UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

RAPHAEL BOTELHO BASSANI

ANÁLISE COMPARATIVA DO DESENVOLVIMENTO COM O USO DE FRAMEWORKS MULTIPLATAFORMA

MEDIANEIRA

2021

RAPHAEL BOTELHO BASSANI

ANÁLISE COMPARATIVA DO DESENVOLVIMENTO COM O USO DE FRAMEWORKS MULTIPLATAFORMA

COMPARATIVE ANALYSIS OF DEVELOPMENT WITH THE USAGE OF MULTIPLATFORM FRAMEWORKS

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Everton Coimbra de Araújo

Coorientador: Prof. MSc. Jorge Aikes Junior

MEDIANEIRA

2021

RAPHAEL BOTELHO BASSANI

ANÁLISE COMPARATIVA DO DESENVOLVIMENTO COM O USO DE FRAMEWORKS MULTIPLATAFORMA

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 26/novembro/2021

Everton Coimbra de Araújo
Doutor
Universidade Tecnológica Federal do Paraná

Juliano Rodrigo Lamb
Doutor
Universidade Tecnológica Federal do Paraná

Ricardo Sobjak
Doutor
Universidade Tecnológica Federal do Paraná

MEDIANEIRA

2021

AGRADECIMENTOS

Agradeço primeiramente a Deus por sempre estar comigo em todos os momentos.

A meus pais Romildo e Erica por sempre depositarem toda confiança e apoio.

A minha eterna companheira Yngrid pela paciência e suporte.

A toda minha família pelas palavras de incentivo e conselhos, vocês sabem o quão foi relevante para minha vida.

Aos meus mestres Prof. Dr. Everton Coimbra de Araujo e Prof MSc. Jorge Aikes Junior pela ajuda e orientação neste trabalho que é de extrema importância para minha vida.

Aos meus colegas “Sobreviventes” pelas experiências compartilhadas.

Aos meus professores durante toda a minha vida acadêmica, com importância em cada um dos ensinamentos.

Enfim, a todos que contribuíram com palavras amigas, sem vocês a realização deste trabalho não seria possível.

Meu MUITO OBRIGADO.

RESUMO

Com a aderência na utilização de dispositivos e aplicações que buscam facilitar tarefas cotidianas, o desenvolvimento destas aplicações tem sido motivado cada vez mais. Ferramentas foram criadas para viabilizar os processos de desenvolvimento e manutenção, buscando otimizar a ampla gama de problemas enfrentados pelos desenvolvedores, possibilitando o aumento de produtividade com o foco em um código único portável. Os *frameworks* multiplataforma suportam ambientes de execução, como dispositivos móveis, *Desktop*, *Web* e *Progressive Web App*, demonstrando as capacidades técnicas e sociais por meio do suporte da comunidade, utilização no mercado e ferramentas relacionadas aos elementos técnicos do desenvolvimento. Neste trabalho os frameworks utilizados foram: *Flutter*, *Ionic*, *React Native*, *Vue Native* e *Xamarin*, com o desenvolvimento de uma aplicação modelo em cada um deles. Esta aplicação permitiu a análise das particularidades, explorando a experiência no desenvolvimento e as vantagens da utilização dos *frameworks* multiplataforma. Assim, percebeu-se o alto grau de maturidade nos *frameworks Flutter* e *React Native*, constituindo elementos relacionados a otimização nos processos de criação e manutenção de aplicativos, com um sólido suporte ao desenvolvimento provido pela comunidade de desenvolvedores, aspectos estes que corroboram para uma experiência otimizada de desenvolvimento com o uso de *frameworks* multiplataforma.

Palavras-chave: Aplicação; Portátil; Estrutura de Software.

ABSTRACT

With the common usage of devices and applications that optimize everyday tasks, the development has growth on motivation. Within this technological area, tools were created to make easier the processes of creation and maintenance. Tools that improve the problems faced by the developers in a wide range, increasing the productivity, focusing on a single and portable code base. Those cross-platform frameworks support a varied set of platforms from mobile, Desktop, Web and Progressive Web App, demonstrating various technical and social capabilities through community support, market utilization and tools related to the technical elements on the process of development. In this paper, the frameworks used were: Flutter, Ionic, React Native, Vue Native and Xamarin, with the development of a model application in each of them. This application allowed the analysis of particularities, exploring the experience in the development and the advantages of using cross-platform frameworks. Thus, it was noticed the high degree of maturity in Flutter and React Native frameworks, constituting elements related to the optimization in the application creation and maintenance processes, with solid development support provided by the developer community, aspects that contribute to an optimized development experience with the usage of cross-platform frameworks.

Keywords: Application; Portable; Software Structure.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AOT	*Ahead-of-time*
API	Interface de Programação de Aplicações
ARM	*Advanced RISC Machine*
CLI	*Command-line interface*
CSS	*Cascading Style Sheets*
GUI	Interface gráfica do utilizador
HTML	*HyperText Markup Language*
IDE	Ambiente de Desenvolvimento Integrado
MVC	*Model-View-Controller*
MVVM	*Model-View-ViewModel*
OSS	*Open Source Software*
PWA	Aplicativo Web Progressivo
REST	*Representational State Transfer*
SDK	Kit de Desenvolvimento de Software
UI	Interface de Usuário
URL	*Uniform Resource Locator*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Uma das áreas tecnológicas mais promissoras nos recentes anos é a de desenvolvimento de aplicativos, motivada pelo fácil acesso, que demonstram o rápido crescimento nos números relacionados a compras de dispositivos, dentre eles estão os celulares, *tablets*, *wearables* e sistemas embarcados (PINTO; COUTINHO, 2018). Estes dispositivos possibilitam a leitura de *emails*, registro de eventos, sejam com fotografias ou filmagens, realização de pesquisas ou consumo de conteúdos de maneira instantânea, além de promover assistentes pessoais, que facilitam uma grande variedade de tarefas a serem concluídas (NAYEEM; WANT, 2014). Dispositivos que inicialmente serviam somente para fazer ligações se tornaram fonte computacional, conectados a informação a qualquer momento, caracterizando uma revolução tecnológica sem precedentes.

O desenvolvimento de aplicações para estes dispositivos conduziu à emergência de plataformas e ferramentas, que utilizam das vantagens de *features* nativas e suas múltiplas particularidades. Por outro lado, a diversidade dos sistemas operacionais traz a necessidade de aprendizado para todas as linguagens de programação presentes nos *Software Development Kits* (SDK). As plataformas e interfaces introduzem diferentes hábitos aos seus usuários e desenvolvedores, que tornam o trabalho de criação de uma aplicação mais desafiadora (BOSNIC *et al*., 2017). Para aumentar o alcance de usuários, um desenvolvedor deve construir sua aplicação para o maior número de plataformas disponíveis. Segundo Blom *et al*. (2008), com os diversos *frameworks* é possível expandir o número de contextos que as aplicações podem cumprir o seu papel, sem necessitar o retrabalho de desenvolvimento.

O uso de tecnologias e *frameworks* economizam recursos de custo de produção, tempo para a replicação das aplicações, além da manutenção unificada (CHARKAOUI *et al*., 2015). A otimização do desenvolvimento para uma única aplicação que funcione em mais de uma plataforma se tornou o objetivo comum dos desenvolvedores, de maneira a manter a concentração na lógica de negócio, novos produtos e melhorias para o aplicativo (SAMBASIVAN *et al*., 2011). Os *frameworks* devem seguir alguns padrões: o primeiro deles é uma linguagem de programação que possua uma baixa curva de aprendizagem, ou ser familiar, também necessitam

que o código possa funcionar em diversas plataformas, devem incluir interfaces que acessam *features* internas ao dispositivo, permitindo um rápido desenvolvimento.

Com a avaliação do desenvolvimento com o uso de *frameworks* multiplataforma, visões mercadológicas, acadêmicas, receptividade na experiência dos usuários, desempenho e todas suas subcamadas de estudos, além da visão dos próprios desenvolvedores (BIøRN-HANSEN *et al*., 2020). Portanto, este trabalho propõe uma análise comparativa entre os *frameworks* com grande popularidade e aderência da comunidade, dispondo de métricas, bem como a criação de um modelo de aplicação e a análise de desempenho sobre o desenvolvimento com o uso das ferramentas *cross-platform*.

1.1 Objetivo geral e específicos

O objetivo deste trabalho é realizar uma análise comparativa com base em métricas para o desenvolvimento de aplicativos utilizando *frameworks* multiplataformas. Este objetivo principal pode ser dividido nos seguintes objetivos específicos:

- Mapear métricas de experiência no desenvolvimento de aplicativos;
- Elaborar um modelo de aplicação para extrair as métricas junto a utilização dos *frameworks*;
- Implementar o modelo de aplicação para comparação de elementos aceitáveis de desenvolvimento;
- Analisar os resultados obtidos pelos *frameworks* que serão adotados para esta pesquisa com o modelo de aplicação.

1.2 Justificativa

Ao passar dos anos os aplicativos estão cada vez mais presentes no cotidiano, com a grande aderência de todas as camadas da sociedade, constituindo uma gama de possibilidades. Estas possibilidades passam também pelas mãos dos desenvolvedores, que necessitam se atualizar a cada nova versão de sistema, ou adição de novas tendências nas plataformas.

