UNICESUMAR - CENTRO UNIVERSITÁRIO DE MARINGÁ
CENTRO DE CIÊNCIAS EXATAS TECNOLÓGICAS E AGRÁRIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

OS DIFERENTES TIPOS DE DESENVOLVIMENTO MOBILE, UMA VISÃO PRÁTICA DO MULTIPLATAFORMA.

GUILHERME PRADO NUNES DE OLIVEIRA

MARINGÁ – PR
2019

Guilherme Prado Nunes de Oliveira

OS DIFERENTES TIPOS DE DESENVOLVIMENTO MOBILE, UMA VISÃO PRÁTICA DO MULTIPLATAFORMA.

Artigo apresentado ao Curso de Graduação em Engenharia de Software da UNICESUMAR – Centro Universitário de Maringá como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software, sob a orientação do Prof. Me. Arthur Cattaneo Zavadski.

MARINGÁ – PR

2019

FOLHA DE APROVAÇÃO

GUILHERME PRADO NUNES DE OLIVEIRA

OS DIFERENTES TIPOS DE DESENVOLVIMENTO MOBILE, UMA VISÃO PRÁTICA DO MULTIPLATAFORMA.

Artigo apresentado ao Curso de Graduação em Engenharia de Software da UNICESUMAR – Centro Universitário de Maringá como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software, sob a orientação do Prof. Me. Arthur Cattaneo Zavadski.

Aprovado em: ____ de _____ de _____.

BANCA EXAMINADORA

_____

Nome do professor - (Titulação, nome e Instituição)

_____

Nome do professor - (Titulação, nome e Instituição)

_____

Nome do professor - (Titulação, nome e Instituição)

OS DIFERENTES TIPOS DE DESENVOLVIMENTO MOBILE, UMA VISÃO PRÁTICA DO MULTIPLATAFORMA.

Guilherme Prado Nunes de Oliveira

RESUMO

A crescente utilização de dispositivos móveis e sua aplicação em modelos de negócio inovadores, têm gerado grande demanda de desenvolvimento de aplicativos. A existência de plataformas móveis diferentes impõe aos desenvolvedores a necessidade de elaborar projetos para diversos sistemas operacionais, sendo que cada um oferece suas próprias ferramentas nativas de desenvolvimento. Neste cenário, o código é específico para a plataforma, exigindo que sejam desenvolvidos projetos distintos para Android (Java ou Kotlin) e iOS (Objective-C ou Swift). Como forma de minimizar este custo surgiram ferramentas que possibilitam o desenvolvimento multiplataforma, no qual um único código fonte gera o mesmo aplicativo para as diferentes plataformas. Visando apresentar as vantagens, desvantagens e pontos de atenção na aplicação de ferramentas para desenvolvimento móvel multiplataforma, conclui-se que, atualmente e nos cenários abordados, este é uma alternativa viável ao desenvolvimento desdobramento nativo.

PALAVRAS-CHAVE: Desenvolvimento. Dispositivos móveis. Multiplataforma. Nativo.

THE DIFFERENT TYPES OF MOBILE DEVELOPMENT, A PRACTICAL VIEW OF THE MULTIPLATFORM.

ABSTRACT

The growing use of mobile devices, and their application in innovative business models, has generated great demand for application development. The existence of different mobile platforms imposes on developers the need to design projects for different operating systems, each offering its own native development tools. In this scenario the code is platform specific, requiring that separate projects be developed for Android (Java or Kotlin) and iOS (Objective-

C or Swift). As a way to minimize this cost came tools that enable cross-platform development, in which a single source code generates the same application for different platforms. In order to present the advantages, disadvantages and points of attention in the application of tools for multiplatform mobile development, it is concluded that, currently and in the scenarios covered, multiplatform development is a viable alternative to native development.

KEYWORDS: Development. Mobile devices. Multiplatform. Native.

SUMARIO

1 INTRODUÇÃO	**6**
1.1 JUSTIFICATIVA	**7**
1.2 OBJETIVOS	**7**
1.3 RESULTADOS ESPERADOS	**8**
2 ANÁLISE SOBRE O DESENVOLVIMENTO MULTIPLATAFORMA	**8**
3 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS	**9**
3.1 DESENVOLVIMENTO NATIVO	**10**
3.2 iOS	**11**
3.3 ANDROID	**14**
4 DESENVOLVIMENTO MULTIPLATAFORMA	**16**
4.1 CORDOVA	**17**
4.2 REACT NATIVE	**18**
5. ANÁLISE EXPLORATÓRIA	**18**
6. METODOLOGIA	**22**
7. CONCLUSÃO	**23**
8. REFERÊNCIAS	**24**

1 INTRODUÇÃO

Nos últimos anos, o uso de smartphones e dispositivos móveis acabou alterando alguns hábitos na comunicação da população, apoiando as pessoas diariamente em suas atividades pessoais e profissionais, sejam elas em uma rede social ou até mesmo escutando músicas. Já no viés profissional é utilizado em dashboards de acompanhamento da empresa, até mesmo na leitura de jornais, aponta Cevallos (2014).

Anteriormente, para se desenvolver um aplicativo para dispositivos móveis, possuíamos apenas um tipo de abordagem que é conhecido como desenvolvimento nativo. Segundo El-Kassas et al, (2015) nesta abordagem, os aplicativos são desenvolvidos utilizando ferramentas e linguagens de programação exclusivas para aquele tipo de plataforma móvel, seguindo os padrões de interface gráfica e claro, oferecendo um ambiente mais agradável ao usuário, uma vez que os aplicativos daquela plataforma seguem um padrão de usabilidade, maximizando os recursos do dispositivo.

