UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS**

SISTEMAS DE INFORMAÇÃO

ALINE CRISTINA DE MELO FONTOLAN

MODELO DE APOIO AO DESENVOLVIMENTO DE APLICATIVOS *MOBILE*

Bandeirantes

2018

ALINE CRISTINA DE MELO FONTOLAN

MODELO DE APOIO AO DESENVOLVIMENTO DE APLICATIVOS *MOBILE*

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Thiago Adriano Coleti

Bandeirantes

2018

ALINE CRISTINA DE MELO FONTOLAN

MODELO DE APOIO AO DESENVOLVIMENTO DE APLICATIVOS *MOBILE*

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____
Prof. Dr. Thiago Adriano Coleti
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. Wellington Della Mura
UENP – *Campus* Luiz Meneghel

_____
Profª. Drª. Daniela Guilhermino Trindade
UENP – *Campus* Luiz Meneghel

Bandeirantes, 04 de dezembro de 2018

RESUMO

Os dispositivos móveis como *smartphones*, *tablets* e semelhantes estão presentes no dia a dia das pessoas, seja para fins de trabalho, entretenimento, informação, comunicação e para muitos outros propósitos. Mas podemos imaginar que estes dispositivos, em curto prazo, estarão ainda mais presentes realizando todo tipo de tarefas simples ou complexas. Por conta disto, a computação móvel tem passado nos últimos anos por um salto impulsionado pela popularização destes dispositivos e por uma consequente demanda por aplicativos que estejam integrados à nossa maneira de agir e de interagir com as máquinas e com outras pessoas. Os desenvolvedores de *software* então, têm a tarefa de traduzir estas demandas em soluções que propiciem aos usuários uma experiência não somente satisfatória, mas excelente. Neste contexto, este trabalho apresenta alguns dos principais desafios enfrentados no desenvolvimento de aplicativos para as diversas plataformas móveis, face à grande diversidade de sistemas operacionais e dispositivos distintos existentes no mercado, bem como três modelos de fluxogramas como solução para minimizar as dificuldades na escolha de qual caminho de desenvolvimento seguir.

Palavras-chave: Desenvolvimento de aplicativos móveis. Dispositivos móveis. Tecnologias de desenvolvimento. Desenvolvimento nativo. Desenvolvimento Híbrido. Desenvolvimento *PWA.* Desenvolvimento *web app*.

ABSTRACT

Mobile devices such as smartphones, tablets and displays are present on people's day, whether for work, entertainment, information, communication purposes and for many other purposes. But, if you imagine that these devices are somehow, they may still exist more. Because of this, mobile computing has in recent years been driven by the popularization of devices and a consequent demand for applications that become our way of acting and interacting with machines and with other people. Software developers, then, have a task of translating their needs into solutions that are convenient, but not satisfactory, but excellent. In this context, this paper presents some of the main challenges faced in the development of applications for several mobile platforms, due to the wide variety of different operating systems and devices in the market, as a main flowchart for solving problems in the choice of development path qualifications following.

Keywords: *Mobile application development. Mobile devices. Development technologies. Native development. Hybrid Development. PWA Development. Web Development app.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

APPS	Aplicativos
SPA	*Single-page Application*
SO	Sistema Operacional
SW	*Service Worker*
IDE	*Integrated Development Environment*
SDK	*Software Development Kit*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
JS	JavaScript
iOS	*iPhone operating system*
API	*Application Programming Interface*
GPS	*Global Positioning System*
OHA	*Open Handset Alliance*
UI	*User Interface*
IPC	*Inter-Process Comunication*
HAL	*Hardware Abstraction Layer*
PWA	*Progressive Web App*
PMI	*Project Management Institute*

SUMÁRIO

1. INTRODUÇÃO

Este capítulo descreve o contexto, o problema de pesquisa, os objetivos e a metodologia utilizada neste trabalho.

1.1. CONTEXTO E DELIMITAÇÃO DO TRABALHO

A esfera dos aplicativos para dispositivos móveis cresce rapidamente a cada dia, tornando-se parte do cotidiano das pessoas. Tanto nas ruas quanto em estabelecimentos em geral, é possível observar indivíduos concentrados em seus celulares, seja para navegar na *web*, em redes sociais, realizar tarefas, jogar ou resolver assuntos de trabalho (LINGRAS, 2016).

Lopes (2016), afirma que os aplicativos móveis se tornaram essenciais e responsáveis pelas interações digitais, já que atualmente é possível cumprir diversas tarefas com o uso destes.

Os *smartphones* transformaram a maneira de como a tecnologia é enfrentada, otimizaram o tempo pessoal e transformaram os hábitos de comunicação e consumo da sociedade. Tudo isso, em grande parte, é graças aos aplicativos desenvolvidos com diferentes abordagens, sendo nativa, híbrida, *web* e *Progressive Web App* (PWA) as mais conhecidas. Esses métodos refizeram os dispositivos móveis e originaram diversos cenários para este aparelho que cabe na palma de uma mão (PREZOTTO; BONIATI, 2014).

Dados alcançados pelas empresas *Hootsuite* e *We are social* mostrados na Figura 1, exemplificam o crescimento dos *smartphones*. Nota-se que são mais de cinco bilhões de usuários que manipulam algum modelo de celular, sendo que este número representa 68% da população mundial. Além disso, mais da metade destes, são usuários de *smartphones* e agentes de 50% do tráfego na *web* (KEMP SIMON, 2018).

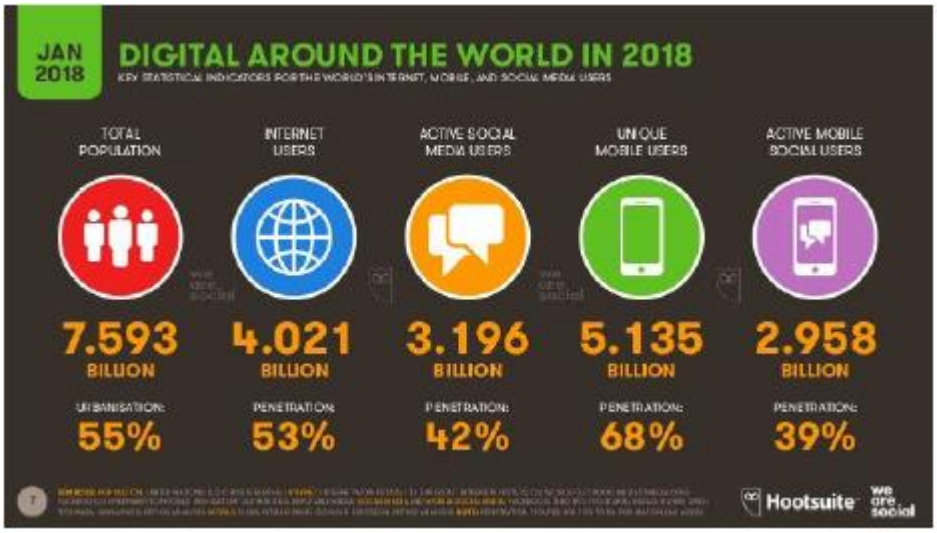

FIGURA 1 - CENÁRIO GLOBAL DIGITAL EM 2018. FONTE: HOOTSUITE E WE ARE SOCIAL.

A empresa norte-americana *App Annie* atua no mercado de aplicativos e conduziu uma pesquisa no final do primeiro semestre de 2018 apontando que mais de 6 milhões de aplicativos para *smartphones* estão disponíveis nas principais lojas de aplicativos, *App Store* e *Google Play.* A mesma pesquisa também indica que os usuários de dispositivos móveis passam em média três horas por dia em aplicativos (SYDOW LEXI, 2018).

A partir das pesquisas apresentadas é possível concluir que uma análise voltada ao desenvolvimento de aplicativos móveis pode ser bastante significativa, pois trata-se de um setor em constante expansão. Além disso, os aplicativos oferecem funcionalidades atualmente necessárias no dia-a-dia das pessoas.

Diante do contexto da grande diversidade de dispositivos móveis, sistemas operacionais, como *IOS* e *Android*, e as várias formas de desenvolvimento como aplicações nativas, híbridas, *web app* e *PWA*, tornou-se fundamental decidir qual abordagem tecnológica de desenvolvimento é a melhor para a construção de uma aplicação móvel.

Para a tomada de decisão entre as diferentes técnicas de desenvolvimento abordadas neste trabalho, deve-se considerar o cenário de cada aplicação, entender como funciona seu método de desenvolvimento e quais as vantagens de cada tecnologia.

Neste trabalho é proposto o desenvolvimento de um modelo de fluxograma para auxiliar a tomada de decisão de profissionais desenvolvedores no momento da escolha entre as tecnologias de desenvolvimento de um aplicativo. Este modelo de fluxo terá como base fatores técnicos que devem ser considerados pelos desenvolvedores na tomada de decisão.

1.2. FORMULAÇÃO DO PROBLEMA

Um aplicativo é executado em diversos dispositivos e sistemas operacionais diferentes. Com isso, a principal dificuldade no desenvolvimento de aplicativos *mobile* é a não uniformidade das plataformas móveis, pois o método de desenvolvimento de um aplicativo é diferente para cada plataforma (MARTINS et al., 2013).

Os aplicativos nativos, *web*, *PWA* e híbridos são as principais soluções para essa falta de uniformidade no desenvolvimento de aplicações móveis. Contudo, ainda há uma fragmentação de soluções que torna a escolha de uma dessas alternativas desafiadora para os desenvolvedores (FLING, 2009).

Diante de um cenário de escolha, convém analisar as características das plataformas de desenvolvimento móvel, as diferenças entre essas abordagens estão relacionadas à diversos aspectos, como o acesso aos recursos do dispositivo, conhecimento da equipe de desenvolvimento, suporte da comunidade, facilidade de aprendizado, experiência de usuário, distribuição, custo, desempenho, entre outros.

Todas as abordagens de desenvolvimento citadas possuem benefícios e limitações em relação a estes requisitos. Este trabalho pretende auxiliar desenvolvedores a identificar a melhor estratégia de desenvolvimento móvel com base nas necessidades de um projeto, tratando do problema causado por este vasto cenário de escolha.

1.3. OBJETIVOS

1.3.1. OBJETIVO GERAL

O objetivo principal é elaborar um modelo de fluxograma que auxiliará a equipe desenvolvedora de aplicativos móveis para tomar decisões relacionadas à tecnologia adotada, analisando estrategicamente um caminho que minimize as dificuldades diante da não uniformidade encontrada atualmente entre as plataformas e dispositivos móveis existentes no mercado.

1.3.2. OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral, os seguintes objetivos específicos devem ser alcançados:

- Um levantamento bibliográfico para identificar características e problemas dos ambientes de desenvolvimento nativo, híbrido, *web* e *PWA*.
- Identificar os principais critérios utilizados no processo de abordagens nativas, híbridas, *web* e *PWA*.
- Investigar vantagens e desvantagens dos ambientes de desenvolvimento *mobile*.
- Comparar os ambientes de desenvolvimento *mobile* com base nas informações coletadas no levantamento bibliográfico.
- Definir e elaborar um modelo de decisão a ser utilizado por desenvolvedores para escolher entre as abordagens nativa, híbrida, *web* e *PWA* com base nas necessidades de seus projetos e equipe.

1.4. JUSTIFICATIVA

Cada plataforma possui seus dispositivos, linguagens de programação, *kits* de desenvolvimento e mercados de distribuição distintos com uma quantidade significativa de usuários que representam os clientes potenciais (FLING, 2009).

Um desafio importante para os desenvolvedores é selecionar a plataforma para desenvolverem suas aplicações móveis sem descartar nenhum grupo de usuários de um aplicativo.

Com isso, pode-se afirmar que este trabalho é relevante pois realizará uma análise de cada abordagem com o objetivo de determinar qual é a mais adequada para determinado cenário. Além de possibilitar apoio ao um desenvolvimento de aplicativos móveis, considerando o aspecto das necessidades do time de desenvolvedores e também a utilização por parte dos usuários finais desta aplicação.

1.5. METODOLOGIA DE DESENVOLVIMENTO

O presente trabalho caracteriza-se como uma pesquisa exploratória, que de acordo com Gil (2002), tem por objetivo tornar explícita a visão do problema, através de uma pesquisa bibliográfica, seguida de análises das plataformas e de tecnologias existentes no desenvolvimento de aplicativos móveis. Com isso, será possível construir hipóteses de apoio aos desenvolvedores *mobile.*

Segundo Wazlawick (2014), como objetivos da pesquisa descritiva espera-se obter dados mais consistentes sobre determinada realidade, contudo, não há interferência do pesquisador ou tentativa de obter teorias que expliquem os fenômenos. Tenta-se apenas descrever os fatos como são.

Este trabalho é classificado também como uma pesquisa bibliográfica qualitativa, pois baseia-se no julgamento humano. A pesquisa bibliográfica implica o estudo de artigos, teses, livros e outras publicações usualmente disponibilizadas por editoras e bases de dados. Esse é um passo inicial fundamental para qualquer trabalho científico, sendo que

ela em si não produz qualquer conhecimento novo, apenas supre o pesquisador de informações públicas que ele ainda não possuía (WAZLAWICK, 2014).

Esta classificação deve-se ao fato de que a ideia é compreender as questões relacionadas aos métodos de desenvolvimento *mobile* por meio do estudo de trabalhos relevantes que abordam este tema (GIL, 2002).

Para o desenvolvimento deste trabalho serão necessários os seguintes passos:

a) Fundamentação Teórica: analisar os estudos literários já existentes sobre aplicativos nativos, híbridos, *web* e *PWA* para as plataformas existentes no mercado atual, com foco nas diferenças entre cada abordagem.

b) Estudo comparativo entre as tecnologias de desenvolvimento: analisar as vantagens e desvantagens de cada tipo de desenvolvimento para explorar qual é mais apropriado baseado na análise de alguns aspectos técnicos para criação do aplicativo.

c) Elaboração de um modelo de decisão para os desenvolvedores *mobile*: este modelo será embasado a partir da definição do contexto, problema e objetivo. Além disso, serão elencados e analisados alguns critérios técnicos utilizados no desenvolvimento de aplicativos.

   Através da abordagem Goal/Question/Metric definida por Basili et al. (1994) serão elaboradas questões de análise técnica aprofundadas para cada critério e serão definidas suas respectivas métricas para apoiar a tomada de decisão.

   Mesmo sendo proposto em 1994, esse modelo ainda é utilizado para elaboração de métricas em projetos de software. Além disso, a escolha dessa abordagem se dá pela particularidade de conseguir criar questões e métricas a partir de objetivos, e não de resultados (VAN SOLINGEN, 2014).