Com a heterogeneidade construída ao passar dos anos, no momento em que ocorre este estudo, as plataformas principais são: *Android* e *iOS* para o ambiente móvel, *Windows*, *Mac OS X* e *Linux* para o ambiente *Desktop* e o ambiente *Web* de execução (STATCOUNTER, 2021b). O desenvolvimento para estes sistemas demanda muito tempo, com cada um deles possuindo suas tecnologias, linguagens e ambientes, o que aumenta a curva de aprendizagem e torna a manutenção exclusiva para cada um dos ambientes.

Desta forma, o desenvolvimento nativo que por muitos anos foi considerado a abordagem ideal, tem perdido espaço, com novas ferramentas que instituíram uma categoria de desenvolvimento que facilita a criação, manutenção e escalabilidade (SHAH *et al*., 2019). Estas ferramentas, apesar de recentes, contam com diversos pontos a serem destacados, além da preocupação de criar uma comunidade que precisa evoluir de maneira conjunta.

Dada a problemática, este trabalho explora os pontos principais para criar uma aderência de desenvolvimento, para que o esforço exigido seja único exclusivamente para a criação de aplicativos, visando uma melhor experiência para os desenvolvedores, de maneira a aumentar a qualidade das aplicações, além de suportar uma experiência personalizada para cada uma das plataformas escolhidas e seus respectivos padrões.

1.3 Organização do documento

Este documento está organizado da seguinte forma: o Capítulo 2 apresenta o retrospecto tecnológico relacionado ao desenvolvimento multiplataforma, a experiência do usuário, bem como os ambientes de execução de aplicativos, descreve também sobre os *frameworks* para desenvolvimento multiplataforma, sendo eles: *Flutter*, *Ionic*, *React Native*, *Vue Native* e *Xamarin* e os critérios sociais relacionados. Os materiais e métodos que foram utilizados na produção deste trabalho são apresentados no Capítulo 3. No Capítulo 4 são dispostos os resultados obtidos a partir da criação do *design* da aplicação modelo e desenvolvimento utilizando os *frameworks* multiplataforma. Por fim, é descrito no Capítulo 5 a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresenta os conceitos básicos e estrutura das plataformas passíveis de desenvolvimento por meio de *frameworks* multiplataforma, com ênfase em pontos relevantes relacionados à perspectiva dos desenvolvedores. Também são apresentados aspectos referentes a experiência do usuário e a maneira em que ocorre a interação com as interfaces. Este capítulo destaca também as tecnologias que antecederam a evolução das ferramentas, assim como os *frameworks* multiplataforma e as respectivas circunstâncias sociais relacionadas.

1.4 Retrospecto tecnológico

Para desenvolver aplicações passíveis de execução nas múltiplas plataformas, diversas tecnologias foram criadas com otimizações que suportam os desenvolvedores com o processo criativo. As plataformas possuem implementações nativas, infraestruturas, arquitetura e suporte, características em *hardwares* e nos sistemas operacionais, trazendo aos desenvolvedores o desafio em prover a melhor solução para todos estes ambientes (RAJ; TOLETY, 2012).

Com o passar dos anos estas tecnologias foram evoluindo para um pensamento mais amplo, sejam elas diretamente relacionadas a uma única plataforma, ou então, focadas em aumentar a compatibilidade entre os diversos ambientes, frequentemente chamadas de aplicações multiplataforma. Os desenvolvedores precisam escolher uma tecnologia acessível para implementar as aplicações, esta tecnologia deve subsidiar flexibilidades, considerando a grande variedade de dispositivos e plataformas (BLOM *et al*., 2008).

Um dos aspectos mais abordados dentro deste âmbito é a criação de uma ferramenta que consiga trazer o reuso de código para o desenvolvimento de uma única aplicação. As aplicações produzidas podem ser categorizadas como: *web*, híbridas, interpretadas e *cross-compiled* (DELIA *et al*., 2015).

Em aplicações *web* o desenvolvimento utiliza tecnologias como *HyperText Markup Language* (HTML), *JavaScript* e *Cascading Style Sheets* (CSS). As aplicações não necessitam de instalação, pois estão expostas por meio de *Uniform*

Resource Locator (URL) podendo ser acessadas diretamente de um navegador (EL-KASSAS *et al*., 2017).

Aplicações híbridas exploram a combinação entre características nativas e *web*, de maneira performática se assemelha ao nativo, em simultâneo, propõe características *web*, além de explorar *plugins* que permitem acesso a determinadas características únicas do desenvolvimento nativo (QUE *et al*., 2017).

O desenvolvimento de aplicações interpretadas ocorre desde um projeto único que é majoritariamente traduzido para código nativo, com o restante sendo interpretado em tempo de execução (CORBALAN *et al*., 2018).

As aplicações *cross-compiled* contam com compiladores de baixo nível, que permitem a tradução da aplicação para diversas plataformas, beneficiando-se de um código único para a aplicação (PUDER, 2010).

No momento do desenvolvimento deste trabalho estas tecnologias se encontram em processo evolutivo, com aspectos que transcendem o ambiente de desenvolvimento na maneira em que os usuários experienciam as aplicações com interações.

1.5 Experiência do usuário

Com base no retrospecto tecnológico, à maneira em que o usuário interage com as aplicações sejam elas *web*, híbridas, interpretadas ou *cross-compiled* é muito importante, independentemente se a interação causa uma experiência de uso negativa ou positiva. Dentro deste contexto, cada plataforma possui o próprio estilo de como realizar as interações, definidas por uma interface específica nos guias de design (ANGULO; FERRE, 2014).

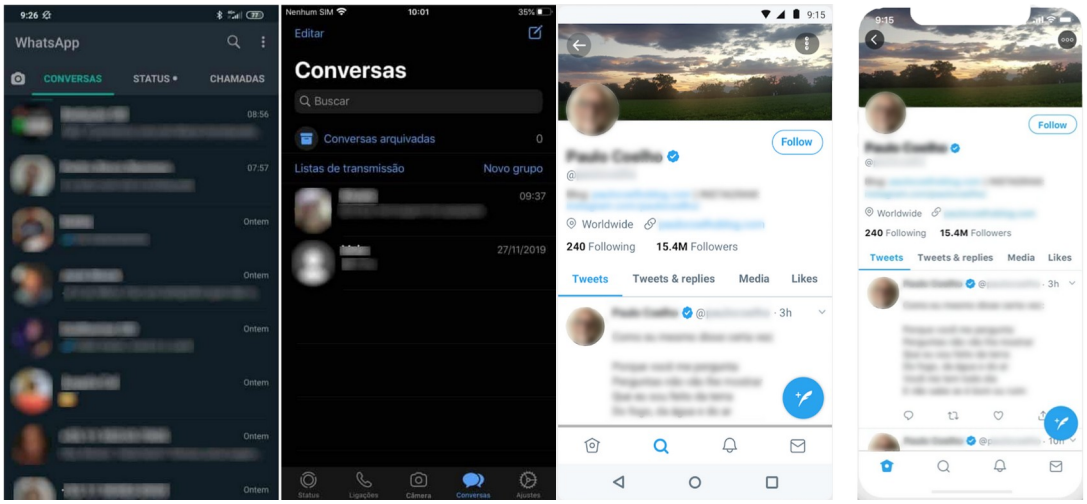
No desenvolvimento da aplicação devem ser consideradas constantes relacionadas as interações do usuário com a interface, os guias de *User Interface* (UI) e os componentes específicos de cada plataforma. Um aplicativo que segue estes conceitos consegue diminuir a curva de aprendizagem e aumenta a satisfação das experiências do usuário (BAREA *et al*., 2013).

Segundo Waljas *et al*. (2010), a transparência de experiências entre as diversas plataformas é de extrema importância para o usuário, onde mesmo sem nenhuma documentação a interação possa ser assimilada, aplicando assim o

conceito da experiência mútua. Estes são aspectos importantes para a usabilidade, utilidade, acessibilidade e confiabilidade de um aplicativo.

Alguns aplicativos famosos optam por interações diferenciadas, como o *Facebook*[1] que possui um design único para as plataformas, já o *WhatsApp*[2] possui designs distintos e existem também aplicativos que trazem uma mistura de experiência baseada em cada plataforma deterministicamente como é o caso do aplicativo do *Twitter*[3] (ANGULO; FERRE, 2014). Os aplicativos anteriores são exemplos da utilização dos sistemas de design, que podem explorar as particularidades de cada ambiente como é demonstrado na Figura 1, no qual o aplicativo do *WhatsApp* no *Android* e *iOS* são demonstrados com disparidades na experiência dos usuários dentro de uma mesma aplicação, além de termologias e *layouts* distintos. Entretanto, para o aplicativo do *Twitter* no *Android* e *iOS* a experiência foi mantida, com diferenças cosméticas dentro de uma curva de usabilidade otimizada para ambos os ambientes de execução.

Figura 1 - Comparação de aplicativos disponíveis para *Android* e *iOS*: (a) *WhatsApp* e (b) *Twitter*



(a) (b)

Fonte: Olhar Digital (2021); Twitter (2021a); Twitter (2021b)

O principal aspecto das aplicações e suas interfaces é a experiência final dos usuários, que possuem interações com cada uma delas de maneira única e complementar. Esta interação depende da responsividade da *Graphical User Interface* (GUI).