Porém, com este aumento na utilização de dispositivos móveis temos mais de uma plataforma no mercado, sendo as principais Android, com aproximadamente 58% do setor, 33% voltado ao iOS e o restante distribuído entre Windows Phone e BlackBerry (JOBE, 2013). Devido às diversas plataformas, existe a necessidade de desenvolver o mesmo app mais de uma vez, o que acaba encarecendo o custo, pois é necessário ter conhecimento para utilizar as diferentes ferramentas, específicas de cada plataforma.

Para resolver este problema, surgiu o desenvolvimento multiplataforma, que segundo El-Kassas, et al, (2015), veio para auxiliar os desenvolvedores, pois, com apenas um código fonte conseguimos desenvolver um mesmo app, implantando-o em diferentes plataformas reduzindo tempo e esforço de desenvolvimento. Esse meio de desenvolvimento utiliza algumas tecnologias WEB bem conhecidas, sendo elas o HTML (HiperText Markup Language), CSS (Cascade Style Sheet) e JavaScript utilizados em lógica de negócios e interfaces do usuário consoante Biorn-Hansen (2018).

Uma das soluções abordadas nesse trabalho que resolve essa problemática é o React Native, que, conforme aponta Charland; Leroux (2011) permite criar aplicativos móveis utilizando o JavaScript e React, um framework de desenvolvimento. Os aplicativos criados com React Native não são aplicativos Web como o Cordova em que se é utilizado o HTML, uma vez que faz uso dos mesmos blocos de construção da interface iOS e Android. Em vez de empregar Swift, Kotlin ou Java, que são as linguagens aplicadas no desenvolvimento nativo, você está colocando esses blocos usando JavaScript e React. Temos algumas empresas que sentiram a necessidade de ter seus aplicativos em mais de uma plataforma que recorreram ao React Native sendo elas o Facebook, Instagram, Skype, Uber entre outras grandes empresas que estão no mundo mobile.

1.1 JUSTIFICATIVA

Com o crescente aumento na utilização de dispositivos móveis de diferentes plataformas, e com a diversidade de tecnologias no mercado, torna-se necessário abranger a maior quantidade de plataformas, atingindo o menor tempo de desenvolvimento, com um valor reduzido, bem como o melhor custo benefício. Esse trabalho, então, analisa as plataformas Android e iOS, com o intuito de auxiliar o desenvolvedor na hora de identificar a tecnologia mais adequada a ser utilizada em seu projeto.

1.2 OBJETIVOS

O objetivo desse Trabalho de Conclusão de Curso é realizar uma análise exploratória entre o desenvolvimento nativo e o híbrido, expondo aspectos positivos e negativos de cada uma dessas abordagens, mostrando os fatores que podem influenciar na escolha da tecnologia a ser utilizada no desenvolvimento de um aplicativo móvel, tendo como objetivos específicos:

- Realizar uma pesquisa que será disseminada, por meio de um formulário online para os desenvolvedores, buscando compreender sobre as tecnologias mais utilizadas no momento.
- Inquirir sobre as tecnologias React Native, Cordova e o desenvolvimento nativo, compreendendo como as mesmas funcionam.
- Compreender as diferentes arquiteturas dos dispositivos móveis mais utilizados atualmente.
- Analisar o desdobramento nativo do desenvolvimento multiplataforma.
- Criar um app nas tecnologias React Native, Cordova e nativo, de modo a avaliar até que ponto uma aplicação multiplataforma se aproxima de uma nativa, visando o acesso ao hardware do dispositivo e os esquemas de interface nas diferentes plataformas.

1.3 RESULTADOS ESPERADOS

Almeja-se, com esse trabalho, elencar alguns pontos positivos e negativos de ambas abordagens, nativa e multiplataforma, auxiliando um desenvolvedor, na hora de escolher a tecnologia que lhe trará melhores benefícios em seu aplicativo, com um bom desempenho de hardware, que atenda às expectativas do usuário.

2. ANÁLISE SOBRE O DESENVOLVIMENTO MULTIPLATAFORMA

De modo geral, o desenvolvimento multiplataforma não iniciou na era do mobile, pois esse assunto já era discutido há algum tempo, uma vez que existe mais de um S.O para computadores. Um exemplo de linguagem de desenvolvimento para desktop é o .NET, que até algum tempo atrás não era passível de execução em plataformas que não no Windows, sendo um ponto que barrava o crescimento dessa ferramenta de desenvolvimento. É

interessante para os desenvolvedores .NET que se possa amplificar aplicações para Linux, por exemplo, sem a necessidade de aprender uma nova tecnologia, algo que é possível devido ao runtime ser baseado em uma máquina virtual denominada CLR.

Um outro exemplo é linguagem de desenvolvimento Java, que também se utiliza de uma máquina virtual chamada de JVM. Uma máquina virtual é capaz de simular um ambiente de máquina física, tendo acesso à gerência de processos, memória e processador. Dessa forma as empresas Oracle/Sun acabam solucionado o problema de rodar softwares em diferentes sistemas operacionais, uma vez que só era possível executar esta máquina virtual em Windows/Linux. Vendo o leque de aberturas que o desenvolvimento multiplataforma atrai, a Microsoft, empresa responsável pelo .NET, criou um projeto chamado de Projeto Mono que visava o desenvolvimento multiplataforma da mesma, uma vez que consiste em uma implementação Open Source do .NET, podendo ser utilizado em máquinas com diferentes SO's. (HOELLER, 2010).

3. DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

Aplicativos mobile são softwares utilizados para funções específicas em dispositivos móveis como smartphones e tablets. Eles estão disponíveis através de plataformas de distribuição de aplicações que são normalmente operadas pelo proprietário do sistema operacional móvel, como App Store, Play Store, entre outros.

