COMPARAÇÃO DE CUSTO BENEFÍCIO ENTRE OS SDKS IONIC E FLUTTER E A BIBLIOTECA REACT NATIVE PARA APLICAÇÕES MÓVEIS

Thiago Silva Ferreira[1]; Giovanni Jatene Barberato[2]; Célio Cazares Shiavao de Morais[3]; José Roberto de Almeida[4]

[1,2,3,4] Universidade de Uberaba
thiagosilvaferreira@edu.uniube.br; jose.almeida@uniube.br

Resumo

O desenvolvimento de aplicativos móveis intensificou-se nos últimos anos devido ao crescimento acelerado e popularização dos *smartphones.* Cada empresa do ramo móvel possui seu próprio sistema operacional, loja de aplicativos, parcela de mercado e ambientes de desenvolvimento. No entanto, quanto mais sistemas diferentes existem, maior o esforço, custo e tempo para desenvolver um aplicativo para todas as plataformas existentes. Visando sanar dificuldades, o desenvolvimento multiplataforma tem a premissa de criação de apenas um código que possa abranger várias plataformas. Nesse contexto, o presente trabalho busca definir vantagens e desvantagens da abordagem multiplataforma quando comparada com a abordagem nativa. Por meio da análise de um exemplo de uso, que consistiu na recriação de um aplicativo nativo, implementado originalmente para a plataforma iOS, utilizando o *framework Ionic,* foi possível comparar e comprovar empiricamente os dados obtidos na literatura. Mediante análise exploratória, foram selecionadas funcionalidades para verificar a viabilidade do desenvolvimento dessas em ambiente multiplataforma e para realizar uma comparação com o desenvolvimento das mesmas em um ambiente nativo. Os dados obtidos foram confrontados com as opiniões de especialistas para avaliar a percepção dos mesmos sobre o cenário atual do desenvolvimento multiplataforma.

Palavras-chave: Internet. Html. Javascript. Multiplataforma.

1 Introdução

Para Abranches (2018), desde 2007, *apps* se tornaram mania, e grande parte deste sucesso vem dos *smartphones*, que crescem rapidamente em todo o mundo e oferecem aos usuários diversas ferramentas para cada tipo de necessidade ao alcance de nossas mãos, e cada vez mais faz parte do nosso cotidiano. Com eles somos capazes de gerir nossas finanças, pedir comida, pedir um táxi, controlar uma empresa, acessar câmeras de nossas casas, fazer compras, realizar vendas, e muitas outras funcionalidades que facilitam nossas vidas.

O uso de *Smartphones* é uma realidade e ainda permanecerão, (sob o avanço da tecnologia) no futuro e, cada vez mais os usuários de um aplicativo serão responsáveis pela divulgação e popularidade de um serviço ou produto (ABRANCHES, 2018).

Os aplicativos híbridos estão sendo muito utilizados no mercado, tendo em vista que um *app* nativo tem um custo mais elevado, pois tem uma linguagem própria e específica do sistema operacional, desenvolvida para cada plataforma. Para publicar o aplicativo nas lojas é necessária uma aprovação, ou seja, ele precisa estar de acordo com as políticas internas de cada uma delas.

Para Madureira (2017), o aplicativo híbrido é construído na linguagem HTML5, CSS e JavaScript, assim como o site *mobile*. Esse código é alocado dentro de

um *container*, integrando as funcionalidades que o seu dispositivo oferece, o que permite uma experiência melhor ao usuário que os *web Apps*.

2 *Materiais e Métodos*

Foram realizadas pesquisas em artigos e análises sobre linguagens hibridas de programação através da internet. Com esses dados, foram feitos estudos sobre *SDK,* ou Kit de desenvolvimento de software, *Flutter*, frameworks *Ionic* e biblioteca *React*.

2.1 SDK

O *SDK* é um conjunto de ferramentas de desenvolvimento de *software* que permite a criação de aplicativos para um determinado pacote de *software*, *framework*, plataforma de *hardware*, sistema de computador,

2.1.1 *Flutter*

Uma alternativa para desenvolvimento é a nova linguagem do *Google*, o *Flutter,* que é o SDK de código aberto do *Google* que permite o desenvolvimento de aplicativos que executem tanto no *Android* quanto no iOS a partir de uma única base de código. Seu objetivo é permitir que os desenvolvedores criem aplicativos de alta *performance* com uma experiência nativa em ambas as plataformas (CORAZZA, 2018).

Seu fluxo de desenvolvimento é orientado ao *design* e os *widgets* são os blocos básicos da *interface* de usuário de um aplicativo *Flutter*. Assim, existem *widgets* para definir elementos estruturais (botões e menus), elementos de estilo (fontes e cores, entre outros), aspectos de *layouts* (margens e espaçamentos), além de *widgets* com *design* específico para a plataforma *Android* (*Material Components*) e iOS (*Cupertino*).

