. Disponível em: <http://www.mobiletime.com.br/05/01/2016/uso-dos-aplicativos-móveis-cresceu-58-em-2015/425202/news.aspx>. Acesso em: 15 set. 2017.

PALMIERI, M.; SINGH, I.; CICCHETTI, A. **Comparison of Cross-Platform Mobile Development Tools**. 2012. Suecia: Semantic Scholar. Disponível em: <https://pdfs.semanticscholar.org/be08/83eab3d3d6eb10c4ff1189163f6453254da1.pdf>. Acesso em: 15 set. 2017.

SALVANESCHI, G.; MARGARA, A.; TAMBURRELLI, G. **Reactive Programming:** A Walkthrough. aug. 2015. Software Engineering (ICSE), 2015. 37th IEEE International Conference. Disponível em: <http://ieeexplore.ieee.org/abstract/document/7203125/?reload=true>. Acesso em: 15 set. 2017.

SCHMITZ, D.; GEORGII, D. P. **Vue.js na Prática**. 2. ed. British Columbia, Canada: Leanpub, 2017.

SMEETS, R.; AERTS, K. **Trends in *Web* Based Cross Platform Technologies**. jun. 2016. International Journal of Computer Science and Mobile Computing. Disponível em: <https://lirias.kuleuven.be/bitstream/123456789/550858/1/V5I6201654.pdf>. Acesso em: 15 set. 2017.

ŠMITALOVÁ, L. **NativeScript Application for Continuous Improvement of Software Engineer's Skills**. 2017. Czech Republic: Faculty of Informatics, Masaryk University. Disponível em: <https://is.muni.cz/th/410198/fi_b/thesis.pdf>. Acesso em: 15 set. 2017.

TELERIK. **NativeScript Documentation**. 2017. Disponível em: <https://docs.nativescript.org>. Acesso em: 15 set. 2017.

VELASCO, I. M. **Desarrollo de una Aplicación Móvil usando las Tecnologías React Native y Phonegap y Diseño e Implementación de un Escenario de Prueba con React Native**. 2016. Universidad Politécnica de Madri. Disponível em: <http://oa.upm.es/42914/1/PFC_IGNACIO_MARTIN_VELASCO_2016.pdf>. Acesso em: 3 out. 2017.

WINTERFELDT, J. **Context-Aware Mobile Crowd Sensing using Mobile Hybrid Application Frameworks**. mai. 2017. Germany: Universität Ulm. Disponível em: <http://dbis.eprints.uni-ulm.de/1486/1/MA_WIN_2017.pdf>. Acesso em: 15 set. 2017.