Segundo Corral; Janes; Remencius (2012), os aplicativos móveis são uma importante extensão dos softwares atuais que estão no mercado, onde envolvem uma vasta gama de aplicações em diversos ramos, sendo eles a comunicação, entretenimento, negócios, mídia entre outros. Porém estamos em um ambiente onde não podemos contar com o hardware, portanto os desenvolvedores devem ter em mente aspectos como a capacidade e especificações dos dispositivos móveis, a mobilidade, o design e navegabilidade de interface gráfica, segurança e privacidade do usuário.

Conforme aponta El-Kassas, et al (2015) devemos considerar alguns pontos na hora de desenvolver um aplicativo mobile, pois existem muitas restrições e desafios, sendo eles:

●Recursos limitados dos dispositivos móveis: Em termos de espaço de armazenamento e poder de computação somos limitados, uma vez que estamos restritos ao hardware do dispositivo.

●Diversidade de sistemas operacionais: Diferentes ambientes operacionais para desenvolver o mesmo aplicativo estão no mercado. O desenvolvedor precisa aprender a desenvolver em diferentes SDKs do sistema operacional usando suas linguagens e APIs de programação.

●Experiência do usuário: Os desenvolvedores têm que definir uma interface simples e amigável para os aplicativos desenvolvidos atendendo diversos tipos de públicos, desde um jovem até um idoso.

●Manutenção nos aplicativos: Atualizações frequentes da plataforma móvel podem afetar os aplicativos, tornando-os inutilizáveis na nova versão do sistema, portanto, a manutenção e as atualizações desses aplicativos são necessárias, uma vez que devemos pensar no usuário final, não deixando que o uso do app seja interrompido.

●Loja de aplicativos: É necessário um conhecimento sobre as lojas das plataformas utilizadas por parte dos desenvolvedores, uma vez que ambas divergem nas configurações de upload da loja. As lojas são essenciais para os usuários, uma vez que é a interface entre usuário e aplicativo, sendo um meio de instalação do mesmo.

3.1. DESENVOLVIMENTO NATIVO

De acordo com Heitkötter; Hanschke; Majchrzak (2012), ao desenvolver aplicativos de forma nativa, os desenvolvedores implementam um aplicativo para uma plataforma de destino específica usando seu kit de desenvolvimento de software (SDK) e frameworks. Desta forma, nos aplicativos feitos para a plataforma android é utilizada a linguagem de programação Java que faz uso

das funcionalidades que o dispositivo Android oferece. Já em contraste com o Android, há os aplicativos para iOS que são empregados na linha de celulares da Apple, que utilizam as linguagem de programação conhecida como Objective-C, que também dá acesso ao hardware dos dispositivos da Apple. Quando se deseja desenvolver um mesmo aplicativo para as diferentes plataformas, julga-se necessário desenvolver a aplicação separadamente para cada uma.

Como o aplicativo é desenvolvido para aquela plataforma de modo específico, são seguidos padrões de usabilidade e projeto, oferecendo um melhor desempenho, maximizando o uso do hardware e recursos do dispositivo. Porém, como cada S.O deve ter seu desenvolvimento específico, acaba sendo mais custoso em relação ao tempo de trabalhado, já que se deve iniciar um desenvolvimento do zero para cada uma. (CEVALLOS, 2014).

É necessário ressaltar que as duas plataformas Android e iOS são muito utilizadas e se possível, você deve oferecer a aplicação nas duas plataformas. No entanto, é necessário considerar certas condições para realizar uma aplicação mais assertiva, principalmente quando os recursos iniciais são curtos, uma vez que o custo de desenvolvimento para a plataforma Android é mais acessível. (EL-KASSAS, et al, 2015)

Nas próximas sessões, serão apresentadas as plataformas Android e iOS com seus elementos arquiteturais.

3.2. iOS

De acordo ao apontamento de Pilar (2013), o iOS é o sistema operacional da Apple, que originalmente foi desenvolvido para o iPhone, e atualmente está presente em todos os dispositivos móveis da empresa, tais como tablets, celulares, reprodutores de músicas e até mesmo em relógios. O iOS foi lançado oficialmente em Junho de 2007 e em Outubro desse ano, a Apple anunciou o SDK (Software Development Kit), que contém ferramentas e recursos para o desenvolvimento de aplicativos iOS nativos. Este SDK

possibilita que ocorra desenvolvimento de apps por terceiros, onde é possível instalar, executar e testar aplicativos no simulador ou no dispositivos físico.

Segundo Rocha; Neto (2011), a arquitetura do iOS é baseada em quatro camadas, sendo que nas camadas superiores são oferecidas abstrações orientadas a objetos, que acaba tornando o desenvolvimento mais simples e eficiente. Porém, caso os recursos das camadas superiores não forem satisfatórios, é possível utilizar as mais inferiores, que permitem um maior controle sobre os recursos do dispositivo.