Segundo Santana (2019), diferente do *React Native* que possui um intermediário (*bridge*) entre a UI (*User Interface*) e o dispositivo, o *Flutter* fica na camada do UI e não chama os componentes nativos do SO (Sistema Operacional), ele é desenhado diretamente em um *canvas* que aumenta a performance e fluidez a nível de um aplicativo desenvolvido exclusivamente nativo. Além do ganho com *performance* o *Flutter* é uma tecnologia recente de fácil aprendizado e já possui algumas *features* sendo desenvolvidas.

No *Flutter*, desenvolvedores encontram ferramentas para criar de forma mais rápida *apps* multiplataforma que, segundo o *Google*, permanecem com uma ótima *performance* — proporcionada pelo uso de um processador acelerado por GPU1 e pelo tempo de execução do código ARM nativo. O design também é algo importante e poderá ser modificado facilmente, sendo adaptados às particularidades de cada sistema. (SAMBO, 2018).

Segundo Silva(2018), para desenvolver no *Flutter*, é necessário ter conhecimento sobre a linguagem *Dart*, pois a proposta que o *framework* traz é justamente ser *ahead of time* (AOT), que significa ter o código compilado antes da execução, onde o carregamento do *app* e das animações do mesmo é mais fluido, devido a essa compilação do código antes da sua execução.

O *Dart* é uma linguagem que permite o desenvolvimento de aplicações em módulos, tornando a percepção do código mais simples e ainda a sua manutenção; Possui ainda uma performance magnífica em termos de desenvolvimento e produção pela capacidade de suportar a compilação *Just-in-Time* e *Ahead-of-Time*. (SAMBO, 2018).

2.2 *Frameworks*

A escolha de um *framework* ou plataforma depende muito de suas habilidades e de seus objetivos com a aplicação. O desenvolvimento multiplataforma tem ganhado muitos adeptos ao longo dos anos. Com isso, o mercado também se expandiu

consideravelmente. Logo, novas ferramentas *cross-platform* foram surgindo no mundo do desenvolvimento mobile. (Micreiros, 2017).

2.2.1 Ionic

É um *framework* baseado em *Cordova* que usa HTML, *JavaScript* e *webview* para o desenvolvimento de aplicações *mobile*. Usam linguagens muito difundidas em desenvolvimento, ajudando na velocidade e redução de custos no desenvolvimento (GIL, 2017).

Segundo Ionic (2018) o site oficial, a ferramenta possui atualmente 4 milhões de aplicativos construídos e 5 milhões de desenvolvedores usando a plataforma, o *Ionic* é a plataforma de escolha para qualquer organização que procura desenvolver aplicativos bonitos que forneçam para o usuário uma experiência rica em qualidade. Com todo esse ambiente que foi construído ao redor do *ionic*, a produtividade é alta. Várias *tags* para os componentes mais comuns em uma aplicação *mobile*, o *layout* personalizado para cada plataforma é feito pelo *framework*, com várias ferramentas disponíveis para ajudar a criar, desenvolver e testar suas aplicações e a integração com o *Angular* deixa esse framework cada dia melhor e mais produtivo (JUNIOR, 2016).

Para Passos (2018), o *Ionic framework* para desenvolvimento de aplicações *mobile* híbridas é um aplicativo escrito com *Ionic* utiliza: *JavaScript*, Html, Css (Sass) e *Angular* (*TypeScript*).

Por ser um *framework front-end*, é possível utilizar os navegadores para o desenvolvimento. Fazendo bom uso das ferramentas de *debug* presentes nos navegadores é possível desenvolver boa parte dos aplicativos (tudo o que não faça uso das APIs nativas dos dispositivos móveis) JANONES (2016).

Com ele é possível criar aplicativos que usem funções nativas, isto é, vem por padrão graças ao *hardware* do dispositivo. Para isso, é necessário instalar *plugins*, que são os *Ionics Natives* e o *Apache Cordova*, com base no componente *WebView* e funciona como um *browser*, mas sem a barra de endereço ou botões para o usuário, assim por ele apenas é possível visualizar os dados. A vantagem do *Ionic* é que o projeto se adapta e muda a aparência conforme o sistema do dispositivo.

Entre as opções, é a mais barata e com maior velocidade de desenvolvimento. Assim como com *React Native*, não é necessário ter duas equipes para desenvolver aplicações diferentes, mas tem a vantagem de ser, sendo simplista, *HTML+JavaScript* dentro de um *webview*, conhecimento amplamente espalhado na comunidade de desenvolvedores (GIL, 2017).