   Desta forma o modelo de decisão será construído num fluxo linear criado na ferramenta *Bizagi* para ilustrar os possíveis caminhos de tomada de decisão.

d) Conclusão da aplicabilidade da abordagem decisória.

1.6. ORGANIZAÇÃO DO TRABALHO

Neste capítulo encontra-se a introdução do trabalho, problema, justificativa, apresentação dos objetivos e da estrutura do documento. O restante do documento está estruturado da seguinte maneira:

- Capítulo 2: Fundamentação Teórica. Neste Capítulo, é feito um estudo sobre como se dá o desenvolvimento de aplicativos *mobile.* Além disso, foi destacado vantagens e desvantagens das quatro abordagens de desenvolvimento trazidas pela literatura.
- Capítulo 3: Desenvolvimento. Neste Capítulo, é elencado algumas funcionalidades comuns que estão presentes nos aplicativos móveis desenvolvidos por diferentes abordagens e sequencialmente é feita a comparação das funcionalidades entre os tipos de desenvolvimento *mobile*. Com resultado destas comparações é construído um modelo de fluxograma que servirá de apoio a tomada de decisão dos desenvolvedores.
- Capítulo 4: Conclusão e Trabalho Futuros. Por fim, este Capítulo, apresentam-se a resposta da questão problema, a conclusão do trabalho, possíveis limitações e trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar os principais conceitos dos temas relacionados com este trabalho.

2.1. APLICAÇÕES MÓVEIS

Segundo Jobs (2007), durante a conferência *World Wide Developers* da *Apple*, na Califórnia citou que “[...] de vez em quando surge um produto revolucionário que muda tudo”, assim foi a chegada da Internet marcando tal transformação no mundo da computação e, consequentemente, na indústria de *software*. Fling (2009) em sua descrição evolucionária dos celulares citou a era dos recursos como o momento que os aparelhos móveis adquiriram vários tipos de aplicações e serviços, inclusive o acesso à internet em um telefone, de onde derivou-se o nome *smartphone*.

Estes importantes pensamentos concluem que a inclusão da Internet nos dispositivos móveis e o desenvolvimento da computação móvel proporcionaram aos usuários novas experiências e praticidade nos serviços e acesso à informação.

Segundo Lopes (2013), os aplicativos móveis são *softwares* que desempenham objetivos específicos em *smartphones, tablets e wearables.* Desde 2007, os aplicativos estão cada vez mais populares e importantes na vida dos usuários, pois facilitam e simplificam atividades do usuário, além de favorecer uma comunicação ampla, com acesso às informações sem barreiras de conexão.

Ao projetar e desenvolver aplicações móveis deve-se pensar que este *software* solucionará problemas de um determinado ambiente usando os recursos disponíveis no celular, por isso, os autores (LOPES, 2013; FLING, 2009) definem os seguintes critérios para a elaboração e o desenvolvimento dessas aplicações:

- Definir o público alvo, esta etapa caracteriza os usuários finais que utilizarão o aplicativo.
- Definir a segmentação a qual o aplicativo será utilizado, esta etapa desenvolve uma comunicação e compreensão entre o usuário e o aplicativo.

- Definir o escopo da aplicação documentando os objetivos, quais e quantas serão as funcionalidades.
- Definir para quais sistemas operacionais a aplicação estará disponível.
- Definir qual a plataforma de desenvolvimento utilizar.

A união de todas as etapas e uma posterior análise, apoiará os desenvolvedores a tomar a decisão de qual a abordagem mais adequada para desenvolver a aplicação, seja ele nativo, híbrida, *web* ou *PWA*.

Para tomar essa decisão, os desenvolvedores devem considerar aspectos como gestos disponíveis, tamanhos de tela, poder de processamento e capacidade de memória ao desenvolver uma aplicação móvel (VISWANATHAN, 2018). Além disso, fatores relacionados ao time de desenvolvedores como conhecimento prévio, tempo, custo e recursos disponíveis nos dispositivos também são fatores relevantes.

Com base nos requisitos e aspectos citados nesta seção, são apresentadas soluções para o desenvolvimento de aplicativos móveis. Tais soluções podem ser divididas em quatro categorias: nativas, híbridas, *PWA* e web *mobile*.

2.2. APLICAÇÕES NATIVAS

Atualmente existem muitas plataformas para desenvolvimento móvel nativo, como *Android, iOS, Windows Phone, Firefox OS, BlackBerry, Ubuntu Touch, Fire OS,* entre outros. Cada plataforma possui sua própria linguagem de programação e *interface* de programação de aplicativos (*API*) nativas para desenvolvimento, o que dificulta o trabalho dos desenvolvedores. As *API's* ajudam os diferentes aplicativos a interagirem entre si, acessarem os recursos nativos da plataforma e criar *interfaces* do usuário (*UI's*) seguindo os padrões de determinada plataforma (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

Existem duas principais categorias de sistemas operacionais (SO) que atualmente dominam o mercado dos dispositivos móveis que são *Android*, da *Google* e o *iOS*, da *Apple*. Dessa forma, as escolhas de desenvolvimento de uma aplicação nativa são

orientadas com base na escolha de uma plataforma na qual o aplicativo funcionará (VISWANATHAN, 2018).

O SO de uma plataforma é o responsável por gerenciar os recursos e funcionalidades do dispositivo móvel como sensores, Sistema de Posicionamento Global (GPS), câmera, calendário, lista de contatos, envio de mensagens, entre outros (SILVA; SANTOS, 2014).

Além disso, cada tipo de SO possui seu próprio padrão de *design* de *interface*, exemplificados na Figura 2, nota-se que existem diferenças entre a *interface* gráfica da *Apple*[1] e a *interface* gráfica do *Google*[2]. Esses padrões devem ser respeitados para proporcionar ótima experiência aos usuários finais do aplicativo.

FIGURA 2 - INTERFACE ANDROID (TELA INICIAL) E INTERFACE IOS (TELA INICIAL).

Para desenvolver um aplicativo nativo é preciso ter conhecimentos específicos sobre as tecnologias utilizadas pela plataforma em questão. Caso ocorra a necessidade de

[1] https://developer.apple.com/design/human-interface-guidelines/
[2] https://developer.android.com/guide/practices/ui_guidelines/

implementar para mais de uma plataforma, a aplicação deverá ser desenvolvida separadamente para cada uma (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

A Tabela 1 apresenta um resumo das linguagens de programação e ferramentas necessárias para desenvolver aplicativos nativos nas plataformas *iOS* e *Android.*

TABELA 1 - CONHECIMENTOS FUNDAMENTAIS DOS SISTEMAS OPERACIONAIS

PLATAFORMA	LINGUAGEM DE PROGRAMAÇÃO	FERRAMENTAS
Google Android	Java, SQL e XML	Android Studio, SDK Android
Apple iOS	Objective-C e/ou Swift	Xcode, SDK iOS

Fonte: Adaptado de (CHARLAND e LEROUX, 2011).

2.2.1. *ANDROID*

Android é um SO proposto pela empresa *Android Inc.* sediada nos Estados Unidos. Em julho de 2005, a empresa foi adquirida pelo *Google*. Essa plataforma tem como base o SO *Linux*, além de possuir diferentes bibliotecas e ferramentas para desenvolver aplicativos (HUBSCH, 2012).

A plataforma *Android* é, sem dúvida, muito utilizado não apenas em *smartphones*, mas também em diversos dispositivos que apresentam um microprocessador como câmeras, relógios, *tablets*, entre outros (ANDROID, 2017).

Segundo dados da *International Data Corporation* (IDC), o *Android* é considerado o SO móvel mais popular do mundo e apresenta rápido crescimento e evolução de melhorias desde sua criação.

A plataforma é aberta para os desenvolvedores, onde qualquer fabricante pode obter a licença e fazer suas próprias alterações e atualizações. Por isso, o *Android* é considerado um sistema flexível e adaptável. Este SO possui várias versões que, ao longo do tempo, foram evoluindo e sendo atualizadas com mais recursos disponíveis à favor da mobilidade para os usuários (PAPAJORGJI, 2015).

Os principais objetivos do *Android* são criar uma plataforma de *software* aberta disponível para operadoras, *Original Equipment Manufacturer* (OEMs) e desenvolvedores para tornar suas ideias inovadoras, reais e apresentar um produto bem-sucedido que melhore a experiência móvel dos usuários (ANDROID, 2017).

A Figura 3 apresenta as camadas da arquitetura *Android* que serão explicadas com detalhes.

FIGURA 3 - ARQUITETURA ANDROID. FONTE: ANDROID, 2017.

Cada camada é detalhada a seguir com base nos conceitos retirados do *site* de desenvolvedores que a plataforma *Android* oferece (ANDROID, 2017). As camadas são descritas na ordem de cima para baixo ilustradas na Figura 3.

- ***Application Framework***: nessa camada as aplicações são construídas e instaladas pelos desenvolvedores.

- ***Binder IPC Proxies***: *IPC* significa *Inter-Process Communication*. É a camada que faz uma ponte entre as aplicações instaladas na camada superior com a próxima camada, a *System Services*. É uma *interface* que permite que *API's* de alto nível interajam com serviços do sistema *Android*.
- ***System Services***: por meio de módulos, permite acesso a *hardwares* específicos. Cada serviço desta camada foi desenvolvido para gerenciar um componente específico, como busca e notificações. Os serviços foram divididos em duas categorias, *Medi*a e *System*, explicadas a seguir.
    - *Media Server*: são os serviços responsáveis por gerenciar conteúdos de mídia como gravação e reprodução de áudio e vídeo.
    - *System Server*: são serviços responsáveis por gerenciar os demais tipos de serviço do sistema como notificações e janelas.
- ***HAL****:* permite que fornecedores de *hardware* criem suas *interfaces* e *drivers*. Com isso, é possível criar novas funcionalidades e implementá-las sem afetar o resto das camadas do sistema.
- ***Linux Kernel****:* é uma versão do *kernel* do *Linux* com algumas modificações, como o gerenciamento de memória mais avançado e próprio para dispositivos móveis e funcionalidades para dispositivos embarcados.

Os aplicativos criados precisam ser publicados no *Google Play*, que é a principal loja de distribuição de aplicativos para *Android*. Os desenvolvedores devem seguir as regras descritas abaixo para a publicação de suas aplicações:

- Registro de uma conta de editor do *Google Play.*
- Configurar uma conta de comerciante do *Google Payments* para vender aplicativos ou produtos dentro dos aplicativos.
- Explorar o *Developer Console* do *Google Play* e as ferramentas de publicação.
- Pagar uma taxa de registro de US$ 25 usando o *Google Payments*.

Porém, antes de serem publicadas, as aplicações passam por processo de revisão pelo *Google* para assegurar que estas não violam as políticas e regras estabelecidas (ANDROID, 2017).

Meier (2015) aponta alguns dos principais desafios de desenvolver para *Android* como (i) não há garantia que a aplicação funcionará perfeitamente em qualquer dispositivo, pois podem ocorrer problemas de compatibilidade já que o *hardware* de cada aparelho móvel é diferente; e (ii) incompatibilidade das versões do *SO Android*, pois nem todos os dispositivos suportam todas as versões.

2.2.2. IOS

O SO *iOS* foi publicado simultaneamente com o lançamento do primeiro *iPhone* em 2007. Esse SO trabalha como uma *interface* entre os aplicativos para *iOS* e o *hardware* de dispositivos móveis da *Apple* como o *iPhone, iPad e iPod*. (APPLE INC., 2017).

A *Apple* desenvolve tanto o SO quanto o *hardware* de seus dispositivos, o que permite que todos funcionem em conjunto. Desta forma, as aplicações em *iOS* aproveitam ao máximo seus recursos nativos como: gráficos, processadores, sensores, entre outros (SILVA; SANTOS, 2014).

O *iOS* tem como um de seus principais objetivos a interação direta do usuário final com o SO (REBOUAS et. al, 2016). Por este motivo, a *Apple* é conhecida no mercado da computação móvel pela preocupação com a experiência do usuário, tornando fácil e intuitivo o uso de seus sistemas e dispositivos (HUBSCH, 2012).

O desenvolvimento para a plataforma *iOS* tem como base duas linguagens de programação, a *Swift* e a *Objective-C*. A linguagem *Swift* foi lançada pela *Apple Inc*. como uma sucessora da linguagem *Objective-C*, e é uma linguagem de *software* livre, concisa e intuitiva. Além disso, pode ser adicionada com a facilidade a um código-fonte já existente em *Objective-C* sem a necessidade de uma readaptação (APPLE INC, 2017).

Para o desenvolvimento e teste de aplicativos para a plataforma *iOS*, é necessário ter um computador com o SO *Mac OS*, onde o *SDK iOS* e o ambiente de desenvolvimento *Xcode* estão instalados (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

A *Apple* aconselha a utilização das camadas de alto nível da arquitetura para facilidade no desenvolvimento. Estas camadas possuem abstrações orientadas à objetos relacionadas à funcionalidades encontradas em camadas de baixo nível da arquitetura.

Essas funcionalidades são encapsuladas por meio de *interfaces*, as quais são disponibilizadas pelo *framework* que contém *DSL* (*Domain-Specific Languages*) e as funcionalidades necessárias para o funcionamento adequado da ferramenta (APPLE INC., 2017).

A Figura 4 detalha a diferença entre as camadas que compõe a arquitetura iOS.

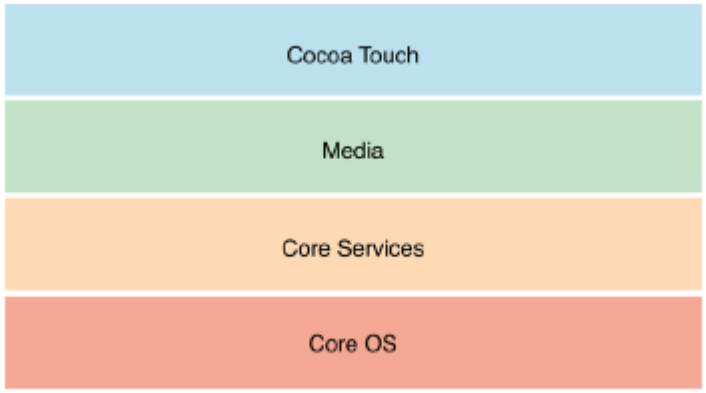

FIGURA 4 - ARQUITETURA DO SO IOS. FONTE: APPLE INC., 2017.