[1] https://facebook.com
[2] https://www.whatsapp.com
[3] https://twitter.com

O motivo principal pelo qual as GUI consomem grande parte do desenvolvimento e quantidade de espaço necessário para a utilização da aplicação, é refletida na importância do efeito de qualidade da aplicação (DIEHL *et al*., 2006). Uma GUI consiste em alguns aspectos principais: controles (também chamados de *widgets* ou componentes), formulários, barras de menu, layouts, interações do usuário por meio da lógica construída que será processada mediante a um resultado demonstrado, além também da responsividade (HASSAN; SARHA, 2020).

A responsividade é uma área da psicologia que estuda a interação de duas partes por meio da comunicação e por meio do apoio emocional. No âmbito computacional a responsividade está relacionada ao objetivo da interface com a interação aos seus usuários. A natureza de interação entre humanos e dispositivos é similar a interação natural entre pessoas e suas conversas (DABROWSKI; MUNSON, 2001).

As possíveis razões para problemas com responsividade são aspectos conhecidos pelos desenvolvedores, com aplicações que consomem grande poder computacional, dependência de serviços relacionados a Internet, operações de banco de dados, além da entrada e saída das interações humanas (YANG *et al*., 2013). A conexão entre percepção humana da performance de uma aplicação é mensurável pelo tempo das respostas, a falta de respostas, ou respostas inesperadas, que trazem ao usuário o sentimento de frustração, irritação e eventualmente raiva (JOVIC; HAUSWIRTH, 2011).

1.6 Ambientes de execução de aplicativos

Para possibilitar as interações com os usuários existem diversos ambientes e plataformas. O entendimento sobre os ambientes passíveis de execução dos aplicativos é essencial. De acordo com Ahti *et al*. (2016), plataformas tecnológicas criaram fragmentações, para alcançar a mesma cobertura para todos os usuários de dispositivos inteligentes, o desenvolvedor deve criar várias aplicações entre os ambientes e os sistemas operacionais. Idealmente um aplicativo deve prover a mesma experiência e funcionalidades por meio dos diferentes ambientes (ALI *et al*., 2017). Alguns diferentes ambientes são: Dispositivos Móveis, *Desktops*, *Web* e *Progressive Web Apps* (PWA).

1.6.1 Dispositivos móveis

Considerados computadores de bolso, os dispositivos móveis criaram um segmento próprio na tecnologia. Estes dispositivos podem ser executados com aplicações *stand-alone* (que não possuem dependências externas) e distribuídas entre cliente e servidor buscando informações via Internet (HOLZER; ONDRUS, 2009). Podem ser encontrados em diversos tamanhos, diversas plataformas e categorias como: *tablets*, celulares, eletrodomésticos, *wearables* (vestíveis), sistemas embarcados e automóveis.

Em vários aspectos o desenvolvimento para dispositivos móveis é similar ao praticado nas demais plataformas, problemas comuns entre elas são: integração com o *hardware*, problemas com segurança, desempenho, armazenamento e confiabilidade (WASSERMAN, 2010). Incluem também algumas características incomuns, sendo: interação potencial com outras aplicações, manuseio de sensores, *interface* e experiência de usuário, complexidade de testes, consumo de energia e compatibilidade.

Os dispositivos móveis contam com uma grande diversidade de plataformas, com diversas configurações e tamanhos. No qual o sistema operacional é o coração de cada dispositivo, que determina todos os recursos, desempenho e aplicações suportadas (LIN; YE, 2009). Existem diversos sistemas operacionais para dispositivos móveis, dentre eles *Android*[4] e *iOS*[5].

Em janeiro de 2007 a *Apple* anunciou o *iPhone* na convenção *MacWorld*, o dispositivo contava com um novo sistema operacional o *iPhoneOS*, que seria futuramente renomeado para *iOS*, baseado na plataforma de seus computadores *Mac OS X* com otimizações e suporte a novas tecnologias como o *Multi-Touch*, que possibilitava toques simultâneos na tela. Construído em uma arquitetura de camadas, com a *Apple* publicando algumas *Application Programming Interfaces* (API) para o acesso e integração dos desenvolvedores às suas aplicações (ANVAARI; JANSEN, 2010).

Para o desenvolvimento nativo da plataforma móvel da *Apple* é requirido um computador que esteja equipado com o sistema operacional *Mac OS X*, sem ser

[4] https://www.android.com/
[5] https://www.apple.com/ios/

necessário grande poder computacional. Para testes mais pontuais o simulador integrado ao sistema é de grande relevância, para testes mais precisos e maior proximidade à experiência final do desenvolvimento um dispositivo físico equipado com o *iOS* é necessário (GOADRICH; ROGERS, 2007).

A Figura 2 apresenta a estrutura arquitetural do *iOS* com às quatro camadas que constituem o sistema operacional e seus métodos de acesso, além disso, a simplicidade de pesquisar e instalar aplicativos no celular por meio da *App Store* é uma das maiores capacidades do *iPhone* (WANT, 2010).

Figura 2 - Arquitetura do sistema *iOS*

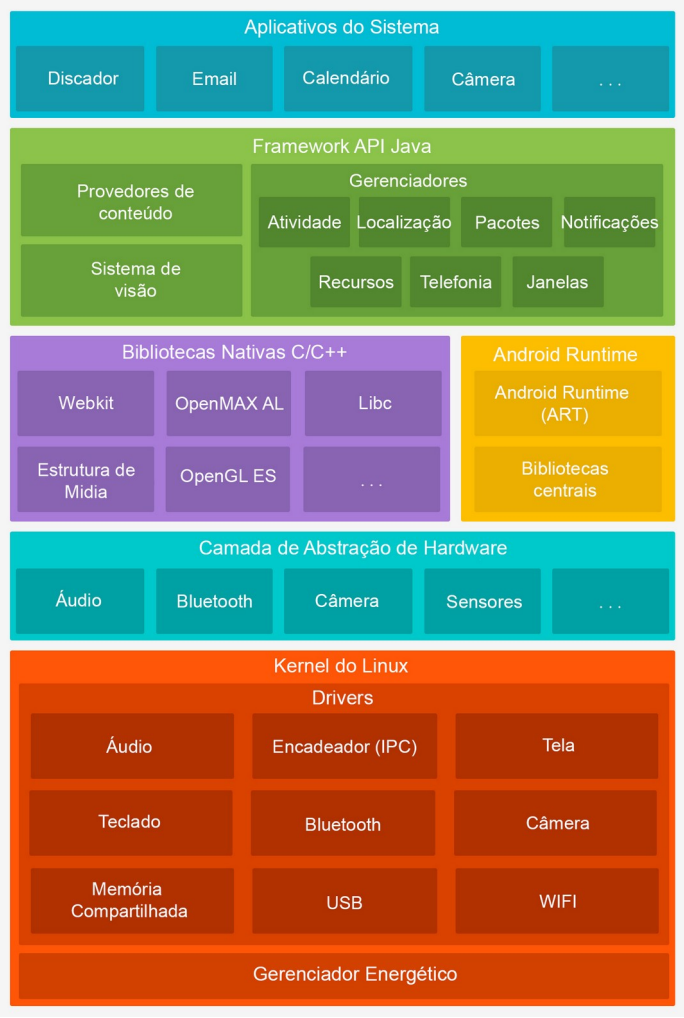
Fonte: Adaptado de Intellipaat (2021)

Em novembro de 2007, o *Google* formou um grupo de empresas chamado de *The Open Handset Alliance*. Empresas estas focadas em tecnologias para dispositivos móveis, com o objetivo de melhorar a experiência nos dispositivos (ANVAARI; JANSEN, 2010). Criaram então o sistema operacional *Android*, baseado no sistema operacional *Linux*, sendo compatível com dispositivos que possuem pequena capacidade de desempenho. O sistema operacional *Android* é de código aberto, o que permite aos vendedores e construtores de dispositivos a customização específica com interfaces únicas, sendo um vasto número de *smartphones* produzidos com a arquitetura de processadores *Advanced RISC Machine* (ARM) (REDDI *et al*., 2018).

A demonstração dos componentes da arquitetura do sistema operacional *Android* (3) é composta pelo *Kernel*, componente responsável na atuação do gerenciamento de memória, sensores e antenas no mais baixo nível. Na camada de abstração são expostas às capacidades dos componentes do *hardware*, seguida

pelas camadas de *Runtime* e de bibliotecas nativas que são responsáveis pelo gerenciamento das funcionalidades. Na camada do *Java API Framework* está presente o sistema de visualização, de gerência de notificações, de atividades e de conteúdos e na última camada estão presentes os aplicativos instalados no dispositivo (ANDROID, 2021). A plataforma *Android* distribuída pelo *Google* foi a primeira que realmente se contrapôs ao *iPhone*, não somente na visão dos usuários, mas também na dos desenvolvedores com uma loja de aplicativos de sucesso (BUTLER, 2011).

Figura 3 - Arquitetura do sistema *Android*



Fonte: Adaptado de Android (2021)

Para o desenvolvimento nativo para o *Android* os sistemas operacionais *Windows*, *Linux* e *Mac OS X* podem ser utilizados. Seu emulador supre bem as necessidades de desenvolvimento e um dispositivo físico é recomendado para melhores testes de experiência final (GOADRICH; ROGERS, 2007).

1.6.2 Ambientes Desktop

Da mesma maneira que no ambiente de dispositivos móveis, existem condições singulares para a gerência de recursos, dada a grande diversidade de plataformas e seus respectivos sistemas operacionais. Computadores pessoais podem utilizar diferentes sistemas operacionais, desenvolvidos com o intuito de gerenciar recursos disponíveis de maneira eficiente (VDOVJAK *et al*., 2020). Os níveis e a capacidade dos recursos dos computadores, o poder de processamento, armazenamento, as informações providas de redes internas e externas são alguns dos aspectos geridos pelos sistemas operacionais (GIRI; NANDGAONKAR, 2017).