Para Gil (2017), seu ponto fraco é a perda de *performance*, principalmente em aceleração do *scroll*, comportamento do teclado e navegação. Em certos cenários, esses problemas podem gerar frustração do seu usuário, mas para a maioria das pequenas, médias empresas ou quem ainda está validando uma ideia, é uma excelente solução. Já que o custo reduzido e a velocidade de desenvolvimento compensam.

2.2.2 *React Native*

Segundo Becker (2019), o *React Native* é um *framework* baseado no *React*, desenvolvido pela equipe do *Facebook*, que permite o desenvolvimento de aplicações *mobile*, tanto para *Android*, como para iOS, utilizando apenas *Javascript*. Todo o código desenvolvido é convertido para a linguagem nativa do sistema operacional.

Para Silva (2019), *React* é uma biblioteca *JavaScript* que possui ferramentas que facilitam a construção de *Interfaces* na *Web*. Ele transforma um ambiente complexo, cheio de *Edge Cases* (casos onde você precisa tratar eventos e como os dados são manipulados) em algo muito mais simplificado.

O *React* aborda o fato de que a lógica de renderização é inerentemente acoplada

à outra lógica da *interface* do usuário: como os eventos são manipulados, como o estado muda ao longo do tempo e como os dados são preparados para exibição. Em vez de separar tecnologias artificialmente, colocando a marcação e a lógica em arquivos separados, o *React* separa as preocupações com unidades fracamente acopladas chamadas "componentes" que contêm ambos.

Antes do surgimento do *React Native*, para criar aplicativos *Android* e *iOS* era algo relativamente complexo, pois era necessário de ter que aprender as linguagens *Objective-C* (*iOS*) e Java (*Android*), o que não permitia reaproveitar o código, porém, com o React Native, o código pode ser reaproveitado por completo entre as plataformas, podendo fazer com que o custo e a duração do projeto caiam pela metade.

Buscando facilitar para que diversos desenvolvedores consigam atingir o máximo de eficiência possível no desenvolvimento de sistemas e aplicativos, algumas tecnologias nascem dentro de grandes empresas para solucionar algumas dores específicas. Assim surgiu o *React,* que é uma biblioteca *JavaScript,* utilizado para elaborar todo o *Front-End* das aplicações, e que contribui para definir soluções de problemas, com eficiência e flexibilidade para a criação de interfaces de usuário (UI). (SILVA, 2019).

Uma característica importante do *ReactJS* é o formato do seu processamento, ele trabalha com os estados do seus elementos, assim podendo analisar o estado para que possa atualizar unicamente a parte necessária do *DOM*.Assim não fazendo necessário ser gasto nem a mémoria do lado do cliente e isso também se estende por parte do servidor, com a popularização do conceito de carregamento por partes que mais adiante iremos explicar, o uso dos componentes. (DANIEL e EDUARDO, 2018).

Para Cabral (2016), o *stack* do *React Native* é muito útil, pois permite utilizar *ECMAScript 6, CSS Flexbox, JSX*, diversos pacotes do *NPM*. Sem contar que também permite fazer debug na mesma *IDE* utilizada para o desenvolvimento nativo com essas plataformas.

4 Resultados

Ao observar os *frameworks* e o *SDK Flutter*, é possível perceber que React é muito utilizado no mercado atualmente e um dos principais motivos para ele ser tão utilizado é por ter sido criado pelo *Facebook*. Uma biblioteca de código aberto, que faz parte dos projetos open-source de uma empresa tão grande, faz com que ele realmente ganhe muita visibilidade. Com esse reconhecimento, muitas empresas acabam optando por adotar o *React*, pois sabem que é uma alternativa segura e que possui pessoas extremamente talentosas dedicadas a evoluir o projeto, construindo uma experiência de valor não somente para o usuário.

React Native é, atualmente, o mais próximo que uma aplicação híbrida consegue chegar de uma nativa. Consegue entregar nas duas plataformas dominantes (*iOS e Android*) produtos de muita qualidade, com performance e comportamento praticamente igual ao das aplicações nativas.

A maior vantagem do *React* vem do fato de não ser forçado a usar classes. As classes complicam demais a base de código sem fornecer nenhum benefício. O *React* não é apenas a interface do usuário, é uma estrutura/ecossistema.

5 Discussão

Por causa do atual mercado de software demandar cada vez mais que as empresas produzam apps leves, rápidos e que funcionem em múltiplas plataformas, ficou bem claro pela pesquisa que pelo fato de usar uma linguagem amplamente conhecida *(javascript*), por ter um suporte muito bom tanto no android como no ios, o React native tem uma certa vantagem sobre o *SDK* e *ionic.* Mesmo os *frameworks* não sendo tão versáteis

quanto o *React Native* , ainda possuem seus pontos fortes como a “simplicidade” e grande lista de componentes já disponibilizadas pelo próprio *Ionic* e a fluidez, velocidade de desenvolvimento e aprendizagem e os *widgets* do *Flutter* que faz com que também sejam bastante utilizados.