A arquitetura iOS, apresentada na Figura 4, é formada por quatro camadas, sendo que em cada camada existe um conjunto de funcionalidades que podem ser utilizados durante o desenvolvimento de *apps*. As camadas do *iOS* são explicadas com mais detalhes a seguir (Apple Inc., 2017):

- ***Cocoa Touch*:** a camada mais alto nível onde são fornecidos serviços básicos de interação com o usuário como entrada baseada em toques, notificações *push* e outras tecnologias necessárias para melhorar a experiência do usuário. Essa camada também possui *frameworks* de alto nível, os quais permitem acesso a funcionalidades do sistema como lista de contatos, eventos do calendário e mapas.
- ***Media*:** contém tecnologias e *frameworks* necessários para a implementação de experiências multimídia com áudio, vídeo e gráficos.
- ***Core Services***: essa camada fica mais próxima do *hardware*, portanto, possui acesso a funcionalidades de baixo nível como localização, telefonia, *threads* e banco de dados. Dois dos *frameworks* mais importantes do *iOS*, o

Foundation e o *Core Foundation*, estão presentes nessa camada. Sendo que ambos estão relacionados ao gerenciamento de dados e serviços, além de definir os tipos básicos de dados utilizados pelos *apps*, como *strings*, data e hora, *sockets* e *threads*.

- ***Core OS*:** camada onde as funcionalidades de baixo nível são construídas e utilizadas por *frameworks* de outras camadas. Se a aplicação possui requisitos de segurança ou comunicação com acessórios externos complexos, pode-se utilizar as funcionalidades dessa camada.

A distribuição dos aplicativos *iOS* é feito na *Apple Store*, a loja de aplicativos d*a Apple* e é necessário ter uma licença de *Apple Developer Program*, que possui taxa anual de US$99,00 dólares. Além da licença exigida, também é preciso enviar a aplicação desenvolvida para análise e avaliação, onde é checado se o aplicativo segue as diretrizes estabelecidas para posteriormente ficarem disponíveis na loja (APPLE INC., 2017).

2.2.3. VANTAGENS DE APLICAÇÕES NATIVAS

Aplicações nativas proporcionam uma boa usabilidade ao usuário, pois além de seguir padrões técnicos de *interface* e de experiência de usuário, a aplicação é desenvolvida em seu próprio ambiente (WHITE, 2013).

Além disso, vários autores afirmam que essas aplicações oferecem ao usuário uma navegação mais completa pois aproveitam por completo os recursos nativos oferecidos pelo *smartphone,* permitindo acesso à recursos como lista de contatos, câmera, sensores, entre outros (EL-KASSAS et al., 2015, CHARLAND; LEROUX, 2011).

Na mesma linha de raciocínio, o desenvolvedor Kinjal Dua (2018) afirma que aplicações nativas possuem melhor desempenho e melhor experiência ao usuário em relação às outras plataformas de desenvolvimento *mobile*. Isso acontece pois são criadas para determinada plataforma, tornando o fluxo de navegação natural.

Contudo, estudos recentes apontam que aplicações nativas, web *mobile* e *PWA's* podem proporcionar uma experiência de usuário similar, sendo satisfatória com as três abordagens (CARDIERI; ZAINA, 2018).

Os autores Heitkotter, Hanschke e Majchrzak (2013) afirmam que aplicações nativas são a opção ideal em alguns cenários específicos, como quando (i) a aplicação exige desempenho crítico, onde com um atraso na mudança de tela o usuário abandonaria a aplicação; (ii) a aplicação exige frequente acesso às *API's* nativas e/ou sensores, como câmera, geolocalização e notificações *push*; e, (iii) é necessária integração com *hardware* externo, como *smartwatches* e *wearables.*

Além das vantagens previamente citadas, ainda podemos citar que aplicações nativas possuem menor custo de acesso, pois parte da *interface* do usuário é previamente instalada no celular, possibilitando um menor tráfego de dados ao acessar um conteúdo *online* (CHARLAND; LEROUX 2011).

2.2.4. DESVANTAGENS DE APLICAÇÕES NATIVAS

É possível observar um grande número de vantagens no uso e desenvolvimento de aplicações nativas, contudo, também pode-se observar variadas desvantagens.

Dua (2018) ressalta que desenvolvimento nativo não é a melhor opção para aplicações simples devido ao grande consumo de memória para instalar o aplicativo e da necessidade de atualizações com frequência de forma manual. A necessidade de atualizar a aplicação também é citada como desvantagem por Charland & Leroux (2011), pois o usuário precisará efetuar uma nova instalação de versão a cada alteração na estrutura ou conteúdo feita pelo time de desenvolvedores.

De acordo com White (2013), a principal desvantagem de uma aplicação nativa é que esta só poderá ser executada na plataforma para a qual foi desenvolvida. Esse fator gera um aumento de custos e esforços para disponibilizar o *app* nos diversos SO existentes.

Ainda relacionado a esse tema, os autores Charland & Leroux (2011) apontam que desenvolver para diferentes SO requer uma equipe com conhecimentos em tecnologias distintas, já que cada plataforma possui sua própria ferramenta de desenvolvimento e *API's* que garantem a consistência dos aplicativos.

Aplicações nativas também sofrem limitações relacionadas ao suporte nativo do dispositivo móvel utilizado. Smartphones apresentam recursos de *software* e *hardware* diferentes, o que pode causar a necessidade de existir versões diferentes do mesmo aplicativo para diferentes versões de um mesmo SO (CHARLAND; LEROUX 2011).

Financeiramente os *apps* nativos possuem uma barreira extra e podem ser um investimento de alto risco. Seu controle não é de autoria exclusiva da empresa desenvolvedora, pois a intermediação da loja de aplicativos disponibilizada pelo fabricante da tecnologia é necessária. Com isso, o retorno financeiro é dividido entre quem desenvolveu e a empresa que possui a loja de aplicações.

Desta forma, não é possível ter controle de como a disponibilização do aplicativo será feita, por exemplo, permitir um maior destaque numa loja com um grande número de aplicações similares (FLING, 2009).

Por isso, é importante que os envolvidos no desenvolvimento da aplicação possuam conhecimento dos requisitos de seu projeto e, a partir disso, possam escolher a abordagem mais adequada para o desenvolvimento de sua aplicação.

Construir um aplicativo nativo pode ser uma solução custosa para algumas empresas, sendo que para alguns produtos não faz sentido ter um aplicativo desenvolvido nativamente.

2.3. APLICAÇÕES *WEB MOBILE*

A existência de diferentes tipos de SO, dispositivos móveis com diversos tamanhos e dimensões de telas motivaram o desenvolvimento das aplicações *web mobile*. Uma aplicação *web mobile*, também chamado de *web app*, usa a mesma tecnologia de um site com *web design* responsivo (WDR).

O WDR permite que uma aplicação se adapte a qualquer dimensão e tamanho de tela que o usuário esteja visualizando em seu dispositivo. Normalmente, os *web apps* são utilizados com frequência para apresentar informações em formato de lista, o que facilita a navegação do usuário (SMUTNY, 2012).

Um *web app* existe no navegador de um dispositivo móvel, o que o torna facilmente encontrado por meio de um mecanismo de pesquisa, afinal, sua base é um *website*. Esse tipo de aplicação não tem necessidade de ser instalada, e suas tecnologias base de desenvolvimento, segundo *W3C*[3], são *HTML*, *CSS* e *JavaScript* (JS). Por isso, a execução de *web apps* ocorre por completo no navegador do dispositivo móvel.

O SO não tem controle sobre seu conteúdo e funções, sendo que alterações feitas pelos desenvolvedores resultam numa atualização imediata aos usuários (WHITE, 2013).

Lopes (2013), afirma que a existência de aplicativos *web* no navegador não permite que eles funcionem sem uma conexão com a *internet*, ou aproveitem o máximo potencial dos recursos de um dispositivo móvel.

2.3.1. VANTAGENS DE UMA APLICAÇÃO *WEB MOBILE*

Aplicações *web* possuem vantagens tanto sobre aplicações nativas quanto híbridas. Apesar de terem custos relacionados a uma possível licença comercial e relacionados a hospedagem do *web app* em um servidor *online*, esse tipo de aplicação costuma custar menos que suas concorrentes. Além do custo, o tempo de desenvolvimento de uma aplicação *web* também é menor, pois o mesmo código é utilizado em diferentes SO, e todos tem seu próprio navegador.

Ao optar por aplicações *web*, os desenvolvedores ainda têm a oportunidade de migrar com facilidade para uma aplicação híbrida ou uma *PWA* (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

Fling (2009), destaca que aplicações *web* possuem vantagens devido sua portabilidade e maturidade. O autor descreve que com a evolução dos navegadores *web* móveis os *web apps* estão tornando-se mais avançadas, com funcionalidades

[3] O Consórcio World Wide Web (W3C) é um consórcio internacional no qual organizações filiadas, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões para a Web. [Informação obtida em: <http://www.w3c.br/Padroes/WebDesignAplicacoes> Acesso em: 02 dezembro 2017].

relacionadas a geolocalização, uso de acelerômetro e entre outros meios de utilização de *hardware*.

Esse avanço permite que, no futuro, aplicações nativas sejam reduzidas a contextos específicos que requerem toda a capacidade do dispositivo. Além disso, com a atual fragmentação no mercado de *smartphones,* não existem meios economicamente baratos de desenvolver *apps* nativos que abranjam uma grande parte de mercado. A plataforma móvel *web* é a única que funciona independente do dispositivo, além de ter o aprendizado mais rápido, menores custos de produção, mais padrões estabelecidos, maior disponibilidade para utilização e fácil distribuição.

De acordo com Lopes (2013), aplicações *web mobile* possuem um *design* mais relacionado a identidade visual da empresa que representa, o que pode ser considerado um ponto positivo. Diferente de aplicações nativas, que possuem seu *design* vinculado a um SO como *iOS* ou *Android*.

Na perspectiva do usuário, uma aplicação *web* é mais facilmente compartilhada, pois só é necessário saber sua *Uniform Resource Locator* (*URL)*. Em aplicações nativas, é necessário checar se o *app* está disponível para a plataforma selecionada, e se estiver, realizar uma busca da loja de aplicativos (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013)

Por fim, o desenvolvimento de *web apps* possibilitam uma abrangência multiplataforma aliado a redução do custo de desenvolvimento, ao passo que os desenvolvimentos de aplicações nativas possibilitam um melhor aproveitamento dos recursos de *hardware* (ATER, 2017).

2.3.2. DESVANTAGENS DE UMA APLICAÇÃO *WEB MOBILE*

Apesar de vantajosas por vários motivos, aplicações *web mobile* também possuem desvantagens. O desenvolvimento com tecnologias *HTML, CSS e JS* não permite acesso a todos os elementos de *hardware* de um *smartphone* (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

Websites como o *"What Web Can Do Today"*[4] apresentam exatamente quais são essas limitações, que estão ligadas ao navegador de cada dispositivo móvel. Além disso, aplicações *web* são dependentes da *internet,* o que implica em um lento carregamento inicial em alguns casos.

Fling (2009), aponta que para obter uma experiência completa ao usuário é preciso desenvolver os aplicativos de forma nativa. A abordagem *PWA,* descrita na Subseção 2.5, complementa a experiência de usuário de uma aplicação *web* (PETELE, 2017).

Ainda relacionado a experiência do usuário, aplicações *web* não tem acesso a elementos nativos da *interface* do usuário dentro do navegador (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

Uma de suas principais desvantagens é a dependência da conexão com a *internet*. Diferente de aplicações *PWA*, nativas e híbridas, um *web app* precisa de uma conexão com a internet até para seu carregamento inicial (LOPES, 2013).

2.4. APLICAÇÕES HÍBRIDAS

Segundo Ribeiro e Freire (2013), o desenvolvimento híbrido é definido como um conjunto de arquivos de código-fonte, bibliotecas e ferramentas que oferecem suporte a mais de uma plataforma.

O desenvolvimento multiplataforma foi a solução encontrada para suprir a demanda dos clientes, pois possibilitam a implementação de um único *app* em diferentes SO (SILVA; SANTOS, 2014). Segundo Palmieri et. Al. (2012), o desenvolvimento híbrido trouxe benefícios como a redução do tempo na implementação, redução de custo na formação de equipes de desenvolvedores, redução da complexidade de desenvolvimento e maior facilidade para desenvolver.

Esse método de desenvolvimento é composto por uma variação de abordagens. Por um lado, existem os aplicativos híbridos que possuem a combinação das tecnologias *web* com viabilidade para acessar funcionalidades nativas através de bibliotecas e *plugins*

[4] https://whatwebcando.today

auxiliares. Os *plugins* são ferramentas que se encaixam ao programa desenvolvido com objetivo de adicionar recursos e funções extras, além disso, são ligados aos diferentes SO por meio de *API's* (SILVA; SANTOS, 2014).

Cavalcanti (2016), descreve a tecnologia híbrida como um desenvolvimento interpretado e não compilado, utilizando uma *webview*, ou seja, um *browser* modificado para exibir o conteúdo renderizado sem as ferramentas e controles de seu *design* padrão. Dentro da *webview,* a linguagem JS constrói toda a aplicação e a comunicação com o contexto nativo é feita via *plugins*.

2.4.1. FERRAMENTAS PARA DESENVOLVIMENTO HÍBRIDO

A ferramenta *Xamarin* permite o desenvolvimento de aplicações multiplataforma com a linguagem de programação *C#*, a qual realiza chamadas nativas no SO dos dispositivos. Nessa ferramenta, o código escrito permite o uso de componentes e chamadas nativas de *API's* para acessarem os recursos de *hardware* oferecidos pelo dispositivo.

Diferente de outras tecnologias, não é necessária a criação de aplicações *web* que serão empacotadas como aplicações nativas. O principal benefício de utilizar *Xamarin* está no reaproveitamento de códigos quando o desenvolvimento de uma aplicação tem o foco em mais de uma plataforma.

É possível modificar somente a *interface* e as chamadas à *API's* específicas de cada SO. Com isso, a camada de negócios é reaproveitada, o que significa que o acesso a dados, chamadas de serviços e classes de domínios, por exemplo, serão funcionalidades reaproveitadas entre o desenvolvimento de aplicativos nas diferentes plataformas (XAMARIN, 2017).

Já a ferramenta *React Native*, juntamente com o *Xamarin*, é atualmente o mais próximo que uma aplicação híbrida consegue chegar de uma nativa. *React Native* permite desenvolvimento nas duas plataformas dominantes (*iOS* e *Android*), além de produzir aplicativos de alta qualidade, com *performance* e comportamento similar ao das aplicações nativas (DEVMEDIA, 2017).

As ferramentas híbridas foram desenvolvidas como forma de simplificar a criação de aplicativos móveis e padronizar as experiências em todos os SO. Inicialmente chamado de *PhoneGap*, o *Cordova* adquiriu esse nome após uma doação da empresa *Adobe* para a *Apache Software Foundation*. *Cordova,* é uma ferramenta para o desenvolvimento de *apps* híbridos, é gratuita e possui código aberto.