Constantemente a criação de sistemas operacionais é vinculada com aplicações comerciais em desenvolvimento contínuo, além de grande suporte para aplicativos. O propósito geral dos sistemas operacionais tem evoluído no ritmo dos aperfeiçoamentos dos computadores e suas capacidades, além de acrescentar compatibilidade com plataformas mais diversificadas (ZHADCHENKO *et al*., 2020). Existem diversos aspectos a serem analisados na escolha do sistema operacional para a realização do desenvolvimento dos aplicativos, com cada sistema contando com características próprias e arquiteturas complexas.

O *Windows* é sistema operacional mais utilizado no mundo dos computadores no momento do desenvolvimento deste trabalho e possui uma vasta história (STATCOUNTER, 2021a). Criado pela *Microsoft* em parceria com a *IBM* no ano de 1981 possuiu grande atuação na evolução do modelo de um sistema operacional, sendo capaz de efetuar a gerência de processos, de memória, multitarefa, de estados e acesso (STATCOUNTER, 2021a).

Com o tempo, o *Windows* evoluiu, produzindo mudanças importantes dentro da sua arquitetura e construção, passos evolutivos que causam sempre grande relevância no mundo da tecnologia, com atualizações que acrescentam compatibilidade com uma grande gama de dispositivos (VDOVJAK *et al*., 2020). Esta relevância mostra o grande apelo do *Windows*, com grandes softwares desenvolvidos única exclusivamente para esta plataforma. Existem diversos trabalhos na literatura que demonstram as atualizações e capacidades impostas por

este grande sistema, movimentando toda a comunidade de desenvolvedores e usuários.

Lançado pela *Apple* no ano de 2001 como o sucessor do sistema *Mac OS Classic*, o sistema operacional *Mac OS X* conta com a certificação *Unix* e provê estabilidade para o kernel focado na usabilidade, é uma plataforma viável para usuários domésticos e profissionais, com grande poder de segurança (KUMAR; SURISETTY, 2012).

O *Mac OS X* é guiado pelos consumidores, principalmente entre os criadores de conteúdo como: profissionais de vídeo, relacionados a aspecto gráficos e desenvolvedores (KAKAR, 2015; MARTEAU, 2005). Este sistema operacional demonstra o grande valor para o desenvolvimento e atualizações de softwares, com diversas características únicas, poder de gerenciamento de recursos e usuários adeptos.

Apesar de existirem muitas diferenças entre os sistemas, a experiência de uso do *Mac OS X* foi evoluindo de maneira simultânea aos demais sistemas como o *Windows* e *Linux* (LOWE, 2004). Este fato evidência a importância da portabilidade entre os diversos sistemas operacionais e suas respectivas plataformas.

O sistema operacional *Linux*, conhecido por ser o maior exemplo de projeto aberto na comunidade de desenvolvedores, possui grande escala e é considerado um dos sistemas mais antigos a serem mantidos e evoluídos. Com o código aberto e milhares de desenvolvedores contribuintes, provando ser um caso de estudo de ampla proporção, com mais de 800 versões desde a primeira lançada em 1994 (ISRAELI; FEITELSON, 2010).

Um projeto caracterizado como *Open Source Software* (OSS) exemplifica grandes princípios singulares, baseiam-se no desenvolvimento com contribuições espontâneas da comunidade possibilitando uma grande troca de ideias (HESS; PAULSON, 2010). Dentro deste universo, o sistema operacional *Linux* foi criado e é mantido até o momento do desenvolvimento deste trabalho.

O *Linux* apresenta diversas distribuições e características que o transforma em um sistema operacional muito customizável. Com diversos *patches* implantados em todos os âmbitos do sistema, o *Linux* pode ser encontrado em todas as camadas da sociedade (XIAO *et al*., 2017). A influência do *Linux* na sociedade demonstra a grande importância deste sistema em conjunto com o *Windows* e *Mac OS X* para a compatibilidade das aplicações.

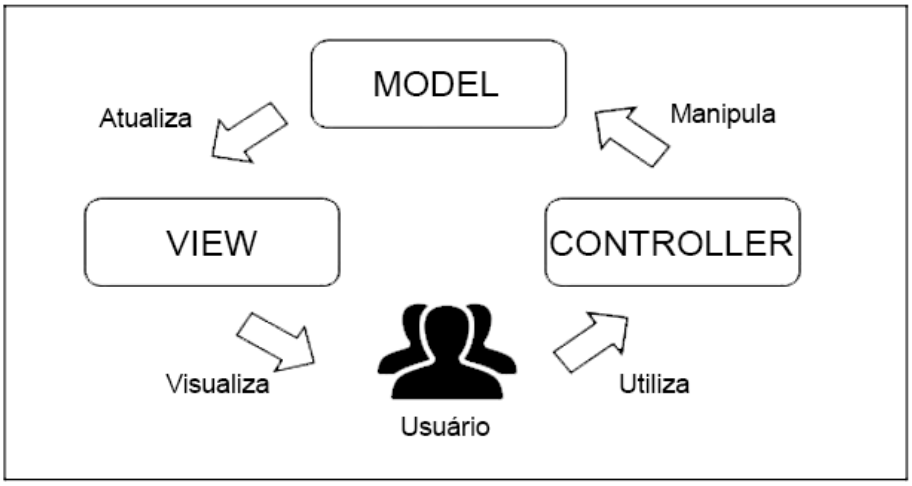
1.6.3 Ambientes Web

Na diversidade dos ambientes de desenvolvimento, como o ambiente de dispositivos móveis e ambientes *desktop*, os ambientes *web* possuem grande relevância. A evolução dos ambientes *web* se expande com a gama de serviços disponibilizados, sejam eles em domínios que vão desde empresas, entretenimento, educação e negócios (CHEN *et al*., 2010; OLSINA; ROSSI, 2002). Com a grande demanda de aplicações, serviços providos por meio da *World Wide Web* (WWW) estão sendo cada vez mais requeridos pela comunidade, de forma em que os padrões de qualidade para estas aplicações se tornaram cada vez mais sofisticados (HUANG *et al*., 2004).

Com a variedade de linguagens e ambientes para a criação de aplicações *web*, *frameworks* foram desenvolvidos com o intuito de facilitar diferentes processos. Diversos fatores e requisitos influenciam a escolha do *framework web* que será utilizado: o suporte dos desenvolvedores da ferramenta, critérios sociais da comunidade de desenvolvedores, facilidade na manutenção, possibilidade de reuso de componentes, padronização de projetos e a otimização na diminuição de erros (COSTA *et al*., 2018; GIZAS *et al*., 2012).

Uma das características mais exploradas pelos *frameworks* no desenvolvimento de aplicações *web* é a possibilidade de utilizar padrões arquiteturais, como o *Model-View-Controller* (MVC), que organiza o projeto em três camadas: Modelos, Visualizações e Controladores respectivamente (COSTA *et al*., 2018; HARIS; HASIM, 2019). O MVC pode ser encontrado em muitos projetos que utilizam dos *frameworks web* e *frameworks* multiplataforma. A Figura 4 exemplifica o uso do padrão MVC com a camada *controller* sendo utilizada pelos usuários. A camada de *controller* então manipula a camada *model*, a camada *model* atualiza a camada de *view* que mostra o conteúdo para os usuários.

Figura 4 - Diagrama do padrão de projeto *Model-View-Controller*



Fonte: Adaptado de Haris e Hasim (2019)

A importância dos ambientes *web* de desenvolvimento transcende entre as diversas plataformas, com a possibilidade da utilização de aplicações *web* em todos os dispositivos que possuem navegadores, um requisito mínimo para dispositivos conectados a Internet (PERCHAT *et al*., 2013).

1.6.4 Ambiente PWA

O uso dos ambientes *web* traz diversos benefícios no desenvolvimento de aplicações, podendo ser utilizado em diversas plataformas, com esforços focados em um único código base e implantação simplificada. Porém, existem algumas capacidades inferiores das aplicações *web* comparadas a outras metodologias de desenvolvimento, limitações no uso de recursos do *hardware* dos dispositivos, a ausência das aplicações nas lojas de aplicativos e a impossibilidade de instalação local. Estas limitações mantiveram o desenvolvimento multiplataforma longe dos ambientes *web* (PUDER *et al*., 2014).