6 Conclusão

Devido ao progresso exponencial dos dispositivos eletrônicos e suas consequentes variedades de plataformas, se tornou de suma importância realizar um desenvolvimento de forma híbrida (aplicações híbridas), faz-se necessário então, ferramentas e linguagens que se adequem de forma eficaz a esse novo paradigma a fim de se possibilitar um custo-benefício, com boa eficiência e desempenho nas diversas plataformas disponíveis para desenvolvimento. Neste artigo tratamos de demonstrar que cada uma das ferramentas e linguagens apresentadas pode ser ideal de acordo com as necessidades do projeto e recursos de hardware, financeiros e humanos (conhecimento).

Referências

ABRANCHES, Junior. **Aplicativos e desenvolvimento mobile híbrido x nativo.** 2018. Disponível em: <https://imasters.com.br/desenvolvimento/aplicativos-e-desenvolvimento-mobile-hibrido-x-nativo> Acessado em: 22/09/2019 às 19:52.

BECKER, Lauro. **O que é React Native?**.2019. Disponível em: < >. Acessado em: 13/09/2019 às 18:48.

CABRAL, Carlos. **React Native: Construa aplicações móveis nativas com JavaScript**.2016. Disponível em: < https://tableless.com.br/react-native-construa-aplicacoes-moveis-nativas-com-javascript/ >. Acessado em: 13/09/2019 às 18:58.

CORAZZA, Paulo Victor. **Um aplicativo multiplataforma desenvolvido com Flutter e NoSQL para o cálculo da probabilidade de apendicite.** 2018. Disponível em: < http://nuvem.ufabc.edu.br/certificados/ii-workshop/2-workshop-nuvem-ufabc_paper_20.pdf > Acessado em: 23/08/2019 ás 19:55.

Ionic. **Ionic – Cross-Platform Mobile App Development.** 2019. Disponível em: <https://ionicframework.com/> Acessado em: 22/09/2019 às 20:28.

JANONES, Ramos de Souza. **Aplicativos mobile com o Angular e Ionic.** 2016. Disponível em: <https://www.ramosdainformatica.com.br/mundojs/ionic-framework/aplicativos-mobile-com-o-angularjs-e-ionic-framework/> Acessado em: 22/09/2019 às 21:30.

JUNIOR, Lazáro. **5 vantagens do Ionic para desenvolver suas aplicações mobile.** 2016. Disponível em: <https://www.alura.com.br/artigos/5-vantagens-do-ionic-para-desenvolver-suas-aplicacoes-mobile> Acessado em: 22/09/2019 às 20:18.

Madureira, Daniel. **Aplicativo nativo, web app ou aplicativo híbrido?** 2017. Disponível em: <https://usemobile.com.br/aplicativo-nativo-web-hibrido/> Acessado em: 22/09/2019 às 19:40.

Micreiros. **9 Frameworks de Desenvolvimento Multiplataforma Móvel.** 2019. Disponível em: <http://micreiros.com/9-frameworks-de-desenvolvimento-multiplataforma-movel/> Acessado em: 22/09/2019 às 20:38.

PASSOS, Clayton K. N. **Oque é o IONIC ?.**2018. Disponível em: < https://tableless.github.io/iniciantes/manual/js/o-que-framework.html>. Acessado em: 13/09/2019 às 18:28.

SANTANA, Fabiano. **Flutter: porque você deveria apostar nesta tecnologia.** 2019. Disponível em: <https://medium.com/tableless/flutter-porque-voc%C3%AA-deveria-apostar-nesta-tecnologia-94a510fffd18> Acessado em: 22/09/2019 às 20:45.

SILVA, Mateus. **Quais são os principais frameworks para desenvolver aplicações móveis híbridas?** 2018. Disponível em: <https://medium.com/@mattlack/quais-s%C3%A3o-os-principais-frameworks-para-desenvolver-aplica%C3%A7%C3%B5es-m%C3%B3veis-h%C3%ADbridas-2093b64d679e> Acessado em: 22/09/2019 às 19:30.

SILVA, Priscylla. **O que é React e para que serve? | Conversando com o CTO.** 2019. Disponível em: <https://gobacklog.com/blog/o-que-e-react-e-para-que-serve/> Acessado em: 22/09/2019 às 20:10.

Tableless. **O que é um Framework?**.2019. Disponível em: < https://tableless.github.io/iniciantes/manual/js/o-que-framework.html > . Acessado em: 13/09/2019 às 18:28.