Essa ferramenta permite que aplicações híbridas tenham acesso às funcionalidades nativas dos dispositivos como a câmera, sensores, GPS, contatos por meio de *plugins* instalados no projeto da aplicação (BEZERRA; SCHIMIGUEL, 2016).

Outro exemplo de ferramenta bastante utilizada é *Ionic*, uma ferramenta também de *software* livre para o desenvolvimento de aplicativos híbridos lançada em 2013 pela empresa *Drifty Co*. Seu objetivo é oferecer um ótimo desempenho e funcionar com padrões e tecnologias *web* modernas como as linguagens de programação *JS, HTML5* e *CSS* (DRIFTY, 2017). A ferramenta *Ionic* tem como base os *frameworks Cordova* (híbrido) e *Angular JS* (*web*) possibilitando criar aplicativos para várias plataformas.

O *Titanium mobile* é mais uma ferramenta para criar aplicativos híbridos e é gerenciado pela *Appcelerator*, que possibilita um ambiente de desenvolvimento aberto e extensível para criar aplicativos nativos em diferentes dispositivos móveis e SO, como *iOS* e *Android* (TITANIUM, 2018).

Essa ferramenta é baseada em tecnologias *web* como *HTML5, JS* e *CSS*, além de incluir uma biblioteca de código aberto com diversos dispositivos e *API's* do SO escolhido. A *Appcelerator* fornece, juntamente com a ferramenta, o *Titanium Studio*, um (*IDE)* baseado em Eclipse que busca facilitar a configuração do ambiente de desenvolvimento para as diferentes plataformas (APPCELERATOR, 2017).

2.4.2. VANTAGENS DE UMA APLICAÇÃO HÍBRIDA

Aplicações híbridas possuem vantagens sobre os outros três métodos de desenvolvimento apresentados nesta seção. Cavalcanti (2016), Fowler (2012) e Heitkotter, Hanschke e Majchrzak (2013) apresentam cenários onde o desenvolvimento de uma aplicação híbrida seria a melhor opção, esses cenários estão descritos abaixo:

1. **Aplicações para o mercado corporativo:** é necessário atender necessidades específicas e não existem aplicações no mercado com a mesma função que está sendo proposta. Sabe-se especificamente quem são os usuários e quais dispositivos utilizarão;
2. ***Time to Market/Fast Deploy***: há uma grande necessidade de publicar a aplicação em pouco tempo. O foco do produto é validação, para que futuramente possa ser complementado;
3. **Manutenibilidade**: a capacidade de facilmente fornecer manutenção e atualizações constantes ao aplicativo é fundamental, pois inicialmente será importante realizar e testar melhorias;
4. **Seguir padrões de identidade visual**: é necessário manter um *design* similar em aplicações de diferentes plataformas;
5. **Aplicações com pouca interação com *API's* nativas**: quando não há necessidade de acesso frequente a *API's* nativas via *plugins*.

Cavalcanti (2016) ainda afirma que aplicações híbridas possuem a vantagem de necessitar de um trabalho menor na reusabilidade de código quando comparadas com aplicações nativas. Nas híbridas somente alguns ajustes são necessários, enquanto nas nativas o código não consegue ser reutilizado.

Na perspectiva de uma empresa que precisa de uma aplicação, as vantagens do desenvolvimento híbrido estão relacionadas com a possibilidade de disponibilizar aplicativos com a mesma identidade visual em plataformas distintas.

Em relação aos desenvolvedores, pode-se citar a variedade de ferramentas disponíveis e a redução de atividades redundantes. Além de aproveitar as tecnologias e práticas desenvolvidas anteriormente como o *design* do *software* e as aplicações anteriores (FERREIRA, 2012).

Além disso, as empresas fornecedoras de ferramentas têm a vantagem de lançar um único produto que poderá ser usado em diferentes plataformas. Por sua vez, os desenvolvedores de SO móveis encontram privilégios relacionados à concorrência, isso acontece, pois, com o aumento das ferramentas disponíveis no mercado maior a

competição e busca na melhoria da qualidade do serviço ofertado, dos preços e da disponibilidade de aplicativos previamente indisponíveis para certo SO (CAVALCANTI, 2016).

Por último, os usuários finais serão beneficiados com um maior número de aplicativos disponíveis para seu SO preferido (CORRAL; JANES; REMENCIUS, 2012).

2.4.3. DESVANTAGENS DE UMA APLICAÇÃO HÍBRIDA

A mudança no desenvolvimento de *softwares* móveis traz desvantagens que podem resultar em aplicativos malsucedidos. Diversos autores (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013; CAVALCANTI, 2016) afirmam que aplicações híbridas não possuem a mesma responsividade e fluidez que um aplicativo nativo, por este motivo uma aplicação híbrida não proporciona uma experiência de usuário tão completa quanto uma aplicação desenvolvida nativamente.

Um dos principais fatores para isso é devido ao fato de que um desenvolvedor nativo possui acesso a todo *hardware* do dispositivo sem a necessidade de *plug-ins*. Além disso, dependendo da aplicação desenvolvida e de seu objetivo, ajustes finos para cada navegador podem ser necessários pela parte do desenvolvedor utilizando o desenvolvimento híbrido.

Além da questão do *hardware*, a qualidade de uma aplicação híbrida está ligada a qualidade da ferramenta/*framework* de desenvolvimento utilizada, o que não acontece com aplicações nativas (CAVALCANTI, 2016).

Tecnologias nativas têm acesso às novas tecnologias lançadas por cada SO imediatamente. Por exemplo, ao lançar um novo *smartphone,* a *Google* disponibiliza para os desenvolvedores todos os recursos e métodos para acessar novas funcionalidades deste dispositivo. Todavia, no desenvolvimento híbrido os desenvolvedores precisam esperar um tempo até a criação de novos *plugins* para acessarem os novos recursos, o que é um fato negativo para empresas que desenvolvem exclusivamente no formato híbrido (CAVALCANTI, 2016).

Ainda com *plugins*, nem todos os recursos de um dispositivo são suportados por ferramentas híbridas. Esse fator implica em aplicações com problemas de desempenho, afetando a experiência do usuário. Por fim, os desenvolvedores são responsáveis por um bom gerenciamento dos recursos disponíveis nos variados dispositivos para evitar problemas de desempenho (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

2.5. PROGRESSIVE WEB APP (PWA)

PWA é a evolução dos aplicativos *web* para aplicativos da *web* "progressivos", que são aprimorados e aumentam as possibilidades técnicas das aplicações móveis (GOOGLE, 2017).

Essa abordagem de desenvolvimento de aplicações móveis foi proposta pela *Google*, sendo que o termo *PWA* foi definido em 2015 pelo engenheiro de *software* Alex Russell e pela *designer* Frances Berriman (RUSSELL, 2015).

Segundo Petele (2017), essa abordagem proporciona experiências que combinam o melhor da *web mobile* e dos *apps* nativos, sendo que não está relacionada a tecnologia de aplicações híbridas. Diferente de aplicações híbridas, as *PWA's* não precisam ser baixadas em lojas de aplicativos como *Apple Store* ou *Google Play*. Em outras palavras, as *PWA's* misturam diversas estratégias, técnicas e *API's* para criar uma experiência próxima a de uma aplicação nativa (SHEPPARD, 2017).

Uma *PWA* é inicialmente apresentada como uma aplicação *web mobile*, contudo, enquanto o usuário constrói uma relação com o *app* através do tempo, ela se torna mais poderosa. Diferente de aplicações *web*, uma *PWA* possui um manifesto, um arquivo com informações que permitem uma experiência do usuário imersiva com a tela cheia (PETELE, 2017).

Petele (2017) aponta seis características principais de uma *PWA:*

1. **Progressivo:** usa como base o progressive *enhancement,* o que torna uma *PWA* disponível para o usuário independentemente de seu navegador;
2. **Responsivos**: adaptam as *interfaces* do usuário de acordo com o tamanho da tela através do *web design* responsivo;

3. **Independentes de conectividade**: funcionam *offline* ou em redes de baixa qualidade através da tecnologia de *Service Workers;*
4. **Similares a aplicações nativas:** possuem interações, navegação e aparência similares a aplicações nativas;
5. **Atualizações**: toda vez que o *app* estiver *online* novas atualizações serão buscadas, não é necessário baixar uma atualização de tempos em tempos;
6. **Seguros:** utilizam o protocolo *HTTPS*, que assegura a proteção de dados;
7. **Descobríveis**: pois possuem um Manifesto[5] e *Service Workers*, o que permite que sejam encontrados por ferramentas de busca.
8. **Engajáveis**: permitem o uso de notificações *push* para trazê-los de volta à experiência;
9. **Instaláveis:** permitem que usuários salvem um ícone do *app* na tela inicial de seus dispositivos;
10. **Linkáveis**: possuem uma *URL* que pode ser facilmente compartilhada.

Os *service workers* (*SW's*) são *scripts* executados no plano de fundo de uma aplicação *web* com o objetivo de atuar como intermediário entre o *app* e a *internet*, enviando mensagens ao aplicativo quando determinada tarefa foi cumprida.

Seu funcionamento é completamente separado da *interface* do usuário, não causando lentidão ou congelamento da *UI*. Os *SW's* são responsáveis por deixar parte de conteúdo da *PWA* armazenada na memória *cache* do navegador, e recuperar esses dados quando a aplicação é iniciada. Essa funcionalidade garante um carregamento rápido e a disponibilidade *offline* dessas aplicações (SHEPPARD, 2017).

Quando um usuário abre uma aplicação nativa, uma tela de carregamento é exibida em poucos segundos. Já na *web*, normalmente o carregamento demora vários segundos e o usuário interage com uma tela branca. O modelo *shell* de aplicativos resolve esse problema nas *PWA's*. Uma *shell* é o conteúdo mínimo necessário para construir uma *UI*, o que pode ser um menu, abas, ou qualquer outro elemento de *interface*. Esse esqueleto é exibido enquanto o conteúdo dinâmico da aplicação é carregado, essa arquitetura é parte natural de um *front-end* de qualquer aplicação, e é a parte de uma

[5] https://developers.google.com/web/fundamentals/web-app-manifest/

PWA que deve ser armazenada em *cache* utilizando *SW's* para ficar disponível *offline* (SHEPPARD, 2017).

Single-page applications (*SPA's*) são aplicações *web* que utilizam somente uma página *HTML* como *shell* para todas as outras páginas *web* da aplicação. As interações dos usuários são implementadas usando *HTML, CSS* e *JS*. Essas aplicações não realizam uma nova requisição ao servidor, e sim são atualizadas no próprio *front-end* (FINK; FLATOW, 2014). *SPA's,* são recomendados pela *Google* como formato ideal para o desenvolvimento de uma *PWA* (GOOGLE DEVELOPERS, 2018).

Apesar de ser uma abordagem recente, a *Google* já conduziu vários estudos de caso com *PWA's*[6]. Em um desses estudos a *PWA* da rede social *Twitter*, o *Twitter Lite*, foi analisada. O objetivo da empresa era obter uma experiência *web mobile* mais rápida, segura e engajável. O antigo *web app* do *Twitter* migrou globalmente para uma *PWA* em Abril de 2017, e é mais robusto em relação a sua versão *web*, mas reduziu o consumo de dados e aumentou o engajamento com os usuários.

Com essa migração, a empresa obteve ótimos resultados, aumentando o número de envio de *Tweets* em 75%, reduzindo o custo de desenvolvimento, e crescimento de 65% no número de páginas que um usuário acessa por sessão. É importante ressaltar que o *Twitter Lite* ocupa menos de 3% do espaço necessário para instalação de um aplicativo nativo do *Twitter* para *Android* por exemplo (GOOGLE DEVELOPERS, 2017).

2.5.1. FERRAMENTAS PARA O DESENVOLVIMENTO DE *PWAS*

Atualmente, existem diversos *frameworks front-end* que permitem o desenvolvimento de *SPA's*. As ferramentas mais populares são *Vue.js*[7], *React.js*[8] e *Angular*[9]. Cada um dos *frameworks* serão descritos a seguir.

[6] https://developers.google.com/web/showcase/
[7] https://vuejs.org/
[8] https://reactjs.org/
[9] https://angular.io/

Vue.js é um *framework* progressivo em JS que tem o objetivo de construir *UI's*. Essa ferramenta foi criada em 2014 por Evan You, e suas principais características são a versatilidade de escalamento, bom desempenho pois seus projetos não ocupam muito espaço, e facilidade de aprendizado para desenvolvedores com conhecimento básico de *HTML5, CSS* e *JS* (VUEJS, 2018).

Vue.js possui um *template* para aplicações *PWA*, que pode ser escolhido durante a criação de uma aplicação[10].

React.js é uma biblioteca *JS* para construir *UI's* apresentada pelo *Facebook*. Suas principais características são o uso de visualizações declarativas do estado da aplicação, base em componentes e possibilidade de criar novas funcionalidades sem modificar o código previamente escrito (FACEBOOK INC, 2018).

Essa biblioteca não oferece um *template* próprio para *PWA*, porém, *plugins* podem ser utilizados para adicionar as funcionalidades extras, como *SW's*.

Angular é uma plataforma, proposta pelo *Google*, para construir aplicações *web*, *mobile* ou *desktop*. Suas principais características são o uso de *templates* declarativos, injeção de dependências e a integração de boas práticas para resolver problemas de desenvolvimento (GOOGLE, 2018). Assim como *React.js*, *Angular* não oferece um *template* para o desenvolvimento de *PWA's*, mas pode-se utilizar *plugins*.

2.5.2. VANTAGENS DE UMA *PWA*

Devido a sua natureza mista entre aplicações *web* e nativas, um *PWA* possui vantagens relacionadas aos dois tipos de abordagens. Sheppard (2017), afirma que uma *PWA* funciona em qualquer lugar, mesmo que seja uma simples aplicação estática sem muitas funcionalidades. A seguir, são listadas as vantagens de aplicações *PWA* em relação a aplicações *web mobile,* híbridas e nativas.

[10] https://github.com/vuejs-templates/pwa

Ao comparar *PWAs* com aplicações *web mobile,* observa-se que os ***PWA's*** possuem disponibilidade *offline* através do uso de ***SW's.*** Essa tecnologia permite um carregamento inicial em poucos segundos, além de gerar um menor consumo de dados, já que parte da aplicação foi previamente armazenada em memória *cache*. Além desse fator, *PWAs* podem ser adicionados à tela inicial do dispositivo móvel, e ao serem iniciados mostram uma tela de carregamento similar a de aplicações nativas (SHEPPARD, 2017; PETELE, 2017).