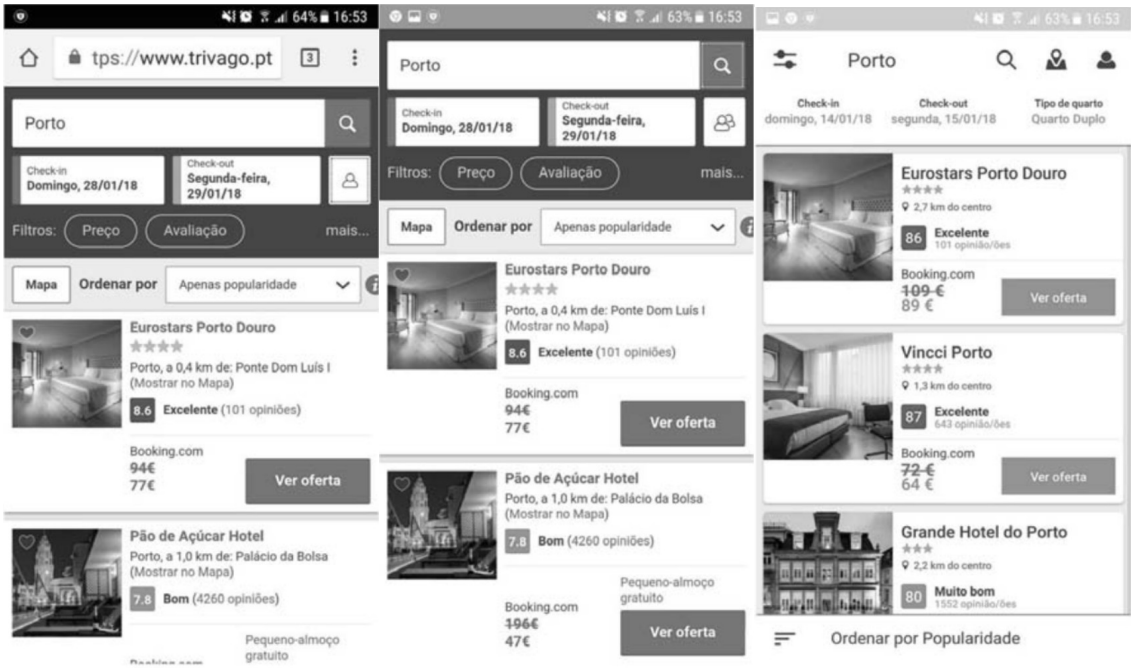
Com a demanda de um ambiente que conseguisse igualar capacidades como: suporte a aplicações desconectadas à rede Internet, carregamento de informações no *background*, instalação local e uma performance melhorada, foi criado então o ambiente dos *Progressive Web Apps* (BIøRN-HANSEN *et al*., 2017).

O termo PWA foi criado por Russel e Berriman (2015) em um *blog* após algumas ideias iniciais do conceito. As PWAs são: progressivas, responsivas, independem de conexões, são facilmente atualizáveis, contam com grande

capacidade de segurança, são acessíveis, fortemente engajáveis, possuem instalação simplificada e são facilmente compartilháveis (OSMANI, 2021).

Empresas multinacionais como: *Twitter*, *Trivago*[6], *Tencent News*[7] e *Sina Weibo*[8] apostam nas PWAs, além da presença do suporte nativo em diversos navegadores como: *Google Chrome*, *Mozilla Firefox*, *Microsoft Edge*, *Opera*, *Samsung Internet* e *Safari* (STEINER, 2018). A Figura 5 apresenta as aplicações do *Trivago* nos diferentes ambientes, com a aplicação *web*, PWA e aplicação nativa respectivamente. Esta figura demonstra a evolução disposta pela tecnologia das PWAs, possibilitando uma maior versatilidade com uma experiência otimizada para os usuários mais próximo do que é disposto por aplicações nativas em comparação com o *website*.

Figura 5 - Comparativo entre (a) aplicação *Web*, (b) aplicação PWA e (c) aplicação Nativa



(a) (b) (c)

Fonte: Fortunado (2018)

O desenvolvimento de aplicações nos ambientes das PWAs aumentam a compatibilidade com diversas plataformas, possibilitando otimizações para as aplicações *web*, além de quebrar fronteiras com as demais metodologias de desenvolvimento multiplataforma (LEE *et al*., 2018).

[6] https://www.trivago.com
[7] https://www.tencent.com/en-us/media/news.html
[8] https://weibo.com

1.7 Frameworks para Desenvolvimento Multiplataforma

Conforme apontado anteriormente na Seção 2.3, são diversos os ambientes de execução de aplicativos e seus respectivos sistemas operacionais. A utilização de ferramentas de desenvolvimento que tornem possível a compatibilidade com estes ambientes é necessária, pois a gama de tecnologias e linguagens de programação para o desenvolvimento de aplicações para cada um dos ambientes é extensa (PERCHAT *et al*., 2013).

O uso de *frameworks* multiplataforma possibilitam à compatibilidade das aplicações em vários ambientes de execução, permitindo aos desenvolvedores a capitalização das vantagens especificas a esta metodologia, a redução no tempo consumido para o desenvolvimento, redução de custos e reutilização de artefatos são algumas das vantagens do uso dos *frameworks* multiplataforma (HEITKOTTER *et al*., 2013; LATIF *et al*., 2017). Estes *frameworks* contam com um suporte em mais de dois ambientes de execução, além de uma comunidade solida como: *Flutter*[9], *Ionic*[10], *React Native*[11], *Vue Native*[12] e Xamarin[13].

1.7.1 Flutter

Flutter é um *framework* multiplataforma de código aberto criado pelo *Google* em 2017, que possibilita o desenvolvimento de aplicações de código único compiladas nativamente para sistemas operacionais como: *Android*, *iOS*, *Windows*, *Mac OS X*, *Linux*, *Web* e sistemas embarcados (FLUTTER, 2021; WASILEWSKI; ZABIEROWSKI, 2021). De acordo à Szczepanik (2020), o *Flutter* provê grande eficiência de segurança e performance, sendo capaz de produzir resultados melhores que aplicações nativas utilizando uma arquitetura em camadas.

Com lançamento oficial em quatro de dezembro de 2018, o *framework Flutter* possui uma nova filosofia para *frameworks* multiplataforma, trazendo aos desenvolvedores a possibilidade de escrever códigos nativos para os ambientes

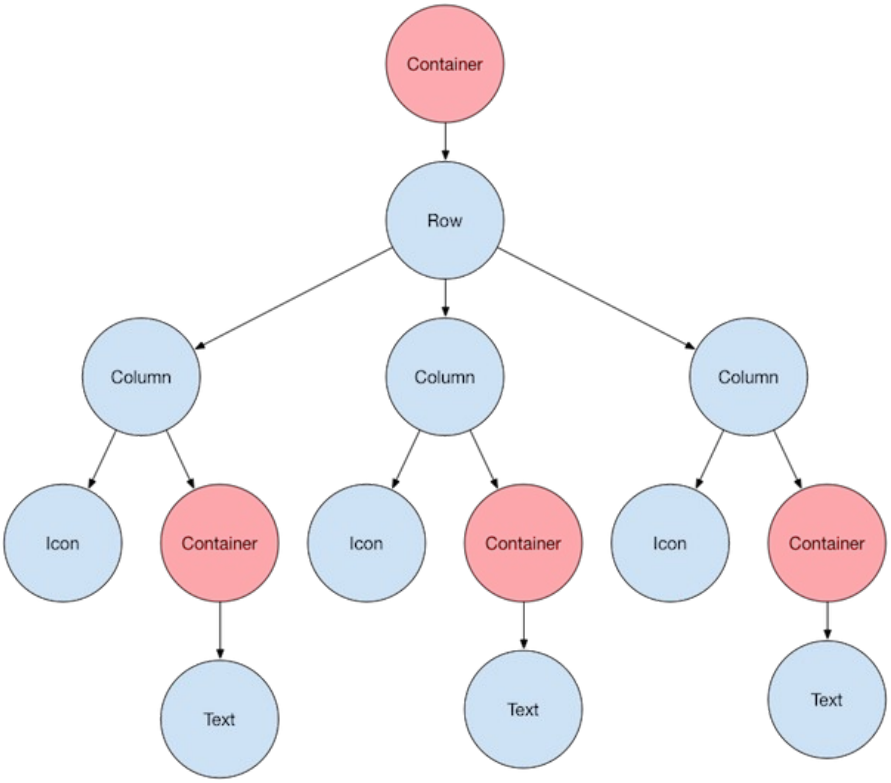
[9] https://flutter.dev
[10] https://ionicframework.com
[11] https://reactnative.dev
[12] https://vue-native.io
[13] https://dotnet.microsoft.com/apps/xamarin

suportados, o que remove limitações específicas das funcionalidades nativas de cada ambiente (BOUKHARY; COLMENARES, 2019).

O *Flutter* utiliza-se de *Dart* que é uma linguagem de programação do paradigma orientado a objetos, definida por classes. O *Dart* conta com ferramentas de *Garbage Collector* e compilação por meio da tecnologia *Ahead-of-time* (AOT), compilação esta que ocorre antes da execução da aplicação (BOUKHARY; COLMENARES, 2019). *Dart* foi criado com o intuito de trazer uma experiência otimizada para o desenvolvimento de aplicativos com uma compilação incremental. A linguagem também conta com o *hot reload* que é uma ferramenta de recarregamento instantâneo dos programas em tempo de projeto (DART, 2021).

O *framework Flutter* utiliza-se também do motor gráfico *Skia 2D* que permite recriar experiências nativas por meio do *Skia Canvas* (BIøRN-HANSEN *et al*., 2020). O *framework Flutter* baseia-se no desenvolvimento com o uso de *widgets*, onde a aplicação é um *widget* composto por outros *widgets* em uma estrutura de árvore (MAHENDRA; ANGGOROJATI, 2020). Na Figura 6 é demonstrado como a estrutura de uma árvore de *widgets* é construída no desenvolvimento das interfaces no *Flutter*.