A camada mais superior é chamada de Cocoa Touch. De acordo com Pilar (2013), essa camada é formada por frameworks que oferecem as principais funcionalidades para o desenvolvimento, tais como multitarefa, proteção de dados e serviço de notificação push e diversos outros serviços de alto nível do sistema. O serviço de proteção de dados, por exemplo, permite que os aplicativos trabalhem com esses de maneira criptografada, quando é armazenado num arquivo em disco. Enquanto o dispositivo estiver bloqueado, esse arquivo fica inacessível para o aplicativo e potenciais intrusos, quando o dispositivo é desbloqueado pelo usuário, uma chave de decodificação é criada, liberando o acesso para aquele arquivo. (PILAR, 2013)

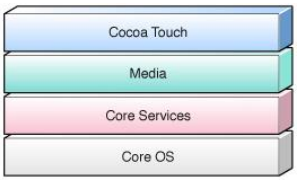
A camada Media Services, possui tecnologias de áudio, vídeo e gráficos complexos e gráficos 3D renderizados em tempo de execução, sendo que, com estes frameworks, torna-se mais fácil a implementação de aplicativos que utilizam multimídia. Nessa camada são encontradas bibliotecas como OpenGLES (biblioteca de gráficos tridimensionais), QuartzCore, fornecendo um alto padrão de renderização e animações suaves, sem sobrecarregar a CPU e desacelerando seu aplicativo. (MENDONÇA; BITTAR; DIAS, 2011)

A camada abaixo, chamada de Core Services, é responsável pela gestão dos serviços fundamentais que um aplicativo iOS nativo emprega. Mesmo que não seja utilizada diretamente, existe muita dependência do sistema em cima dessa camada. Aqui temos acesso a algumas funcionalidades, tais como SQLite onde é possível criar um banco de dados para utilizar na aplicação, localização, in-app purchase para realizar vendas dentro do aplicativo. (ROCHA; NETO, 2011)

A última camada, mas não menos importante, é a Core OS, que possui entrada para o baixo nível, dando acesso à criação de outros frameworks. Essa camada se comunica diretamente com o hardware, dando acesso ao Accelerate, que oferece computação de alto desempenho e eficiência de energia na CPU. A camada tem, encapsulado, o kernel e as interfaces do UNIX, porém estas não são liberadas por razão de segurança. (ROCHA; NETO, 2011)

Abaixo temos uma imagem exemplificando as camadas da arquitetura iOS.

Figura 1 – Arquitetura iOS.



Fonte: O autor.

De acordo com Bezerra; Schimiguel (2016), para se desenvolver um aplicativos para o iOS, é necessário possuir um dispositivo com o sistema operacional Mac OS onde é utilizado o Xcode, uma IDE de desenvolvimento da Apple. É necessário conhecimento nas linguagens de programação Objective-C e Swift, já que ambas são utilizadas para o desenvolvimento de um aplicativo iOS nativo.

Após a finalização do aplicativo, é necessário submetê-lo à avaliação da Apple e, para isso, é necessária uma licença de desenvolvedor Apple, que atualmente custa em torno de 99 dólares anuais. Nesta avaliação, ocorrem algumas revisões internas da Apple que são realizadas por humanos, quando é verificado se a aplicação está de acordo com as diretrizes da Apple, que são orientações de interfaces gráficas, não ter conteúdo inadequado sem avisos prévios e até mesmo citações de concorrentes da própria Apple. (BEZERRA; SCHIMIGUEL, 2016)

3.3. ANDROID

O sistema operacional conhecido como Android, teve início na companhia Android Inc. em dezembro de 2003 e foi fundado em setembro de 2006, juntamente com o SDK para a plataforma. Inicialmente, a proposta da Android Inc era voltada para câmeras digitais, porém, com a validação de mercado, não se tornou financeiramente viável e acabaram desviando o foco para um sistema operacional para dispositivos móveis, gerando uma disputa de mercado com o Symbian e Windows Phone. Conforme o projeto começou a ocorrer, os investimentos da empresa se extinguiram, assim a empresa Google adquiriu a empresa Android Inc. em agosto de 2005, continuando o projeto voltado para sistemas operacionais para dispositivos móveis, baseado no kernel do Linux. (Android Inc., 2019)

Atualmente, o sistema operacional Android é uma operação Open Source, isso significa que qualquer pessoa pode utilizar esse código e realizar as alterações que forem necessárias para executar o mesmo em um dispositivo. (NAIR; YADAV, 2018). De acordo com Bordin (2012), a arquitetura do sistema operacional Android utiliza o Kernel Linux para prover serviços e aplicações, porém foi modificada visando atender às necessidades existentes em um dispositivo móvel, como gerenciamento de memória, energia e o ambiente de execução que inicialmente operava com restrições de hardware. Esse sistema é constituído por uma pilha de softwares onde atualmente existem 5 camadas:

A camada mais baixo nível é chamada de Kernel e atua gerenciando processamento de memória, processos e segurança e ainda trabalha como uma camada de abstração entre hardware e as camadas superiores, permitindo o acesso a recursos como áudio, vídeo e os protocolos de rede (PILAR, 2013).

Acima desta camada, existe a camada denominada HAL, onde é fornecido um padrão de interfaces que são expostas as capacidades de hardware do dispositivo, que consiste em bibliotecas específicas como câmera,

bluetooth e sensores, tornando-se mais fácil criar novas funcionalidades isoladas das outras camadas (Android, 2019).

Acima da camada HAL, existem mais duas, a Runtime Android e as Bibliotecas C/C++ nativas, cada uma com uma função distinta. A Runtime Android tem como objetivo executar várias máquinas virtuais em dispositivos de baixa memória, executando arquivos DEX, um formato de bytecode projetado especialmente para Android, otimizado para oferecer consumo mínimo de memória e fornecer um conjunto de bibliotecas que permitam aos desenvolvedores escrever aplicativos usando a linguagem de programação Java.

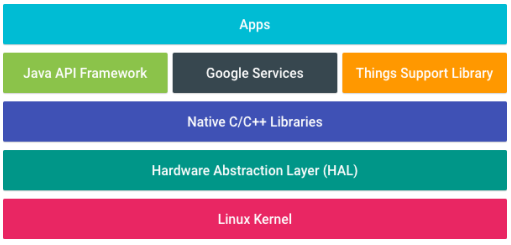
Como a camada HAL e a Runtime Android executam códigos nativos que exigem bibliotecas nativas programadas em C e C++, a camada das bibliotecas, é onde fica armazenado o código fonte das funcionalidades da tecnologia, como, por exemplo, bibliotecas para suporte ao banco de dados e funcionalidades para navegação web (Android, 2019).