Aplicações nativas e híbridas, precisam ser pesquisadas numa loja de aplicativos, o que não ocorre com ***PWA's***. Essas aplicações progressivas podem ser encontradas através de ferramentas de busca da *web*, em repositórios de ***PWA's*** como *PWA Rocks*[11], *Awesome PWA*[12] e lista de estudos de caso da *Google*[13] (CARDIERI; ZAINA, 2018).

Ainda nesse aspecto, ***PWA's*** estão sempre atualizados quando há conexão com a *internet*, diferente de *apps* nativos e híbridos que requerem uma autorização manual dos usuários para instalar novas atualizações. A natureza baseada em *web design* responsivo permite que ***PWA's*** sejam executados em qualquer dispositivo com um navegador, como *desktops* (SHEPPARD, 2018).

2.5.3. DESVANTAGENS DE UMA *PWA*

Atualmente, uma das principais desvantagens de uma *PWA* é a falta de suporte de navegadores. De acordo com o *website "Can I Use?"*[14], ***SW's*** são suportados por completo pelas versões mais recentes da maioria dos navegadores como *Chrome*, *Firefox, Safari, Edge* e *Opera*.

Contudo, o SO *iOS* ainda não suporta o uso da *Push API*, responsável pelo envio de notificações *push* pela *web*. Essa funcionalidade é a responsável pelo re-engajamento dos usuários, um dos principais benefícios listado nos estudos de caso conduzidos pela

[11] https://pwa.rocks/
[12] https://github.com/hemanth/awesome-pwa
[13] https://developers.google.com/web/showcase/
[14] https://caniuse.com/#feat=serviceworkers

Google[15]. Já o manifesto só é suportado pelos navegadores *Safari mobile, Edge*, *Chrome* e *Firefox for Android*[16] (SHEPPARD, 2018).

Por meio do estudo realizado sobre as diferentes tecnologias e SO é possível fazer um comparativo entre as quatro abordagens de desenvolvimento de aplicações móveis. Esse comparativo será apresentado no próximo capítulo.

[15] https://developers.google.com/web/showcase/
[16] https://caniuse.com/#feat=web-app-manifest

3. DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento da abordagem proposta neste trabalho. Seu objetivo é proporcionar apoio à decisão dos desenvolvedores de aplicativos móveis que precisam escolher entre as abordagens nativa, híbrida, *web mobile* ou *PWA*.

3.1. FLUXO DE ATIVIDADES

Ao longo do desenvolvimento deste trabalho, buscou-se mostrar as diferentes abordagens para o desenvolvimento de aplicativos móveis, incluindo suas vantagens, desvantagens e requisitos técnicos mais relevantes para a tomada de decisão dentre as tecnologias sendo elas, nativa, híbrida, *web mobile* e *PWA*.

De acordo com Fling (2009), em um projeto de construção de aplicativos, é necessário considerar aspectos como: público alvo, funcionalidades, SO, dispositivos, prazos e custos. Todas essas variáveis tornam os aplicativos móveis um grande desafio para os desenvolvedores, principalmente na escolha da tecnologia mais aderente.

Desta maneira, quanto mais detalhadas e trabalhadas forem as informações sobre o que deve ser desenvolvido e qual o público-alvo a ser atingido, mais assertiva será a escolha de uma abordagem de desenvolvimento.

No desenvolvimento de um sistema computacional, a escolha do modelo de ciclo de vida é a primeira etapa a ser realizada no processo de criação do *software*. O ciclo de vida é a estrutura composta por processos, atividades e tarefas envolvidas no desenvolvimento, manutenção e operação de um produto de *software,* desde a coleta de requisitos até a entrega.

A Figura 5, apresenta o fluxo de atividades macro das fases de construção de um *software*, baseado no Guia do Conhecimento em Gerenciamento de Projetos do *PMI,*

sendo que a seguir será aprofundado o processo de decisão da abordagem de desenvolvimento de acordo com os requisitos necessários.

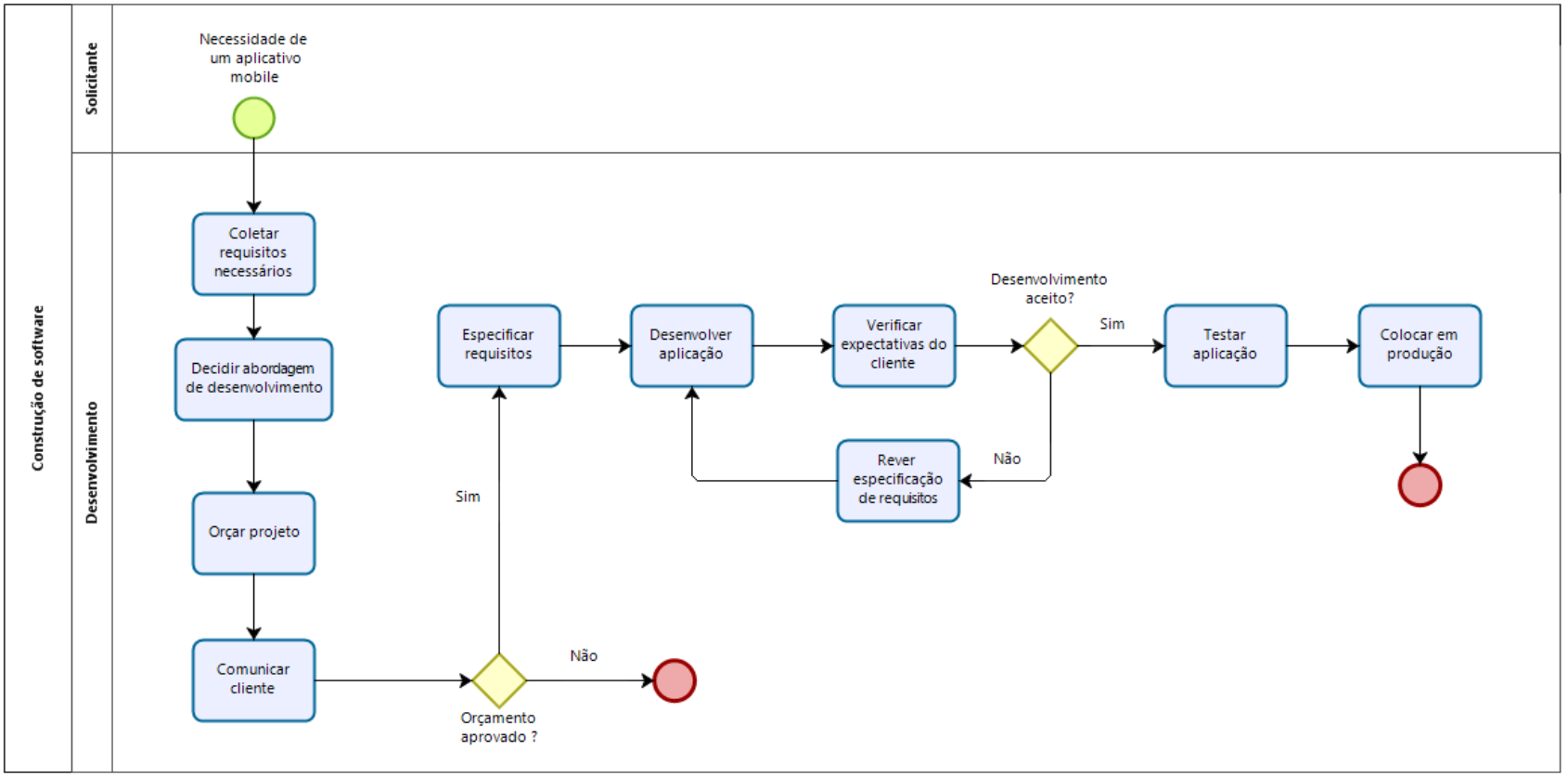

FIGURA 5 – FLUXO DE ATIVIDADES MACRO – FONTE: A AUTORA, 2018.

3.2. OBTENÇÃO DE REQUISITOS

Durante a coleta de requisitos, são identificadas as principais funcionalidades técnicas solicitadas pelas partes interessadas para compor o aplicativo. Cada aplicação possui requisitos diferentes e próprios, com isso, é possível analisar qual tecnologia será mais adequada diante do cenário proposto.

As próximas etapas visam aprofundar as avaliações comparativas entre as abordagens de aplicação móvel. Esta análise irá se basear em critérios técnicos a fim de auxiliar e fornecer insumos aos desenvolvedores no momento da adoção de um método de desenvolvimento móvel.

3.3. VARIÁVEIS DO FLUXOGRAMA

Após a leitura e comparação das três plataformas realizadas na Seção 2, foram identificadas as variáveis técnicas mais relevantes para a tomada de decisão, sendo elas: tempo, conhecimento de equipe, recursos de *hardware,* custo, atualizações, uso *offiline,* esforço inicial do usuário, desempenho, experiência do usuário e plataforma do público alvo.

Os critérios técnicos serão elencados nesta seção por meio da utilização da abordagem *GQM* e serão utilizados posteriormente para o desenvolvimento dos fluxogramas destinados a apoiar a tomada de decisão em relação a abordagem que será utilizada para o desenvolvimento de aplicativos *mobile*.

Com base na abordagem *GQM*, será elaborado um modelo composto pelas variáveis identificadas juntamente com os itens que integram o GQM, sendo eles:

(a) critérios técnicos de desenvolvimento móvel;

(b) questões relacionadas a análise técnica;

(c) métricas.

Uma descrição de cada um deles é feita a seguir.

a. Critérios técnicos: são características técnicas que devem conter o aplicativo móvel. Os critérios técnicos possuem particularidades que podem torna-los mais ou menos vantajosos em relação a escolha de determinada abordagem de desenvolvimento.
b. Questões de análise técnicas: são as perguntas que desenvolvedores devem indagar-se para encontrar a solução mais adequada para o desenvolvimento de uma aplicação móvel.
c. Métricas: tem como propósito, guiar os desenvolvedores a responder às questões de maneira objetiva.

Na Tabela 2, estão descritos os critérios, as questões técnicas e as métricas selecionadas.

TABELA 2 - CRITÉRIOS, QUESTÕES DE ANÁLISE E MÉTRICAS PROPOSTAS PARA ESCOLHA DA MELHOR ALTERNATIVA DE DESENVOLVIMENTO MÓVEL.

Item	Critérios	Questões de análise	Métricas
I1	Tempo	Qual das opções técnicas leva menos tempo para ser desenvolvida?	M1. Estimar o tempo de desenvolvimento de um *app* móvel.
I2	Conhecimento da equipe	A equipe de desenvolvedores atual possui conhecimento para quais tecnologias?	M2. Analisar se a equipe atual consegue desenvolver a aplicação requerida.
I3	Recursos de *hardware*	O aplicativo utilizará muitos recursos do *hardware*? Quais recursos serão utilizados? Quais alternativas podem alcançar esta necessidade?	M3. Determinar quais recursos de *hardware* a serem utilizados.
			M3.1 Determinar qual a frequência de utilização de recursos.
			M3.2 Checar quais abordagens permitem o uso dos recursos na frequência requerida.
I4	Custo	Qual das opções de desenvolvimento é a mais barata?	M4. Estimar o custo de desenvolvimento de cada tecnologia.
I5	Atualizações	É um produto que requer atualizações de conteúdo frequentes? Em qual das tecnologias essa característica é mais viável de se realizar?	M6. Determinar qual a frequência de atualização de conteúdo da aplicação.
I6	Uso *offline*	É necessário que o aplicativo esteja disponível quando não há acesso à internet? Qual das alternativas possibilita este tipo de acesso?	M7. A solução deve funcionar *offline*.

I7	Esforço inicial do usuário	A facilidade no uso inicial de uma aplicação é um fator relevante? Há necessidade de publicar o app em uma loja para o usuário instalá-lo?	M8 Checar quais plataformas exigem que o app seja instalado a partir de uma loja de aplicativos.
I8	Desempenho	O desempenho do aplicativo durante o uso é um fator relevante?	M9 Determinar qual abordagem oferece o melhor desempenho.
I9	Experiência do Usuário	A experiência do usuário durante as interações com a aplicação é um fator relevante?	M10 Definir qual das opções de desenvolvimento proporciona a maior satisfação dos usuários.
			M10.1 Definir qual das opções de desenvolvimento é mais eficaz para os usuários completarem suas tarefas.
			M10.2 Definir qual das opções de desenvolvimento é mais eficiente nas interações com o usuário.
			M10.3 Definir qual das opções de desenvolvimento proporciona o mais alto grau de usabilidade.
I10	Plataforma do público-alvo	É necessário que a aplicação funcione em mais de uma plataforma? A maioria dos usuários utiliza um SO específico?	M11 Checar quais em quais SO o app deve funcionar, considerando o perfil do público-alvo.

Fonte: A Autora (2018)

3.3.1. CRITÉRIOS

A escolha de cada um dos critérios apresentados na Tabela 4 tiveram como base a comparação entre métodos de desenvolvimento realizada na Seção 2, a partir disso, pode-se compreender as principais individualidades das abordagens nativa, híbrida, web *mobile* e *PWA*. A seguir, serão justificadas as escolhas dos critérios e os fluxogramas para tomada de decisão

As desvantagens das aplicações nativas mostram a relevância de critérios como *I1- Tempo, I2 - Conhecimento da equipe, I4 - Custo, I5 - Atualizações* e *I7 - Esforço inicial do usuário.* Já que seu desenvolvimento possui um alto custo, pois, desenvolver para plataformas nativas exige ambiente, infraestrutura e conhecimentos diferentes para cada plataforma. Dessa forma, quanto mais plataformas se quer alcançar, mais custoso o projeto será.

A abordagem nativa requer um tempo maior de desenvolvimento, pois necessita de um desenvolvimento particular para cada SO que a aplicação será utilizada, pois, quanto mais plataformas para atender, maior é o tempo necessário para desenvolver a solução. Se o prazo do projeto for apertado para desenvolvimento de mais de uma solução nativa, há de considerar o desenvolvimento híbrido, *PWA* ou *web app*, visto que apenas será codificada uma solução que poderá atender várias plataformas diferentes (WHITE, 2013).

Em relação ao critério conhecimento de equipe, o desenvolvimento nativo requer uma equipe com conhecimentos em tecnologias distintas, pois, cada equipe possui um conjunto único de habilidades e conhecimentos. No momento da escolha de uma abordagem, esses conhecimentos devem ser levados em consideração, visto que é a equipe de desenvolvimento que irá conceber o produto final.