Figura 6 - Estrutura da árvore de *widgets*



Fonte: Flutter (2021)

1.7.2 Ionic

Framework multiplataforma criado em 2013 pela empresa *Drifty Co*, o *Ionic* utiliza de outros *frameworks* como: *Angular.js*[14], *React*[15] e *Vue.js*[16] de maneira a facilitar tarefas e potencializar o desenvolvimento multiplataforma (BOSNIC *et al*., 2017). O *Ionic* possui como foco principal a aparência da UI da aplicação, com artefatos nativos que enriquecem a experiência dos usuários (WILLOCX *et al*., 2016).

O *Ionic* utiliza-se da tecnologia híbrida para a construção das aplicações (BIøRN-HANSEN *et al*., 2020; ZHOU *et al*., 2020). Segundo Rieger e Majchrzak (2019) o *Ionic* simplifica aspectos no desenvolvimento como o uso de componentes modulares entre diversos projetos, podendo ser extensível por bibliotecas, além de possibilitar o acesso a recursos nativos dos dispositivos por meio do *HTML5*.

[14] https://angularjs.org
[15] https://reactjs.org
[16] https://vuejs.org

Os ambientes suportados nativamente pelo *Ionic* são: *Android*, *iOS* e PWA. Para a compatibilidade com ambientes *Desktop* o *Ionic* faz o uso do *framework Electron*[17] (BIøRN-HANSEN *et al*., 2019; IONIC, 2021). A experiência com o *Ionic* se assemelha com o desenvolvimento *web*, onde as ações de execução podem se resumir em um arquivo de texto (BRITO *et al*., 2018).

A linguagem utilizada pelo *Ionic* é o *JavaScript*, uma linguagem multi-paradigma, interpretada e orientada a objetos, que emprega tecnologias como o *HTML* e *CSS* para personalizações customizadas na interface do usuário (JAVASCRIPT, 2021). O *JavaScript* possibilita o uso de modelos de projeto como MVC e *Model-view-viewmodel* (MVVM) que facilitam no desenvolvimento e posteriores manutenções (WILLOCX *et al*., 2016).

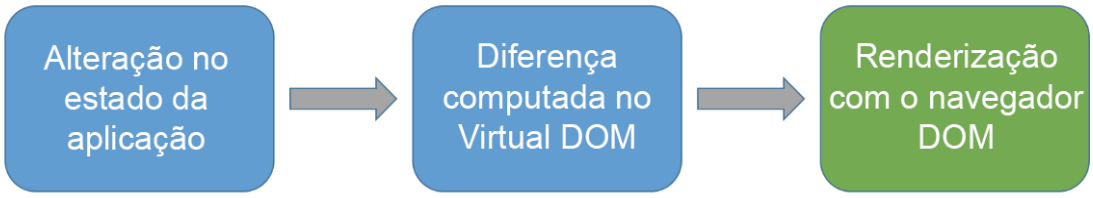
1.7.3 React Native

Com boa capacidade performática e ferramentas para o desenvolvimento de interfaces de usuários, o *React Native* é um *framework* multiplataforma de código aberto criado pelo *Facebook* em abril de 2015 (ZHOU *et al*., 2020). O *React Native* é baseado no *framework React* e segue o conceito de métodos declarativos, com o uso da linguagem de programação *JavaScript*, que traz níveis de abstração para as aplicações possibilitando um desenvolvimento simplificado (SHAH *et al*., 2019).

O *React Native* utiliza de tecnologias como o navegador *Virtual DOM* que viabiliza a gerência dos dados em memória, otimizando o processo de atualização que acontece somente nas porções que ocorreram as alterações (WAREN, 2016). Na Figura 7 é demonstrado a maneira em que o processo de otimização utilizando o *Virtual DOM* ocorre.

[17] https://www.electronjs.org

Figura 7 - Utilização do *Virtual DOM* presente no *React Native*



Fonte: Adaptado de Warén (2016)

A combinação entre interfaces renderizadas nativamente e a lógica de negócio da linguagem *JavaScript* em tempo de execução, traz ao *React Native* a possibilidade de customizar componentes padronizados especificamente para cada ambiente (BIøRN-HANSEN *et al*., 2020). As plataformas suportadas pelo *React Native* estão presentes no ambiente de dispositivos móveis com os sistemas *Android* e *iOS* (REACTNATIVE, 2021).

Com grande aderência à arquitetura de centralização de componentes o *React Native* possibilita uma boa manutenibilidade, com uma abordagem flexiva em tempo de execução de aplicativos (RIEGER; MAJCHRZAK, 2019). O *React Native* se destaca pela sua simplicidade e efetividade no processo de desenvolvimento de aplicações multiplataforma (WU, 2018).

1.7.4 Vue Native

Vue Native é um *framework* multiplataforma em desenvolvimento que tem como base de sua estrutura o *framework Vue.js*. O *Vue.js* traz inspirações e melhorias nas virtudes e problemáticas de *frameworks* como *React* e *Angular.js* (FORTUNATO; BERNARDINO, 2018). Utiliza-se do padrão MVVM, dirigido por componentes, com projetos de desenvolvimento incrementais de baixo para cima (*bottom-up*) (LI; ZHANG, 2021).

O *Vue Native* utiliza da linguagem de programação *JavaScript* e foca na construção de componentes com a separação da lógica, técnica conhecida como separação de preocupações (*separation of concerns*), possibilitando assim o reuso destes componentes em diversos projetos (ALTAMIMI *et al*., 2020).

Nativamente o *Vue Native* possui compatibilidade com sistemas do ambiente de execução em dispositivos móveis (*Android* e *iOS*), focado na camada de

visualização, ou seja, seu objetivo é trazer interfaces otimizadas para as aplicações (VUENATIVE, 2021).

O *Vue Native* possibilita a integração de ecossistemas como o *Vue.js* e o *React Native*, possui também uma curva de aprendizagem baixa para desenvolvedores habituados a tecnologias como *HTML* e *JavaScript*. Apesar de ser um *framework* em desenvolvimento o *Vue Native* possibilita fácil integração entre componentes e bibliotecas, podendo ser compartilhadas entre diversos projetos.

1.7.5 Xamarin

Criado em maio de 2011 e adquirido pela *Microsoft* em fevereiro de 2016, *Xamarin* é um *framework* multiplataforma construído sobre o *framework Mono*[18] que foi desenvolvido a partir da plataforma *.NET*[19]. Esta ampla retrocompatibilidade permite ao *Xamarin* extrair diversas características positivas das demais tecnologias (BOUSHEHRINEJADMORADI *et al*., 2016).

A linguagem de programação utilizada pelo *Xamarin* é o *C#*, linguagem que utiliza do paradigma orientado a objetos e paradigma orientado a componentes. Com uma abordagem de linguagem natural, o *C#* possui ferramentas como: *Garbage Collector*, ótima estrutura de tratamento de erros e gerência de categorias de variáveis com proteção de nulos (MICROSOFT, 2021d).

Aplicações desenvolvidas com o *Xamarin* possuem ótimos resultados performáticos, possibilitando comparações com aplicações desenvolvidas de forma nativa (BIøRN-HANSEN *et al*., 2019). O *Xamarin* possui compatibilidade com ambientes de dispositivos móveis e ambientes *Desktop*. Os sistemas operacionais suportados são: *Android*, *iOS*, *Mac OS X* e *Windows* (DELIA *et al*., 2015).

Existem discussões para um consenso de qual abordagem o *Xamarin* se enquadra. Segundo El-Kassas *et al*. (2017), o *Xamarin* é um *framework* que possui um conjunto de condutas semelhante ao *Titanium Appcelerator*[20] que utiliza da metodologia interpretada multiplataforma. Entretanto, Que *et al*. (2017) e Willocx *et al*. (2015) reivindicam que a melhor definição em que o *Xamarin* se encaixa é na

[18] https://www.mono-project.com
[19] https://dotnet.microsoft.com
[20] https://www.appcelerator.com

metodologia *cross-compiled*, parecido com a *framework Qt*[21], utilizando de um único código que é compilado nativamente para diferentes ambientes.

O uso da abordagem *cross-compiled* traz para o *Xamarin* uma característica de segregação de ferramentas que realizam a compilação de cada plataforma separadamente. Esta problemática é solucionada com o uso da extensão *Xamarin.Forms*[22] que é uma ferramenta que compila um código único para *Android*, *iOS* e *Windows* (MICROSOFT, 2021c).

1.8 Critérios sociais

Com a variedade de *frameworks* multiplataforma, cabe aos desenvolvedores a decisão de qual é a mais apropriada para os projetos, com as singularidades positivas e negativas de cada uma das ferramentas. Neste contexto, alguns elementos são considerados de extrema relevância, como é o caso dos critérios sociais relacionados, que vão desde a comunidade de desenvolvedores dos *frameworks*, a utilização no mercado tecnológico, a qualidade da documentação e quão constantes são as atualizações e melhorias (IVANOVA; GEORGIEV, 2019).