Segundo Bordin (2012), na camada Java API Framework, ficam os frameworks de aplicações que fornecem as funcionalidades necessárias para a construção dos aplicativos, através de bibliotecas nativas.

A última camada, segundo Android (2019), são os aplicativos do sistema, onde ficam armazenados todos os aplicativos instalados pelo usuário e ainda um conjunto de aplicativos principais para e-mail, envio de SMS, calendários, navegador de internet, contatos que já vêm por padrão no dispositivo.

Abaixo, temos uma imagem exemplificando as camadas da arquitetura Android.

Figura 2 – Arquitetura Android.



Fonte: O autor.

De acordo com Mendonça; Bittar; Dias (2011), o desenvolvimento para a plataforma Android, diferente do iOS, pode ser desenvolvido em qualquer sistema operacional, seja ele Windows, Linux ou até mesmo Mac OS X, que além da flexibilidade no sistema operacional, o desenvolvedor pode utilizar a IDE que tem preferência. Com isso ocorre um maior alcance de desenvolvedores da plataforma, uma vez que também é Open Source, sendo necessário apenas conhecimento na linguagem de programação Java, que é a utilizada no desenvolvimento de um aplicativo Android nativo.

Como ocorre no iOS, no Android também é necessária uma licença para submeter o aplicativo na loja PlayStore, que atualmente está custando 25 dólares, pagos uma única vez. A avaliação do aplicativo para ficar disponível na loja é bem mais simples do que a da Apple, uma vez que não é realizada uma conferência tão rigorosa, porém ainda existe uma averiguação de que não se violam as políticas estabelecidas pela empresa responsável (BORDIN, 2012).

4. DESENVOLVIMENTO MUTIPLATAFORMA

De acordo com Heitkötter; Hanschke; Majchrza (2012), o desenvolvimento multiplataforma permite que desenvolvedores implementem apenas um código base e executem em mais de uma plataforma, seja ela Android ou iOS. Esses tipos de aplicações são categorizadas em webapps e aplicações híbridas.

Podemos identificar dois tipos diferentes desse desenvolvimento. O primeiro é baseado em HTML, CSS e JavaScript, utilizando o browser do dispositivo como ambiente de execução. Já o híbrido é compilado e executado de forma nativa dentro da aplicação.

Conforme Heitkötter; Hanschke; Majchrzak, (2012), aplicações webapp costumam ter o comportamento e a aparência das páginas web, portanto alguns comportamentos divergem de uma aplicação desenvolvida no híbrido e

nativo, ainda mais quando comparamos nas diferentes plataformas Android e iOS, que possuem browsers padrão em cada uma das plataformas.

Segundo Stark (2010), a abordagem híbrida é uma combinação entre o webapp e as funcionalidades nativas de cada uma das plataformas, ou seja, a aplicação desenvolvida é empacotada em um aplicativo nativo, de forma a implementar o mesmo.

4.1. CORDOVA

O Cordova é um framework que possui o objetivo de, com apenas um código-fonte, criar aplicações mobile multiplataforma utilizando HTML, CSS, JavaScript, mais conhecidos como webapp. Dentre suas vantagens, podemos apontar a velocidade com que se é possível criar uma aplicação mobile, uma vez que não é necessário o conhecimento da linguagem nativa da plataforma, aponta (BEZERRA; SCHIMIGUEL, 2016).

De acordo com Biorn-Hansen (2018), este framework possui um repositório com milhares de plugins disponíveis, incluindo câmera, GPS, entre outras funcionalidades, ou seja, concedem o acesso às funcionalidades específicas do hardware, que agem como uma ponte entre a aplicação e o hardware propriamente dito. As mesmas são escritas de forma nativa a serem compiladas dentro da aplicação, e podem ser instaladas por meio de um CLI (Comand Line Interface), disponibilizado pela própria empresa.

Diante de suas vantagens, Holzinger; Treitler; Slany (2012) aponta que o Cordova possui uma vasta documentação de apoio aos desenvolvedores que possuem interesse em utilizar o mesmo, ainda mais por ser um framework considerado maduro o suficiente, devido ao seu tempo de mercado, que atualmente é em torno de 10 anos. Uma outra vantagem é a possibilidade de combinar o Cordova com outras ferramentas de desenvolvimento, como Angular e JQuery, para produzir uma aplicação com uma melhor interface.

4.2 REACT NATIVE

De acordo com Charland; Leroux, 2011), o React Native é uma plataforma desenvolvida pelo Facebook e de acesso à comunidade, uma vez que é open source. Possibilita aos desenvolvedores gerar em um único código, um aplicativo idêntico para o Android e iOS utilizando JavaScript, e React, porém executa o código nativo. Essa plataforma não depende de HTML e CSS, o código é escrito em JavaScript e é compilado para o código nativo específico da plataforma, porém, diferente do Cordova, é otimizado para que o aplicativo funcione sem problemas em 60 fps.