Por sua vez, o critério esforço inicial do usuário aborda o empenho necessário para instalação do aplicativo e para as possíveis atualizações (DUA, 2018; CHARLAND; LEROUX 2011). Para as aplicações desenvolvidas nativamente e híbridas, há a necessidade que o usuário demande um maior esforço, pois, o mesmo deve acessar a loja de aplicativos do seu SO, procurar pela aplicação desejada e instalar no seu dispositivo. Para as atualizações, as etapas que requerem esforço dos usuários são iguais a de instalação de um novo aplicativo.

Já os critérios *I3 - Recursos de hardware*, *I6 - Uso off-line, I8 - Desempenho* e *I9 - Experiência do usuário* são justificados pelas desvantagens de aplicações *web* e *PWA*. Em relação a recursos de hardware, essas aplicações não têm acesso a todos os recursos nativos de *hardware* do dispositivo pois são acessados via navegadores, os

mesmos não possuem todas as tecnologias necessárias para acessar diretamente determinados recursos de hardware (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013; SHEPPARD, 2018).

Em relação ao critério uso *off-line*, diferente de um aplicativo nativo e híbrido, aplicações *PWA* precisam acessar a internet para baixar os conteúdos contidos no aplicativo para o *cache* do celular, após isso, o usuário consegue acessar alguns recursos de maneira *off-line.*

Já aplicações web funcionam apenas quando os usuários estão com acesso à internet, portanto, se o usuário não tiver conexão em seu aparelho, ele não conseguirá acessar o aplicativo.

O critério de desempenho torna aplicativos web desfavoráveis em relação a aplicações hibridas e nativas, pois, latência, instabilidade, baixa largura de banda e cobertura precária das operadoras são alguns dos problemas que afetam os navegadores dos dispositivos móveis e isto prejudica de maneira drástica a *performance* da aplicação.

Aplicações *web mobile* e *PWA* são páginas da web que simulam um aplicativo nativo, por isso, não acessam todas as funcionalidades de um dispositivo como notificações *push* e sensores, tornando a usabilidade incompleta (FLING, 2009).

Por fim, o critério *I10 - Plataforma do público-alvo* foi selecionado como maior desvantagem para aplicações desenvolvidas pela tecnologia *PWA,* pois, não possuem suporte completo no SO *iOS* (SHEPPARD, 2017).

Este critério também é relevante pois auxilia a estimar o custo e tempo de construção de aplicações nativas e híbridas, já que o código final é completamente diferente para cada SO existente no mercado.

Cada plataforma móvel domina uma parcela do mercado e possui um grupo de usuários com características, gostos próprios e necessidades diferentes típicas de cada plataforma. Com isso, no momento de criar um *app* deve-se pensar para quem é esse aplicativo. Se ele será concebido para suprir uma demanda de um grupo específico,

talvez não haja a necessidade de criá-lo para várias plataformas existentes (LOPES, 2013).

3.3.2. QUESTÕES DE ANÁLISE

De acordo com as vantagens e desvantagens de cada tecnologia de desenvolvimento móvel abordadas na Seção 2, pode-se ter um conceito geral das respostas para as questões de análise construídas. Porém, as respostas das questões e o resultado das métricas podem variar de acordo com a finalidade da aplicação exigida por cada empresa solicitante. Por isso, todas as questões levantadas na Tabela 2 serão descritas de maneira mais detalhada abaixo:

Pergunta I1: Qual das opções técnicas leva menos tempo para ser desenvolvida?

Resposta I1: Aplicações *web mobile*, *PWA* e híbridas costumam levar menos tempo para serem desenvolvidas pois possuem linguagem padronizada tornando possível o aproveitamento do mesmo código desenvolvido (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013). Já o desenvolvimento nativo, devem ser desenvolvidos uma versão diferente para cada sistema operacional móvel alvo, aumentando assim o seu tempo de criação (CHARLAND; LEROUX, 2011).

Pergunta I2: A equipe de desenvolvedores atual possui conhecimento para quais tecnologias?

Resposta I2: O time de desenvolvimento deve listar os conhecimentos de cada um dos membros de desenvolvedores e em conjunto com sua empresa devem chegar à conclusão de quais tecnologias e abordagens já são conhecidas pelo time.

Pergunta I3: O aplicativo utilizará muitos recursos do *hardware*? Quais recursos serão utilizados? Quais alternativas podem alcançar esta necessidade?

Resposta I3: Como são instalados diretamente no aparelho, aplicativos nativos e híbridos permitem o uso de todos os recursos de *hardware* disponíveis no dispositivo móvel (EL-KASSAS et al., 2015, CHARLAND; LEROUX, 2011). Porém, aplicativos

nativos possuem acesso imediato as novas tecnologias lançadas, como exemplo: realidade virtual. Isso se dá, pois, os fornecedores disponibilizam *frameworks* que possuem as funções para acessar as novas tecnologias.

Já no desenvolvimento híbrido, é preciso criar *plugins* para acessar estes novos recursos. Os desenvolvedores de aplicativos híbridos precisam aguardar até que o fabricante reporte as novas tecnologias oferecidas e que o *plugin* de acesso seja criado (CAVALCANTI, 2016).

Aplicações *PWA* e web *mobile* estão limitadas aos recursos do navegador utilizado (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013). Por isso, é importante listar os recursos necessários e verificar se é compatível com o navegador mais utilizado pelo público alvo.

Com isso, podemos concluir que se o aplicativo for complexo, com acesso frequente à vários recursos do *hardware* do dispositivo, o desenvolvimento nativo proporcionará melhor experiência ao usuário (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013). Caso contrário, aplicações híbridas, *web mobile* ou *PWA* conseguem cumprir os requisitos necessários.

Pergunta I4: Qual das opções de desenvolvimento é a mais barata?

Resposta I4: Aplicativos nativos precisam ser desenvolvidos separadamente para cada SO móvel, o que requer um número maior de horas trabalhadas e desenvolvedores com conhecimentos específicos para cada plataforma (WHITE, 2013; CHARLAND; LEROUX, 2011). Aplicações híbridas podem ser aproveitadas para múltiplas plataformas, contudo, ajustes no código podem ser necessários (CAVALCANTI, 2016).

Já aplicativos *web* e *PWA* podem ser desenvolvidos uma única vez e não precisam ser submetidos em lojas de aplicativos, as quais cobram taxas dos desenvolvedores (FLING, 2009). Com isso, aplicações *web* e *PWA* oferecem o menor custo.

Pergunta I5: É um produto que requer atualizações de conteúdo frequentes? Em qual das tecnologias essa característica é mais viável de se realizar?

Resposta I5: *Apps web* e *PWA's* são mais facilmente atualizáveis, a própria equipe de desenvolvimento realiza a atualização do novo conteúdo para o servidor. Com isso, a atualização é exibida ao usuário assim que ele se conectar à *internet* (PETELE, 2017).

Já as atualizações dos aplicativos nativos ou híbridos precisam ser submetidas às lojas virtuais. Essa modificação passa por um processo de análise na loja de aplicações, assim como é feito com a publicação de um novo aplicativo. Após aceito, os usuários que já aderiram o aplicativo precisam baixar a atualização manualmente para acessar as mudanças (DUA, 2018; CHARLAND; LEROUX, 2011).

Pergunta I6: É necessário que o aplicativo esteja disponível quando não há acesso à internet? Qual das alternativas possibilita este tipo de acesso?

Resposta I6: Aplicações web dependem da internet em cada nova interação (LOPES, 2013). Já aplicações nativas, híbridas e *PWAs* podem ser acessados *off-line*. Esses *apps* utilizam tecnologias para deixar parte do seu conteúdo, como UIs e arquivos base, baixados no dispositivo (SHEPPARD, 2017; PETELE, 2017).

Pergunta I7: A facilidade no uso inicial de uma aplicação é um fator relevante? Há necessidade de publicar o *app* em uma loja para o usuário instalá-lo?

Resposta I7: Para conseguir utilizar um aplicativo nativo ou híbrido, o usuário deve buscar, e posteriormente baixar, o aplicativo na loja (FLING, 2009). Com isso, um maior esforço do usuário é necessário. Já para acessar uma PWA ou *app web*, o usuário precisa somente digitar o endereço no navegador do *app* desejado (SHEPPARD, 2018).

Pergunta I8: O desempenho do aplicativo durante o uso é um fator relevante?

Resposta I8: Os aplicativos desenvolvidos por tecnologia nativa ou híbrida geralmente oferecem melhor desempenho. Isso acontece, pois, parte da aplicação fica instalada no dispositivo, além de possuir acesso a todos os recursos de hardware (EL-KASSAS et al., 2015, CHARLAND; LEROUX, 2011). Se a finalidade do aplicativo a ser construído consome muitos recursos do hardware, o aplicativo nativo atenderá melhor a

necessidade por ser mais rápido e confiável (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

No caso de a aplicação não necessitar de acesso a todos os recursos de *hardware* ou não os acessar com frequência, aplicações *PWA* e *web mobile* podem ser consideradas (CARDIERI; ZAINA, 2018).

Vale ressaltar que para atingir um bom desempenho em qualquer forma de desenvolvimento de um *software* móvel, é importante que os desenvolvedores tenham conhecimento do *framework* que estão utilizando para a construção do código.

Pergunta I9: A experiência do usuário durante as interações com a aplicação é um fator relevante?

Resposta I9: Neste quesito, alguns autores (CHAMMAS, QUARESMA, MONT'ALVÃO, 2014; DUA, 2018; WHITE, 2013) afirmam que o desenvolvimento nativo alcançará melhor experiência de uso, uma vez que utiliza a experiência nativa do dispositivo.

Contudo, estudos recentes afirmam que a experiência do usuário não possui diferenças entre aplicações nativas, *web* e *PWA's* (CARDIERI; ZAINA, 2018). Em um desenvolvimento híbrido, *web* ou *PWA*, a experiência depende do *framework* utilizado e dos conhecimentos do desenvolvedor (CEVALLOS, 2014). Para as tecnologias web mobile e PWA, há restrição em relação ao acesso de algumas funcionalidades, que tornam a experiência do usuário incompleta.

Pergunta I10: É necessário que a aplicação funcione em mais de uma plataforma? A maioria dos usuários utilizam um SO específico?

Resposta I10: Uma aplicação desenvolvida por tecnologias *web* e *PWA* podem ser acessadas por usuários de qualquer SO através de um navegador (PETELE, 2017). Contudo, *PWA* possui limitações relacionadas ao engajamento de usuários de *iOS* (SHEPPARD, 2017). Já para aplicações nativas e híbridas, os usuários só podem baixar as aplicações que estão disponíveis em seu SO, já que nem todo aplicativo é

desenvolvido para todas as plataformas (WHITE, 2013). Com isso, é necessário entender se o público-alvo da aplicação está concentrado em somente um SO ou não.

3.3.3. MÉTRICAS

Na Tabela 3 serão descritas com detalhes as métricas selecionadas para responder às questões de análise propostas na Tabela 2. O objetivo da descrição é guiar empresas e desenvolvedores a entender como a resposta de uma questão de análise pode ser medida.

TABELA 3 - CRITÉRIOS, MÉTRICAS E SUAS DESCRIÇÕES.

Item	Critérios	Métricas	Descrição
I1	Tempo	M1. Estimar o tempo de desenvolvimento de um *app* móvel.	Considerar aspectos relacionados ao contexto da aplicação, número e experiência de membros da equipe de desenvolvimento (I2) e plataformas necessárias (I10)
I2	Conhecimento da equipe	M2. Analisar se a equipe atual consegue desenvolver a aplicação requerida.	Criar um questionário para permitir que cada membro da equipe se auto avalie em relação aos conhecimentos técnicos que possui.
I3	Recursos de *hardware*	M3. Determinar quais recursos de *hardware* a serem utilizados.	Listar as funcionalidades de *hardware* que devem ser utilizadas pela aplicação.
		M3.1 Determinar qual a frequência de utilização de recursos.	Utilizar a escala Likert[17] para classificar o quão frequente é a utilização de cada funcionalidade de *hardware*.
		M3.2 Checar quais abordagens permitem o uso dos recursos na frequência requerida.	Considerando as plataformas do público-alvo (I10), checar se os recursos necessários de *hardware*

[17] https://www.britannica.com/topic/Likert-Scale

			são acessíveis em todas as abordagens.
I4	Custo	M4. Estimar o custo de desenvolvimento de cada tecnologia.	Estimar o custo com base no tempo (I1) e número de pessoas na equipe de desenvolvimento.
I5	Atualizações	M6. Determinar qual a frequência de atualização de conteúdo da aplicação.	Utilizar a escala Likert para determinar a frequência de atualização de conteúdo.
I6	Uso *offline*	M7. A solução deve funcionar *offline.*	Analisar e definir com o cliente e os desenvolvedores a necessidade de permitir que a aplicação funcione *offline*.
I7	Esforço inicial do usuário	M8. Checar quais plataformas exigem que o *app* seja instalado a partir de uma loja de aplicativos.	Analisar e definir com o cliente e os desenvolvedores a necessidade da aplicação ser encontrada via loja de aplicativos ou via web.
I8	Desempenho	M9. Determinar qual abordagem oferece o melhor desempenho.	Considerando aspectos de recursos de *hardware* (I3) e experiência do usuário (I9), escolher abordagens onde o desempenho esperado possa ser alcançado.
I9	Experiência do Usuário	M10. Definir qual das opções de desenvolvimento proporciona a maior satisfação dos usuários.	Utilizar métodos de avaliação com o usuário para definir qual abordagem pode proporcionar maior satisfação durante as interações.
		M10.1 Definir qual das opções de desenvolvimento é mais eficaz para os usuários completarem suas tarefas.	Utilizar métodos para medir a eficácia de uma aplicação desenvolvida com determinada abordagem.
		M10.2 Definir qual das opções de desenvolvimento é mais eficiente nas interações com o usuário.	Utilizar métodos para medir a eficiência de uma aplicação desenvolvida com determinada abordagem.

I10	Plataforma do público-alvo	M11. Checar em quais SOs o *app* deve funcionar, considerando o perfil do público-alvo.	Traçar um perfil do público-alvo para entender se estão concentrados em algum SO ou se estão espalhados.

Fonte: A Autora, 2018

3.4. FLUXOGRAMA DE TOMADA DE DECISÃO

A seguir estão representados os fluxogramas do processo de tomada de decisão baseados nos critérios técnicos elencados. Os fluxogramas para tomada de decisão visam apoiar a escolha entre a melhor opção dentre as tecnologias de desenvolvimento para a construção de aplicações *mobile* de acordo com suas vantagens e desvantagens em relação aos requisitos necessários.

Mesmo que as variáveis conhecimento da equipe, tempo e custo foram identificadas como critérios e descritas na Tabela 2, não serão utilizadas na construção dos fluxogramas, pois, são requisitos técnicos interdependentes, o que torna difícil análises individuais de cada um destes critérios.