1.8.1 Comunidade de desenvolvedores

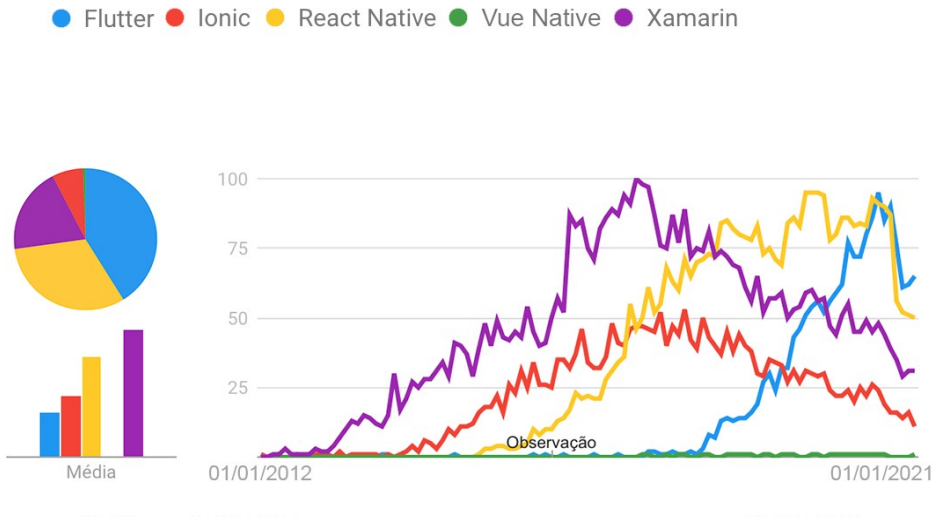
Para atingir um patamar de sucesso as tendências tecnologias dependem de diversos aspectos. Um destes aspectos são os critérios de atividades sociais desenvolvidas pela comunidade de desenvolvedores, sendo de extrema importância para mensurar o grau de popularidade e utilização dos *frameworks*, que por natureza envolvem pessoas de diferentes cargos e áreas de atuação (TAMBURRI *et al*., 2016).

A comunidade acadêmica têm depositado diversos esforços no estudo do entendimento das características sociais que exercem influência no desenvolvimento da popularidade dos *frameworks* multiplataforma. A interação constante destes grupos sociais traz diversos benefícios, como é o caso da rapidez na resolução de problemas, possibilitando efeitos de diminuição de esforços e custos (PALOMBA *et al*., 2018).

[21] https://www.qt.io
[22] https://dotnet.microsoft.com/apps/xamarin/xamarin-forms

Existem diversas maneiras de medir a popularidade dos *frameworks* multiplataforma, como é o caso da ferramenta *GoogleTrends*[23], que permite explorar quais são os temas mais pesquisados no buscador do *Google*. A Figura 8 ilustra os níveis de interesse em buscas que envolvem os *frameworks* multiplataforma no intervalo de 2012 ao momento desta pesquisa, com a ascensão do *Xamarin* à alguns anos e os *frameworks Flutter* e *React Native* mais consolidados entre os anos de 2020 e 2021.

Figura 8 - Interesse de pesquisas para os *frameworks* multiplataforma

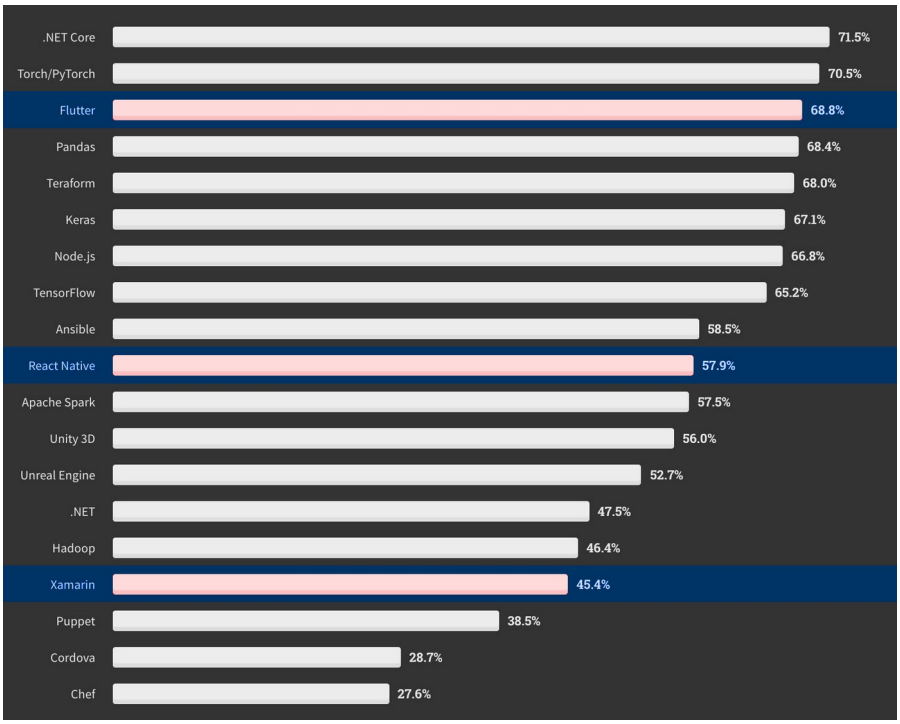


Fonte: Adaptado de Google (2021c)

Outra maneira de mensurar a popularidade dos *frameworks* multiplataforma é a atividade relacionada em sites como o *Stack Overflow*[24]. Estes sites se destacam por possuir uma grande comunidade com milhões de usuários ativos e diversos fóruns tecnológicos (HUANG *et al*., 2016). Neste contexto, o *Stack Overflow* disponibiliza pesquisas anuais a respeito da popularidade das ferramentas na comunidade. A Figura 9 apresenta o resultado de uma destas pesquisas, demonstrando a grande popularidade dos *frameworks* multiplataforma como: *Flutter*, *React Native* e *Xamarin*, o que descreve a popularidade no contexto de ferramentas para desenvolvimento multiplataforma de aplicações.

[23] https://trends.google.com.br/
[24] https://stackoverflow.com

Figura 9 - *Frameworks* mais populares na comunidade do *Stack Overflow*



Fonte: Adaptado de Stack Overflow (2020)

1.8.2 Utilização no mercado

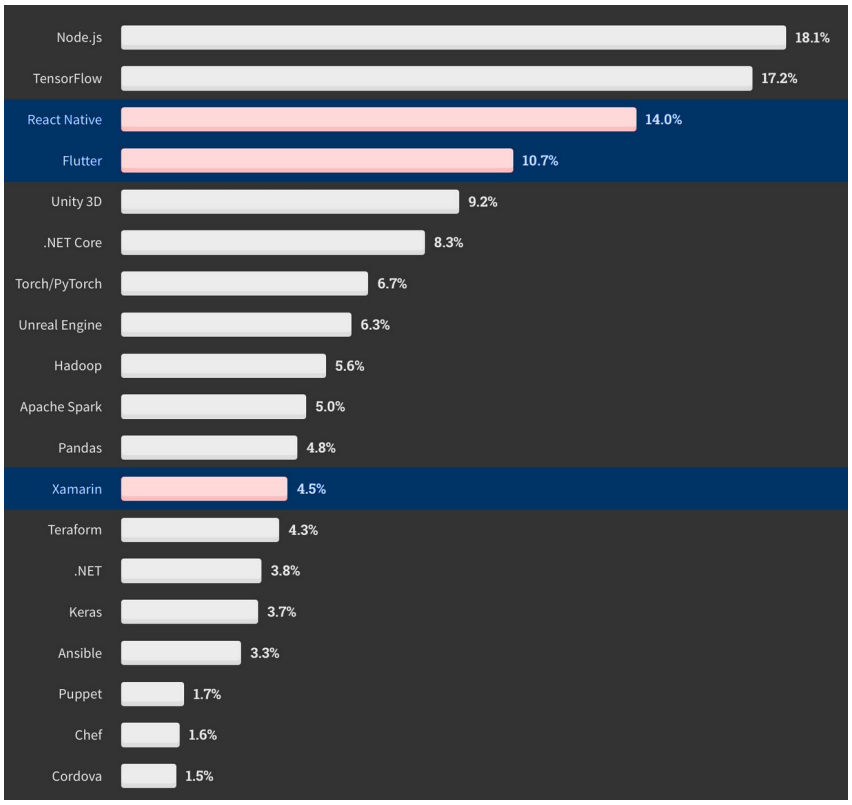
Com a grande popularidade na comunidade dos desenvolvedores, grandes empresas transparecem confiança para definirem o uso dos *frameworks* multiplataforma, apoiados por tópicos como: redução de custos, padronização de versionamentos entre os ambientes e otimização de resultados (CHEN *et al*., 2019). Estes fatos fomentam as adequações dos profissionais para desenvolverem qualidades técnicas de maneira a suprir a demanda das oportunidades.

O uso dos *frameworks* multiplataforma está presente em diversas organizações como:

- *Flutter*: *BMW*, *Nubank* e *Tecent* (FLUTTER, 2021);
- *Ionic*: *BBC*, *Burger King* e *EA Games* (IONIC, 2021);
- *React Native*: *Discord*, *Pinterest* e *Tesla* (REACTNATIVE, 2021);
- *Vue Native*: *FinTech Consortium*, *TurkDevOps* e *SpickedMelon* (VUENATIVE, 2021);
- *Xamarin*: *BBVA*, *Olo* e *Outback Steakhouse* (MICROSOFT, 2021f).

Na pesquisa realizada pelo *Stack Overflow* também foram demonstradas quais são as tecnologias e os *frameworks* mais procurados pela comunidade, o que deduz a grande utilização destes *frameworks* pelas organizações. Uma destas pesquisas é demonstrada na Figura 10, com a presença dos *frameworks* multiplataforma *Flutter*, *React Native* e *Xamarin*.