A plataforma possui acesso direto ao hardware do dispositivo, uma vez que, apesar de ser desenvolvido em JavaScript, é gerado um código nativo para as diferentes plataformas, possibilitando uma melhor comunicação entre elas e o hardware. O React Native possui também um repositório de bibliotecas com componentes que são específicos de cada plataforma, como ToastAndroid ou PushNotificationIOS, por exemplo. Porém, ainda assim não se torna necessário conhecimento da linguagem de programação da plataforma específica, uma vez que cada componente possui uma interface para comunicação com o JavaScript. (Facebook Inc., 2019)

5. ANÁLISE EXPLORATÓRIA

Nesta análise exploratória foram criados dois aplicativos utilizando as tecnologias Cordova e React Native e, um terceiro, fazendo uso do desenvolvimento nativo. Ambos com as mesmas funcionalidades, visando um estudo de performance, e a aderência ao esquema de interface, independente da plataforma utilizada, com o objetivo de melhorar o embasamento para os desenvolvedores na tomada de decisão entre o desenvolvimento nativo ou multiplataforma. Esses aplicativos criados têm apenas duas funcionalidades: a) o acesso ao GPS do dispositivo que captura sua latitude e longitude e

apresenta em tela, e b) com base na câmera do dispositivo, é tirada uma foto e salva no disco do celular.

Abaixo é apresentado um gráfico para facilitar a compreensão da comparação realizada entre o desenvolvimento das três plataformas em um teste de stress padrão com 10 repetições, realizado de forma síncrona entre as funcionalidade de busca das coordenadas e a captura da fotografia:

Tabela 1 - Estudo comparativo.

Plataforma	Cordova	React Native	Nativo
Tempo de processamento	216,64ms	134,43ms	114,24ms
Uso de CPU	5,5%	3,51%	3,28%
Uso de memória	37.61%	36,28%	36,83%
Tamanho do APK	2,4MB	6,8MB	1,8MB
Quantidade de linhas de código	54	60	114

Fonte: O autor.

Conforme se pode constatar nessa pesquisa, não existe uma grande divergência entre os dados. A maior diferença que podemos verificar, foi no tamanho do apk entre as diferentes plataformas, uma vez que o maior apk foi no React Native. O React Native, por sua vez, acaba levando módulos javascript que possuem dependências de terceiros, o que acaba tornando o apk um pouco maior em relação às outras duas abordagens verificadas.

O Cordova, por sua vez, ficou com o maior uso tanto de CPU, memória, e tempo de processamento. Isso acontece devido ao aplicativo renderizar em um browser e não ser aperfeiçoado como um que simula comportamentos nativos tal qual nas outras abordagens. O tempo de processamento maior no Cordova ocorre devido aos acessos de hardware ocorrerem por meio de uma API

nomeada como “Cordova Native API”, já nas outras plataformas o acesso ao hardware se dá por meio direto, entre aplicativo e hardware.

Em relação aos trechos de código, a aplicação nativa possui maior extensão, uma vez que toda a parte de interface é escrita em código XML e Java. De outro modo, o Cordova é escrito em trechos HTML e JavaScript, um facilitador, pois a maioria dos desenvolvedores já teve algum tipo de contato com o desenvolvimento web. Já o React Native é escrito em códigos JavaScript com tags simulando o nativo. Abaixo, temos os trechos de código de ambas as abordagens:

Figura 3 – Código React Native.

```
import React, { Component } from 'react'
import Geolocation from '@react-native-community/geolocation';
import ImagePicker from 'react-native-image-picker';
import { SafeAreaView, StyleSheet, ScrollView, View, Button, Text, Alert } from 'react-native'

export default class main extends Component {
  getCoordinates = () => {
    Geolocation.getCurrentPosition(res => {
      Alert.alert(`Sucesso`, `Latitude: ${res.coords.latitude}, Longitude: ${res.coords.longitude}`)
    }, error => {
      Alert.alert(`Ops...`, `${error}`)
    }, {})
  }

  render() {
    return (
      <View style={styles.body}>
        <View style={styles.marginHorizontal}>
          <Button onPress={() => this.getCoordinates()} title='GPS' />
        </View>
        <View style={styles.marginHorizontal}>
          <Button onPress={() => this.openCamera()} title='CAMERA' />
        </View>
      </View>
    )
  };
}
```

Fonte: O autor.

Figura 4 – Código Cordova.

```
<div class="row-center" style="height: 100%">
  <div class="col-20 row-center">
    <button onclick="openGps()" class="green">GPS</button>
  </div>
  <div class="col-20 row-center">
    <button onclick="takePicture()" class="green">Câmera</button>
  </div>
</div>
<script>
  function takePicture() {
    navigator.camera.getPicture(function (imageData) {
      var image = document.getElementById('myImage')
      image.src = '' + imageData
    }, function (message) {
      alert('Failed because: ' + message)
    }, {
      quality: 100,
      saveToPhotoAlbum: true
    })
  }
</script>
```

Fonte: O autor.

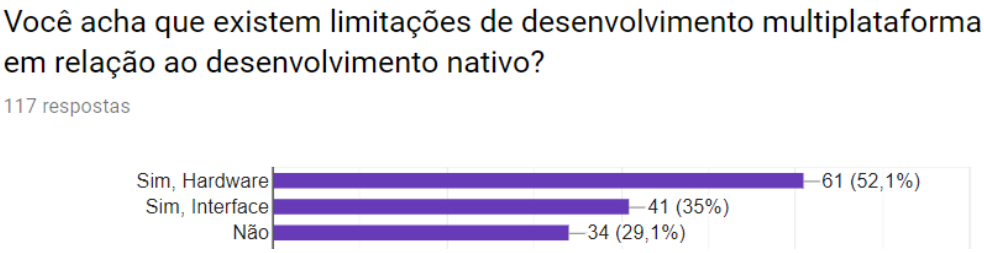
Figura 5 – Código nativo.

```
public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button gpsButton = findViewById(R.id.button2);
      gpsButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          if (ContextCompat.checkSelfPermission(getApplicationContext(),
                  Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_DENIED
          ) {
              ActivityCompat.requestPermissions(MainActivity.this, new String[]{
                      Manifest.permission.ACCESS_FINE_LOCATION
              }, GPS_REQUEST_CODE );
              return;
          }

          LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
          Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
          String text = "Latitude = " + location.getLatitude() + "Longitude = " +
location.getLongitude();
          new AlertDialog.Builder(MainActivity.this)
                  .setTitle("GPS")
                  .setMessage(text)
                  .setCancelable(true)
                  .show();
      }
    });
  }
}
```

Fonte: O autor.