Por exemplo, o critério técnico tempo está relacionado a senioridade dos desenvolvedores, ou seja, do conhecimento de equipe. Ainda, em alguns casos de projetos de *software mobile*, as empresas podem já ter desenvolvido aplicativos similares ao solicitado. Com isso, por mais que o aplicativo tenha alto grau de complexidade, os desenvolvedores podem reutilizar um código utilizado já desenvolvido. Desta forma, os critérios acima estão diretamente relacionados a precificação do aplicativo a ser desenvolvido.

Foram construídos três fluxogramas com base na similaridade das saídas dos critérios técnicos analisados. O fluxograma da Figura 6 apresenta como requisitos de entrada, três critérios técnicos, sendo eles: Recursos de *Hardware*, Desempenho e Experiência do Usuário.

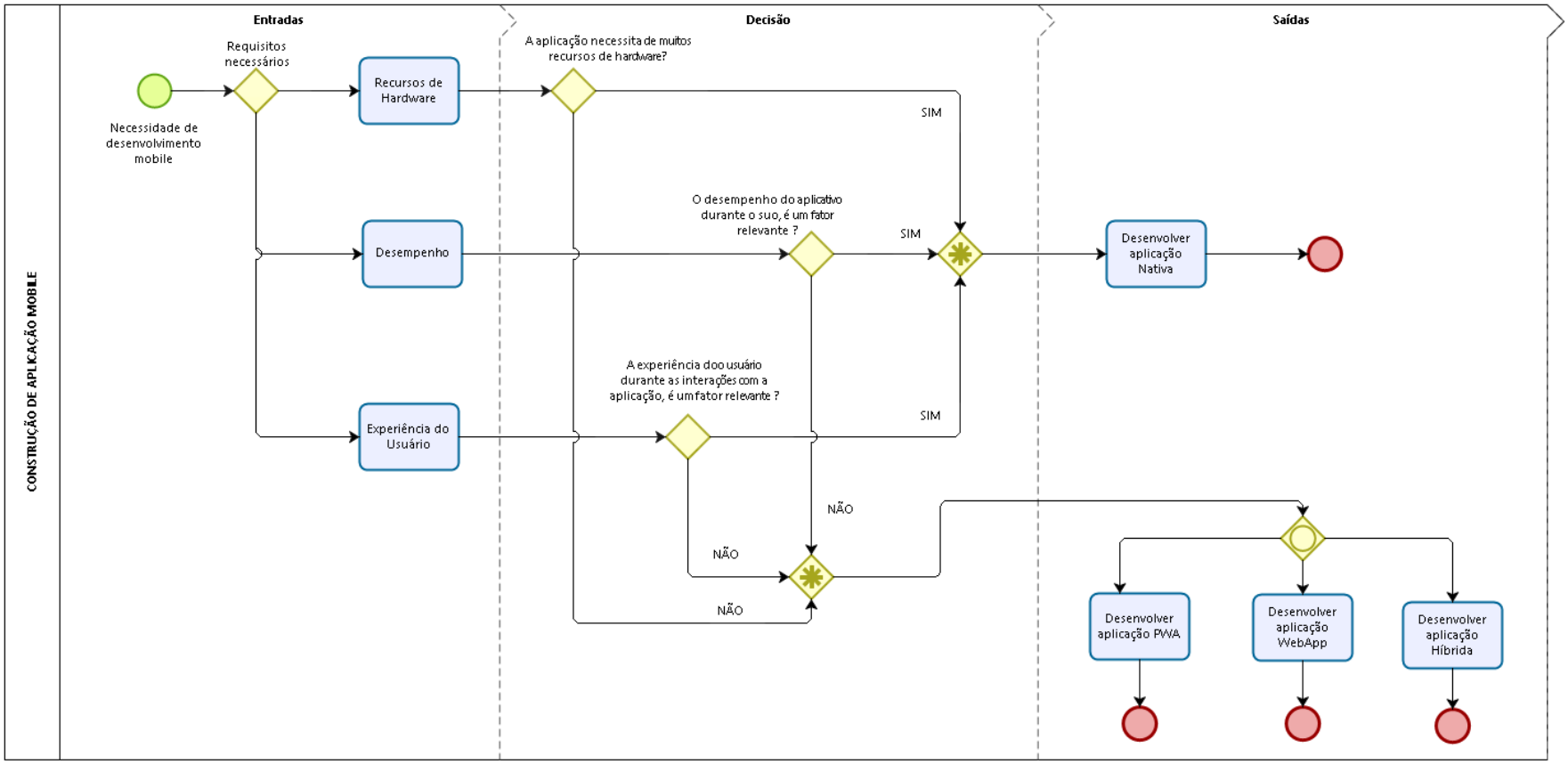

FIGURA 6 – FLUXOGRAMA DE TOMADA DE DECISÃO I - FONTE: A AUTORA, 2018

Em relação ao critério técnico do uso intenso de recursos de *hardware* e do uso imediato de novos recursos, a melhor escolha de desenvolvimento da aplicação é a abordagem nativa, pois, os aplicativos são desenvolvidos com as linguagens de programação própria do sistema operacional oferecidas pelo seu *kit* de desenvolvimento de *software*.

Os aplicativos nativos possuem a grande vantagem de poder acessar as funcionalidades do sistema operacional de maneira mais fácil, uma vez que a linguagem de desenvolvimento é nativa ao próprio sistema. Além disso, os próprios fabricantes disponibilizam de maneira imediata aos desenvolvedores *frameworks* que possuem as funções de acesso a determinados recursos.

Para o critério técnico desempenho, a melhor escolha também é a abordagem nativa caso o aplicativo exija excelência na responsividade e fluidez. Como os aplicativos nativos possuem acesso direto ao sistema operacional do dispositivo e por terem sido programados na linguagem padrão do dispositivo, são mais velozes que os outros tipos de desenvolvimento.

Além disso, a *performance* é mais negativa nos outros tipos de desenvolvimento, uma vez que, os dispositivos passam por constantes atualizações e nem sempre o desenvolvedor do aplicativo híbrido, *web* e *PWA* conseguem acompanhar os novos recursos e melhorias no sistema do *smartphone*.

O último critério abordado no fluxograma da Figura 6 é a experiência do usuário, sendo a melhor escolha de desenvolvimento nativo caso a aplicação necessite de um alto nível de usabilidade e sabe-se que o público-alvo é adepto a utilizar as novas tecnologias.

Os aplicativos nativos promovem uma experiência mais consistente ao sistema operacional e aos aplicativos padrões do SO móvel. Eles podem ser visualmente mais agradáveis, pois são desenhados de acordo com a linguagem e *design* do SO.

O desenvolvimento nativo irá focar ao máximo em extrair as características de cada sistema o que gera maior facilidade de uso para os usuários que estão adeptos a

utilização rotineira de uma determina plataforma. Com isso, irá criar uma experiência fiel ao sistema operacional usado, seja ele *Android*, *iOS* ou qualquer outro.

O fluxograma da Figura 7 apresenta como requisitos de entrada dois critérios técnicos, sendo eles: uso *off-line* da aplicação e a plataforma do público-alvo.

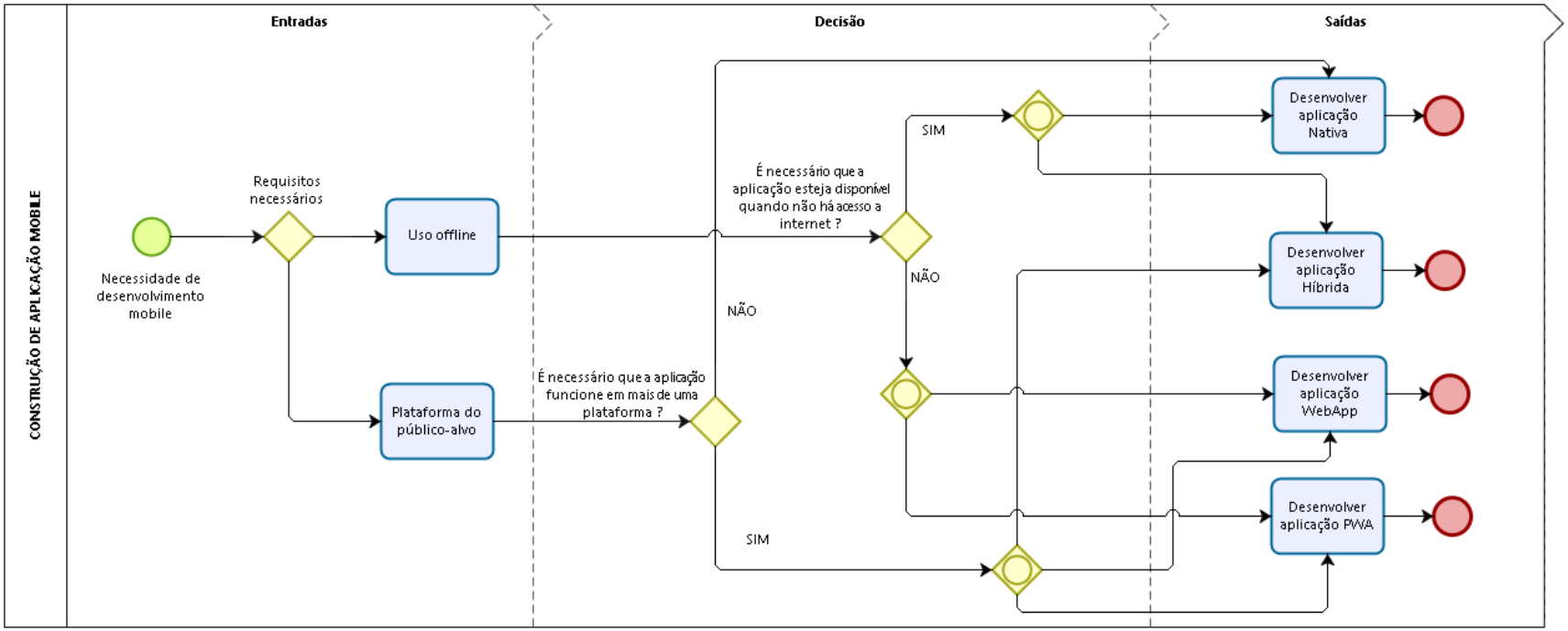

FIGURA 7 – FLUXOGRAMA DE TOMADA DE DECISÃO II – FONTE: A AUTORA, 2018.

O critério técnico do uso *off-line* da aplicação, mostra o conteúdo mesmo quando o celular estiver sem conexão com a *internet*. Exceto para aplicações web, basta que o aplicativo seja instalado no aparelho e tenha sido aberto pelo menos uma vez de maneira *online,* para que seja feita a armazenagem do conteúdo em cache. De forma que quando o aparelho estiver *off-line* o aplicativo exibirá o último conteúdo acessado.

Caso seja relevante manter a funcionalidade *off-line* no aplicativo, a melhor saída é a construção de aplicativos nativos ou híbridos, uma vez que, suportam esta prévia sincronização de dados.

Por sua vez, o segundo critério técnico abordado no fluxograma da Figura 7 é a necessidade de que a aplicação funcione em mais de uma plataforma específica. Caso o perfil traçado do público-alvo sejam usuários de diferentes sistemas operacionais, tem como saída o desenvolvimento híbrido, *web* e *PWA* pois os aplicativos podem ser acessados por usuários de qualquer SO através de um navegador e no caso de aplicativos híbridos instalados no dispositivo. Enquanto os aplicativos nativos possuem um desenvolvimento específico para cada tipo de sistema operacional.

Por fim, o fluxograma representado na Figura 8, apresenta dois critérios técnicos, sendo eles: atualizações de conteúdo e esforço inicial do usuário.

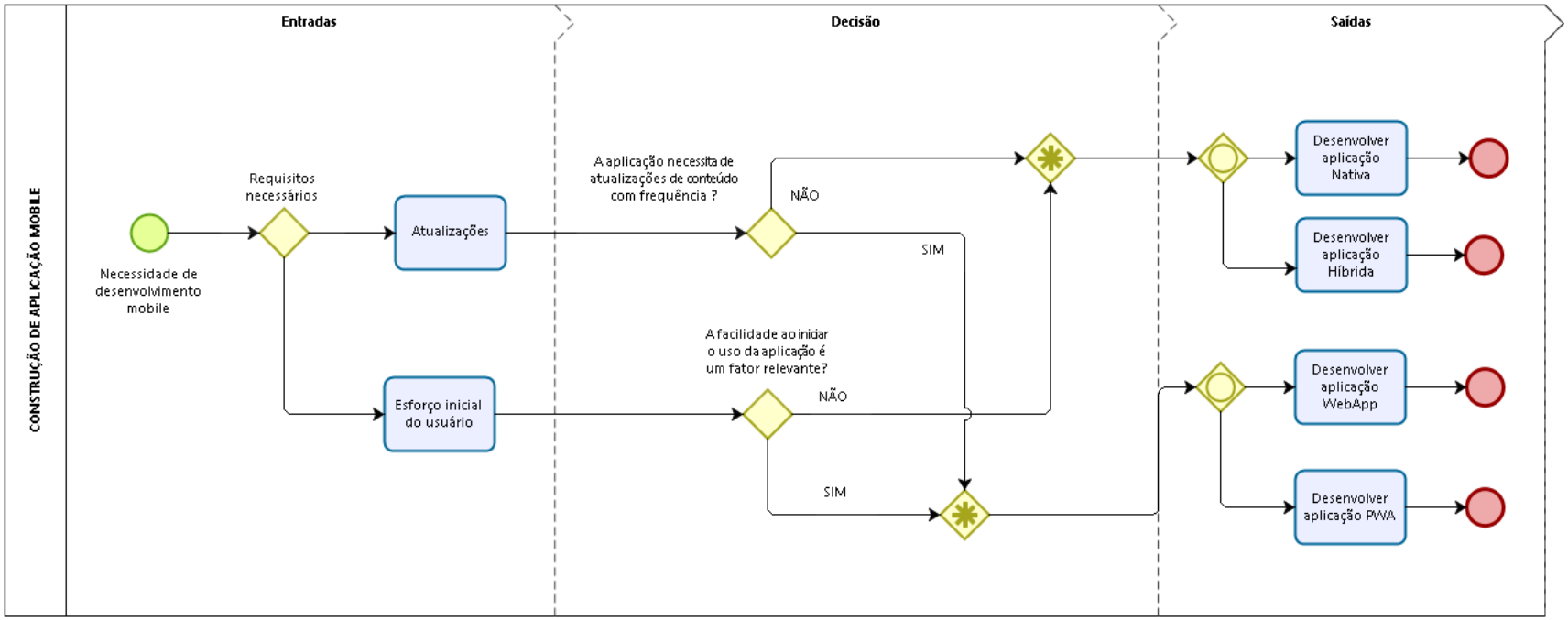

FIGURA 8 – FLUXOGRAMA DE TOMADA DE DECISÃO III – FONTE: A AUTORA, 2018.