Com base na pesquisa de campo disponibilizada, constatou-se que os desenvolvedores, atualmente, acreditam na existência de limitações no desenvolvimento multiplataforma em relação ao desenvolvimento nativo, conforme é possível observar no gráfico abaixo.

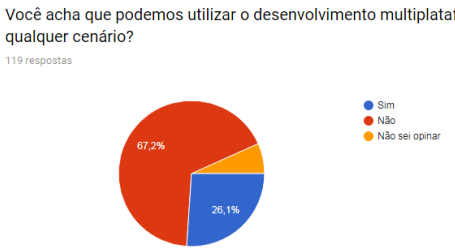
Figura 3 – Limitações do desenvolvimento multiplataforma.



Fonte: O autor.

Diante disso, questionou-se quanto a possibilidade de uso do desenvolvimento multiplataforma em qualquer cenário, considerando as limitações acima citadas e, novamente, foi possível constatar (Figura 4) que a maioria dos desenvolvedores acreditam não ser possível esta utilização.

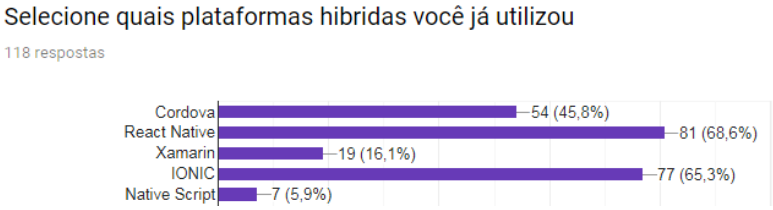
Figura 4 – Utilização do multiplataforma em qualquer cenário.



Fonte: O autor.

Conforme pode ser verificado no gráfico abaixo, todas as respostas obtidas na pesquisa realizada, foram dadas por usuários que já atuaram profissionalmente com o emprego de algum framework multiplataforma.

Figura 5 – Utilização de frameworks multiplataforma.



Fonte: O autor.

6. METODOLOGIA

A abordagem dessa pesquisa pode ser identificada como quantitativa, por ter como intuito compreender e elencar as diferentes abordagens tecnológicas no desenvolvimento móvel, com base em informações teóricas, que consiste na pesquisa de obras, livros, artigos de periódicos e documentos eletrônicos. Em relação à natureza, é caracterizada como pesquisa aplicada, gerando conhecimento para identificar a tecnologia mais adequada na construção de um projeto mobile.

No que se refere aos objetivos, este trabalho é de cunho descritivo uma vez que se pretende selecionar prós e contras para um contexto específico, o desenvolvimento móvel. Para a coleta de dados, foram utilizadas as técnicas de questionário, por meio de um formulário online.

7. CONCLUSÃO

Na primeira fase desse trabalho, foi realizada uma pesquisa sobre o desenvolvimento nativo e multiplataforma para compreender ambas as abordagens. Ao final do estudo teórico, foi realizado um estudo prático, para validar a pesquisa, a fim de obter uma comprovação das diferenças encontradas.

Em relação ao esquema de interfaces, quando nos referimos ao Cordova, torna-se necessária a utilização de um framework que adapte os componentes da plataforma utilizada, ou seja, o loading no iOS deve se comportar de forma diferente ao comportamento no Android. Como previamente descrito acima, existem frameworks compatíveis com o Cordova que trata esta particularidade mais acessível ou até mesmo automática.

Atualmente ambas as plataformas abordadas, seja nativa ou React Native, comportam-se de forma semelhante, quando comparadas ao acesso ao hardware do dispositivo, excepcionalmente para o Cordova que possui uma diferença de performance, uma vez que aplicativos desenvolvidos de forma nativa têm acesso direto ao hardware do dispositivo, enquanto para o Cordova esse acesso se dá por meio de plugins instalados.

Atualmente, o envio para as lojas ocorre de maneira igual para ambas as plataformas, apenas uma ressalva quando se trata de aplicativos enviados para a Apple Store, quando ainda se torna necessária uma máquina da Apple para compilar e enviar o aplicativo para a loja.

Ainda assim, aplicativos multiplataforma apresentaram mais vantagens do que desvantagens, comparados com o desenvolvimento nativo. As

funcionalidades que foram planejadas para o experimento prático no início do projeto foram desenvolvidas de forma congruente.

Portanto, podemos concluir que, pelos dados obtidos nessa pesquisa, o desenvolvimento multiplataforma se encaixa em todos os cenários em que seria necessário o desenvolvimento nativo, tendo em vista as vantagens descritas, ou seja, podemos criar um aplicativo mais rápido para a mesma plataforma de uma única vez. A quantidade de código é relativamente menor para o desenvolvimento multiplataforma, porém existem algumas ressalvas em relação a aplicações como jogos ou até mesmo as que utilizam realidade aumentada, uma vez que se é necessário um maior processamento e provavelmente ainda não se exista componentes para tratar essas situações Sendo assim, devemos analisar o cenário desejado, de forma a utilizar um ou outro, a fim de garantir o sucesso do projeto a ser desenvolvido.