A necessidade de manter um aplicativo com atualizações de conteúdo frequentes, tem como melhor saída de desenvolvimento as tecnologias *web* e *PWA*, pois, são mais facilmente atualizáveis. A própria equipe de desenvolvimento realiza a atualização do novo conteúdo para o servidor. Com isso, a atualização é exibida ao usuário assim que ele se conectar à *internet*.

Em relação ao segundo critério abordado referente ao esforço inicial do usuário para utilizar a aplicação, tem como melhor saída o desenvolvimento *web* e *PWA*, pois, o usuário precisa apenas digitar o endereço do aplicativo no navegador para acessá-lo não tem a necessidade de acessar a loja de aplicativos para baixa-lo.

4. CONCLUSÃO

Com este trabalho, pode-se compreender mais sobre a evolução da computação móvel por meio dos mais populares métodos de desenvolvimento atuais: desenvolvimento nativo, híbrido, *web mobile* e *PWA*.

Todas as abordagens de desenvolvimento possuem suas vantagens e desvantagens, conforme apresentado nos capítulos anteriores deste trabalho. Por isso, viu-se a necessidade de uma análise de critérios técnicos como *performance*, experiência do usuário, uso *offline*, atualizações, acesso aos recursos de *hardware*, esforço inicial do usuário e plataforma do público-alvo para decidir e escolher qual a melhor abordagem de desenvolvimento para determinado aplicativo.

Neste trabalho foram propostos três fluxogramas com intuito de auxiliar profissionais desenvolvedores a tomar decisões relacionadas a tecnologias de desenvolvimento de aplicações móveis.

O modelo de fluxo tem como entrada fatores técnicos, tem como processo questões de análise e métricas para avaliar os critérios e como saídas as alternativas tecnológicas de desenvolvimento *mobile.* Com isso, espera-se que os desenvolvedores consigam ser mais produtivos ao optar pelo método de desenvolvimento mais apropriado ao contexto da aplicação a ser desenvolvida.

Pôde-se concluir então que o desenvolvimento móvel requer uma análise aprofundada em cada um dos fatores técnicos citados no decorrer do trabalho. Para cada cenário de desenvolvimento de aplicativos existem critérios técnicos que são mais relevantes que outros.

Por isso, é essencial que os desenvolvedores tenham em mente a superioridade em determinados aspectos dentre as diferentes tecnologias de desenvolvimento para fazerem a melhor escolha no momento de criar os aplicativos. Não existe uma melhor abordagem para o desenvolvimento *mobile*, as alternativas tecnológicas são distintas e devem ser avaliadas individualmente de acordo com o escopo que o solicitante necessita.

4.1. TRABALHOS FUTUROS

Para trabalhos futuros, sugere-se uma avaliação do modelo que foi descrito baseado em critérios técnicos, questões de análises e métricas com o objetivo de avaliar a qualidade da corretude, completude, consistência e utilidade por meio da aplicação de um *survey* em alguns profissionais da área.

De acordo com o ponto de vista de responsáveis em desenvolvimento de aplicativos móveis, elaborar um questionário que avalie se:

a. Os critérios técnicos levantados e se os fluxogramas criados possuem algum erro, possuem critérios suficientes para encontrar a melhor opção,
b. O modelo é útil para empresas que precisam decidir entre as alternativas de desenvolvimento *mobile* e
c. Existem outros fatores técnicos que poderiam ser adicionados no modelo para torná-lo mais completo e auxiliar ainda mais na tomada de decisão do desenvolvimento de aplicativos *mobile*.

REFERÊNCIAS

APPLE, Inc. **About Objective-C**. Disponível em: <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>. Acesso em: 2 de Dezembro de 2017.

APPLE, Inc. **About the iOS Technologies.** Disponível em: <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>. Acesso em: 2 de Dezembro de 2017.

APPLE, Inc. **Apple Developer Program - Apple Developer**. Disponível em: <https://developer.apple.com/programs/>. Acesso em: 2 de Dezembro de 2017.

APPLE, Inc. **Developing for iOS 11 - Apple Developer.** Disponível em: <https://developer.apple.com/ios/>. Acesso em: 2 de Dezembro de 2017.

APPLE, Inc. **Submitting Your App to the Store.** Disponível em: <https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/SubmittingYourApp/SubmittingYourApp.html>. Acesso em: 2 de Dezembro de 2017.

APPLE, Inc. **Swift - Apple (BR)**. Disponível em: <http://www.apple.com/br/swift/>. Acesso em: 2 de dezembro de 2017.

ANDROID. **Meet Android Studio | Android Studio**. Disponível em: <https://developer.android.com/studio/intro/index.html>. Acesso em: 2 de Dezembro de 2017.

ANDROID. **Overview**. Disponível em: <https://source.android.com/devices/architecture/>. Acesso em: 2 de Dezembro de 2017.

ATER, T. **Building Progressive Web Apps**. Sebastopol: O'Reilly Media Inc, 2017.

BATISTA, C. A. T.; SOUZA, C. L. C.; JUNQUEIRA, R. P. ***Engenharia de Requisitos para aplicações Móveis.*** *Monografia da disciplina de Engenharia de Requisitos do Pós-graduação da Universidade Federal de Pernambuco, Brasil, 2013.*

BASILI *et. al*. **Goal/Question/Metric Approach***.* Encyclopedia of Software Engineering. John Wiley & Sons, 1994.

CARDIERI, Giulia de A., ZAINA, Luciana M.. **Analyzing User Experience in Mobile Web, Native and Progressive Web Applications: A User and HCI Specialist Perspectives**. 2018. In Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems (IHC 2018). ACM, New York, NY, USA.

CEVALLOS, Esteban Angulo. ***Case Study on Mobile Applications UX:*** **Effect of the Us age of a Cross Platform Development Framework**. 2014. 101f. Tese — Universidade Politécnica de Madrid, Madrid, jun. 2014.

CAVALCANTE, Victor. **Porque aplicativos híbridos?.** Mobile Brazil Conference, 2016. Disponível em: <https://www.youtube.com/watch?v=389Vju2Csh4>. Acesso em: 02 de Novembro de 2018.

CHAMMAS, Adriana; QUARESMA, Manuela; MONT'ALVÃO, Cláudia. **Metodologias para criação de aplicativos: uma análise com foco no design centrado no usuário**. In:14º CONGRESSO INTERNACIONAL DE ERGONOMIA E USABILIDADE, DESIGN DE INTERFACES E INTERAÇÃO HUMANO - COMPUTADOR**,** 2014, Rio de Janeiro.

CHARLAND, Andre and LEROUX, Brian. ***Mobile application development: Web vs. native.*** Communications of the ACM***.*** V. 54, p. 49–53, New York, NY, USA. 2011.

CORRAL, Luis; JANES, Andrea; REMENCIUS, Tadas. **Potential Advantages and Disadvantages of Multiplatform Development Frameworks - A Vision on Mobile Environments**. Procedia Computer Science, Itália, v. 10, p. 1202–1207, jan. 2012.

DUA, Kinjal. **Guide to Mobile App Development: Web vs. Native vs. Hybrid**. Clearbridge

Mobile. Disponível em: <https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/>. Acesso em: 02 de Novembro de 2018.

DRIFTY. ***The Ionic View App | Share your apps with the world*.** Disponível em: <http://view.ionic.io/>. Acesso em: 2 de Dezembro de 2017.

EL-KASSAS, W. S. et al. **Taxonomy of Cross-Platform Mobile Applications Development Approaches**. **Ain Shams Engineering Journal,** 2015.

FACEBOOK INC. **React - A JavaScript library for building user interfaces.** Disponível em: <https://reactjs.org/>. Acesso em: 01 de Novembro de 2018.

FINK, G.; FLATOW, I. **Pro Single Page Application Development: Using Backbone.Js and ASP.NET**. 1st. ed. Berkely, CA, USA: Apress, 2014. ISBN 1430266732, 9781430266730.

FLING, Brian. ***Mobile Design and Development*.** 1ª ed. Estados Unidos da América: O'Reilly, 2009.

FOWLER, M. **Developing Software for Multiple Mobile Devices**. Martin Fowler, 2012. Disponível em: <http://www.martinfowler.com/articles/multiMobile/>. Acesso em: 11 de Maio 2018.

GOOGLE. **Your First Progressive Web App**. 2017. Disponível em: <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/?hl=pt-br> Acesso em: 2 de Dezembro de 2017.

GOOGLE. **Angular - What is Angular?.** 2018. Disponível em: <https://angular.io/docs>. Acesso em: 15 de Novembro de 2018.

GOOGLE DEVELOPERS. **Introduction to Progressive Web App Architectures.** 2018. Disponível em: <https://developers.google.com/web/ilt/pwa/introduction-to-progressive-web-app-architectures>. Acesso em: 15 de Novembro de 2018.

GOOGLE DEVELOPERS. **Twitter Lite PWA Significantly Increases Engagement and Reduces Data Usage**. 2017. Disponível em: <https://developers.google.com/web/showcase/2017/twitter>. Acesso em: 14 de Novembro de 2018.

HEITKÖTTER, Henning; HANSCHKE, Sebastian; MAJCHRZAK, Tim. *Comparing Crossplatform Development Approaches for Mobile Applications**. In: Web information systems and technologies**.* Springer, 2013.

HUBSCH, Eduardo. **Uma Abordagem Comparativa do desenvolvimento de aplicações para dispositivo móveis**. Faculdade de Tecnologia de São Paulo, 2012, São Paulo.

IDC. ***Smartphone OS Market Share.*** Disponível em: <https://www.idc.com/promo/smartphone-market-share/os> Acesso em: 2 de Dezembro de 2017.

LINGRAS, P et al. **Building Cross-Platform Mobile and Web Apps for Engineers and Scientists: An Active Learning Approach**. Boston: Cengage Learning, 2016.

LOPES, Sérgio. **A Web *Mobile* – Programe para um mundo de muitos dispositivos**. 1ª ed. São Paulo: Casa do Código, 2013.

LOPES, Sérgio. **Aplicações Mobile Híbridas com Cordova e PhoneGap**. São Paulo: Editora Casa do Código, 2016.

MARTINS C. S, et al. **Os desafios para a mobilização de aplicações baseadas em plataforma Web**. In: X ENCONTRO ANUAL DE COMPUTAÇÃO - EnAComp 2013, Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações Campinas/SP, 2013.

MEIER, R. ***Creating Better User Experiences on Google Play | Android Developers*** Disponível em: <http://android-developers.blogspot.com.br>. Acesso em: 2 de Dezembro de 2017.

MICROSOFT. ***Windows Phone.*** Disponível em: <https://msdn.microsoft.com/pt-br/library/hh972585.aspx>. Acesso em: 2 de Dezembro de 2017.

PAPAJORGJI, P. **Automated Enterprise Systems for Maximizing Business Performance**. 1 edition. ed. Hershey, PA: IGI Global, 2015.

PREZOTTO, Ezequiel Douglas; BONIATI, Bruno Batista. **Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas**. 2014.

PELLETIER, J. **Mobile App Manual: The Blueprint**. Withinsight, 2013 ISBN: 9780989072106.

REBOUAS, M. et al. **An Empirical Study on the Usage of the Swift Programming Language**. In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)***.** [S.l.: s.n.], 2016.

RIBEIRO, R.; FREIRE, P. **Frameworks de Desenvolvimento Móvel Multiplataforma.** In: CONFERÊNCIA DA ASSOCIAÇÃO PORTUGUESA DE SISTEMAS DE INFORMAÇÃO, 2013.

RUSSELL, Alex. **Progressive Web Apps: Escaping Tabs Without Losing Our Soul**. 2015. Disponível em: <https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/> Acesso em: 2 de Dezembro de 2017.

SHEPPARD, D. **Beginning progressive web app development creating a native app experience on the web**, 2017. 978–1 p. ISBN 978-1-4842-3089-3.

SILVA, Marcelo Moro da SANTOS, PRADO Marilde Terezinha. **Os Paradigmas de Desenvolvimento de Aplicativos para Aparelhos Celulares**. Tecnologias, Infraestrutura e Software, São Carlos, v. 3, n. 2, p. 167-170, mai-ago. 2014.

SMUTNY, Pavel. **Mobile development tools and crossplatform solutions**. Carpathian Control Conference (ICCC), 2012 13th International.

SYDOW, Lexi. **Global App Store Records Shattered Yet Again in Q1**. App Annie. Disponível em: <https://www.appannie.com/en/insights/market-data/q1-2018-apps-record-downloadsspend/>. Acesso em: 20 de Setembro de 2018.

VAN SOLINGEN. R. **Agile GQM: Why Goal/Question/Metric is more Relevant than Ever and Why It helps Solving the Agility Challenges of Today's Organizations**. In: 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software process and Product Measurement (IWSM-MENSURA). IEEE, 2014, p. 271.

VISWANATHAN, Priya. **What Is a Mobile Application?**. Disponível em <https://www.lifewire.com/what-is-a-mobile-application-2373354> . Acesso em:20 de Setembro de 2018.

VUE.JS. **Introdução - Vue.js**. 2018. Disponível em <https://br.vuejs.org/v2/guide/index.html>. Acesso em 14 de Novembro de 2018.

XAMARIN. ***Xamarin University Self-Guided Learning.*** Disponível em: <https://developer.xamarin.com/> Acesso em: 2 de Dezembro de 2017.

W3C. ***Web Design &*** **Aplicações.** Disponível em: <http://www.w3c.br/Padroes/WebDesignAplicacoes>. Acesso em: 2 de Dezembro de 2017.

WAZLAWICK, Sidnei Raul. **Metodologia de Pesquisa Para Ciência da Computação.** 2ª ed. Rio de Janeiro: Elsevier, 2014.

WHITE, J. **Going native (or not): Five questions to ask mobile application developers**. The Australasian medical journal 6(1): 7, 2013.

WROBLEWSKI, Luke. ***Mobile First*****.** 1ª ed. Nova Iorque: *A Book Apart*, 2011.

YAMAKAMI, T. **Business model engineering analysis on mobile client-side software platform strategies***.* In: 7th International Conference on Mobile Business, 2008.

KEMP, Simon. **Digital in 2018: World's Internet Users Pass the 4 Billion Mark**. Disponível em <https://wearesocial.com/blog/2018/01/global-digital-report-2018>. Acesso em: 20 de Setembro de 2018.