8. REFERÊNCIAS

Android Inc. **Android Interfaces and Architecture | Android Open Source Project**. 2019. Disponível em: < https://source.android.com/devices/architecture>.

Apple Inc. **Apple Developer Program - Apple Develope**r. 2019. Disponível em: <https://help.apple.com/xcode/mac/current/#/devc8c2a6be1>.

BARTH, N. **Análise Comparativa De Ferramentas De Desenvolvimento De Aplicativos Móveis Multiplataforma**. 2014. 63f. Trabalho de Conclusão de Curso. (Graduação em Ciência Da Computação) - Universidade Regional De Blumenau, Blumenau, SC, 2014.

BEZERRA, P.; SCHIMIGUEL, J. **Desenvolvimento de aplicações mobile cross-platform utilizando phonegap**. Observatorio Economía Latinoamericana. v. 1, n. 1. Jan. 2016.

BIØRN-HANSEN, A. et al. **A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development**. Disponível em: <https://www.semanticscholar.org/paper/A-Survey-and-Taxonomy-of-Core-Concepts-and-Research-Bi%C3%B8rn-Hansen-

Gr%C3%B8nli/b9210947315fd7b7c85ca95430c2a7ec7da31aaf>. Acesso em: 08 jan. 2018.

BIØRN-HANSEN, A. et al. **An Empirical Study of Cross-Platform Mobile Development in Industry**. Wireless Communications and Mobile Computing (Hindawi). v. 2019, p. 12, jan. 2019.

CEVALLOS, E. A. **Case Study on Mobile Applications UX: Effect of the Usage of a Cross Platform Development Framework**. 2014. 101pg. Dissertação de Mestrado — UNIVERSIDAD POLITÉCNICA DE MADRID. Madrid. 2014. Disponível em: <http://oa.upm.es/30422/1/EMSE-2014-05_Esteban_Angulo-1.pdf>.

CHARLAND, A.; LEROUX, B. **Mobile Application Development: Web vs. Native**. Communications of the ACM, v. 54, n. 5, p. 49–53, 05/2011. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1941487.1941504>. Acesso em: 10 mar

Cordova. **Apache Cordova**. 2019. Disponível em: <https://cordova.apache.org/docs/en/latest/>.

CORRAL, L.; JANES, A.; REMENCIUS, T. **Potential Advantages and Disadvantages of Multiplatform Development Frameworks**: A Vision on Mobile Environments. Procedia Computer Science. v 10, p. 1202-1207, 2012.

EL-KASSAS, W. et al. **Taxonomy of Cross-Platform Mobile Applications Development Approaches**. Ain Shams Engineering Journal. v. 8, n. 2, p. 163-190, ISSN 2090-4479, June 2015.

Facebook Inc. **React Native**. 2019. Disponível em: <https://facebook.github.io/react-native/>.

HEITKÖTTER, H.; HANSCHKE S.; MAJCHRZAK T. **Evaluating Cross-Platform Development Approaches for Mobile Applications**. In: Web Information Systems and Technologies (WEBIST), 2012, p. 120–138. Porto.

HOELLER, E. **Estudo comparativo entre as plataformas .NET e Java Standard Edition com enfoque em desenvolvimento desktop**. 2010. Trabalho de Conclusão de Curso. (Graduação em Sistemas de Informação) - Universidade Federal de Santa Catarina, Florianópolis, SC, 2010.

HOLZINGER A., TREITLER P., SLANY W. **Making Apps Useable on Multiple Different Mobile Platforms**: On Interoperability for Business Application Development on Smartphones. In: Quirchmayr G., Basl J., You I., Xu L., Weippl E. (eds) Multidisciplinary Research and Practice for Information Systems. CD-ARES 2012. Lecture Notes in Computer Science, vol 7465. Springer, Berlin, Heidelberg.

JOBE, W. **Native Apps vs. Mobile Web Apps**. International Journal of Interactive Mobile Technologies (iJIM), v. 7, n. 4, p. 27-32, out. 2013. ISSN 1865-7923.

MALAVOLTA, I. **Web-based Hybrid Mobile Apps: State of the Practice and Research Opportunities**. Conference on Mobile Software Engineering and Systems (MOBILESoft), 16. mai. 2016.

STARK, J. **Building IPhone Apps with HTML, CSS, and JavaScript**. O'Reilly Media, Janeiro, 2010.

MENDONÇA, V.; BITTAR, T.; DIAS, M. **Um estudo dos Sistemas Operacionais Android e iOS para o desenvolvimento de aplicativos**. In: Encontro Anual de Computação (EnAComp), 9., 2011, Catalão: ISSN 2178-6992, 2011.

NAIR, H.; YADAV, V. **Study on Cross - Platform Mobile App Development With Xamarin**. International Journal of Trend in Scientific Research and Development (IJTSRD). v. 2, n. 4. P. 2554 – 2557, jun. 2018.

PREZOTTO, E.; BONIATI, B. **Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas**. In: Encontro Anual de Tecnologia da Informação (EATI), 5., 2014, Santa Maria: ISSN 2236-8604, 3025

STARK, J;. JEPSON, B. **Construindo Aplicativos Android com HTML, CSS e JavaScript**. O'Reilly Novatec, 2012.

STEIL, R. **iOS Programe para iPhone e iPad**. São Paulo: Casa do Código, 2015.

TUNALI, V.; ERDOGAN, S. **Comparison Of Popular Cross-platform Mobile Application Development Tools**. Ulusal Kongresi (YBS2015), 2. Out. 2015. Erzurum, Turquia.

ZEMEL, T. **Web Design Responsivo**: Páginas adaptáveis para todos os dispositivos. São Paulo: Casa do Código, 2012.