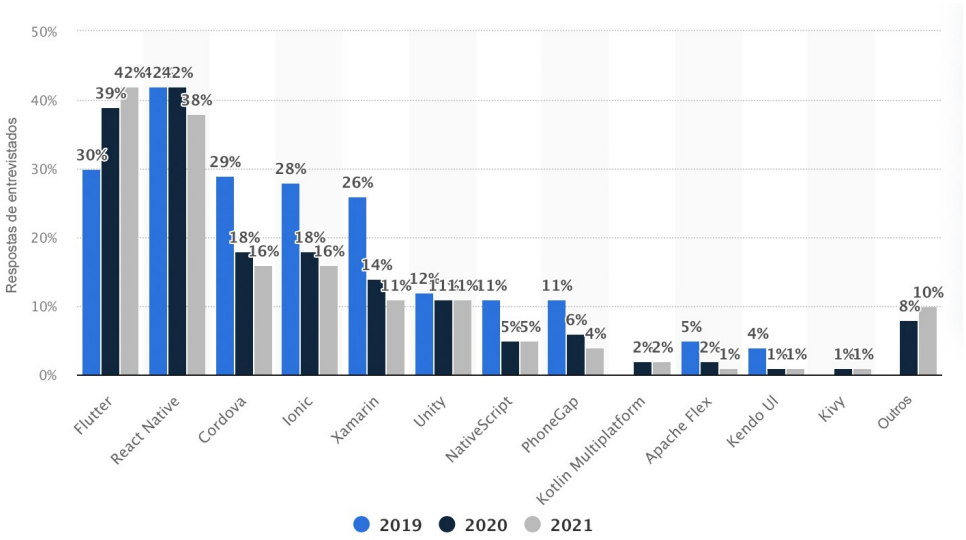
Figura 10 - *Frameworks mais procuradas pela comunidade do Stack Overflow*



Fonte: Adaptado de Stack Overflow (2020)

Entre os anos de 2019 e 2020 houve um crescimento no número de desenvolvedores em cerca de 600 mil novos profissionais (LIU, 2020). Com este crescimento no número de profissionais, o aumento no uso dos *frameworks* multiplataforma tem sido notável. A Figura 11 demonstra uma pesquisa de quais são os *frameworks* multiplataforma utilizados mundialmente para o período de 2019 a 2021.

Figura 11 - *Frameworks* multiplataforma mais utilizados de 2019 até 2021



Fonte: Adaptado de Liu (2021)

1.8.3 Documentação

Com a ampla utilização das organizações, as comunidades estão expandindo o número de usuários ativos, como é o caso do *GitHub*[25], uma ferramenta de hospedagem de códigos e repositórios que conta com mais de 56 milhões de desenvolvedores e cerca de dois bilhões de contribuições (GITHUB, 2020).

Ferramentas como o *GitHub* demonstram a grande participação da comunidade ao contribuir e documentar os projetos. São diversos os *frameworks* que utilizam desta ferramenta, como é o caso dos repositórios do *Flutter*[26], *Ionic*[27], *React Native*[28], *Vue Native*[29] e *Xamarin*[30].

A documentação é uma das práticas mais recomendadas para o desenvolvimento de projetos, o que facilita a compreensão dos aspectos relacionados, resultando no aumento do número de contribuições e na popularidade dos repositórios. Arquivos como: *ReadMe* (leia-me) e de licença de utilização acrescentam diversas informações de *guidelines* (diretrizes), instalação e usabilidade (VENIGALLA; CHIMALAKONDA, 2021).

[25] https://github.com
[26] https://github.com/flutter/flutter
[27] https://github.com/ionic-team/ionic-framework
[28] https://github.com/facebook/react-native
[29] https://github.com/GeekyAnts/vue-native-core
[30] https://github.com/xamarin

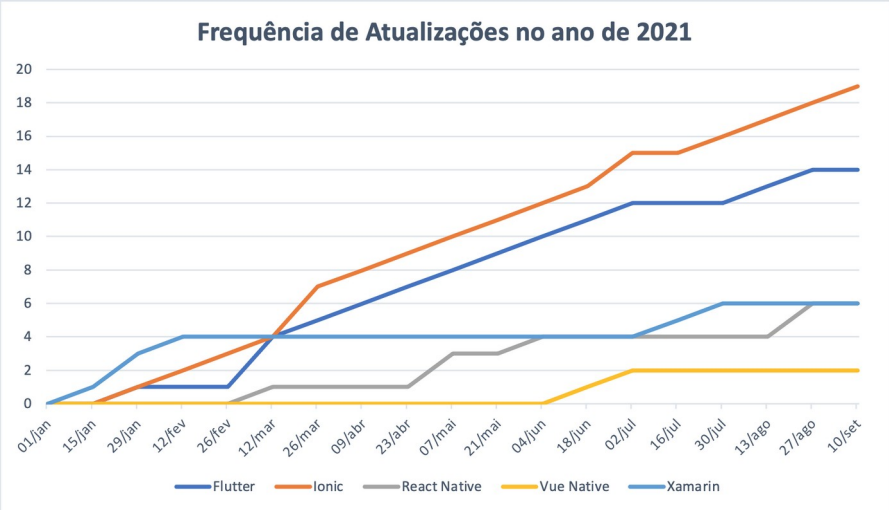
Documentações técnicas descrevem funcionalidades e os respectivos padrões de utilização. Segundo Aghajani *et al*. (2019), as documentações facilitam os processos de desenvolvimento, sanando dúvidas frequentes e explicando conceitos técnicos de funcionamento. Acrescentando benefícios para a comunidade que utiliza as ferramentas que possuem estas documentações, como é o caso dos *frameworks* multiplataforma citados.

1.8.4 Constância de atualizações

Com uma comunidade de desenvolvedores, ampla utilização no mercado e boas documentações, os critérios sociais a respeito dos *frameworks* multiplataforma contam também com aspectos de modernização como: novas funcionalidades adicionadas, revisões de características existentes e a periodicidade em que as atualizações ocorrem (SHAH *et al*., 2019). Estes aspectos de modernização podem ser considerados características essenciais para a escolha da utilização das ferramentas (VITOLS *et al*., 2013).

O histórico de atualizações para os *frameworks* multiplataforma (*Flutter*, *React Native*, *Ionic*, *Vue Native* e *Xamarin*) é vasto, melhorando e acrescentando novas funcionalidades. A constância de atualizações verificadas nos repositórios dos *frameworks* multiplataforma no momento do desenvolvimento deste trabalho, mostram grande representatividade dos aspectos de modernização. Na Figura 12 são demonstradas as atualizações e as datas para cada um dos respectivos *frameworks* multiplataforma.

Figura 12 - Frequência de atualizações dos *frameworks* multiplataforma



Fonte: Autoria própria

A ampla quantidade de atualizações demonstra o zelo da comunidade responsável pelo desenvolvimento e mantimento dos *frameworks*, com uma boa frequência de lançamentos durante o ano de desenvolvimento deste trabalho (2021).

Com o retrospecto de tecnologias antecedentes, motivados pelos elementos relacionados as experiências dos usuários e com os ambientes de execução de aplicativos, os *frameworks* multiplataforma obtiveram um papel notável na comunidade dos desenvolvedores. No próximo capítulo serão descritos os materiais e métodos utilizados para o desenvolvimento deste projeto.

3 MATERIAIS E MÉTODOS

Neste capítulo são descritas as etapas do projeto e as principais tecnologias utilizadas. Diversas tecnologias foram aplicadas para explorar os frameworks multiplataforma, descritas na Seção 3.1. Os métodos utilizados para aplicação dos frameworks multiplataforma são descritos na Seção 3.2.

1.9 Materiais

Nesta seção são apresentados os materiais e as tecnologias a serem utilizadas na execução deste trabalho.

1.9.1 Hardware utilizado

O computador utilizado para o desenvolvimento das aplicações é um notebook *Apple Macbook Pro* equipado com um processador *Intel Core I7-4980HQ* com quatro núcleos de processamento e *clock* de 2.8GHz, 16GB de memória RAM, placa de vídeo *AMD Radeon R9 M370X* com 2GB de memória dedicada e SSD de 512GB. Esta configuração proporciona a extração de todas as possibilidades do desenvolvimento multiplataforma, como descrito na Seção 2.3, onde o uso do sistema operacional *Mac OS X* mais atualizado é requisito para o desenvolvimento de aplicações para os sistemas operacionais *iOS* e *Mac OS*, presentes nos ambientes de dispositivos móveis e *Desktop*.

Para a avaliação das aplicações desenvolvidas foi utilizado o notebook descrito anteriormente, com os respectivos sistemas operacionais presentes nos ambientes *Desktop*: *Mac OS X* versão *Big Sur 11.6* e *Windows 10*. Este notebook também foi utilizado para testes nos ambientes *Web* com os navegadores *Google Chrome* e *Safari*. Para a avaliação no ambiente *Desktop Linux* (*Elementary OS 5.1*) foi utilizado o computador *Dell Inspiron 15 Série 3000*, equipado com um processador *Intel Core I3-4005U* com dois núcleos de processamento e *clock* de 1.7GHz, possuindo memória RAM de 4GB. Testes para o ambiente de dispositivos móveis foram utilizados os aparelhos: *iPhone 11 Pro* e *TCL 5152D* para os sistemas

iOS e *Android* respectivamente. Estes dispositivos também foram utilizados para avaliações nos ambientes PWA.

1.9.2 Ferramentas para desenvolvimento

Com o intuito de extrair as capacidades do hardware utilizado em conjunto com os *frameworks* multiplataforma, foram utilizadas ferramentas de Ambiente de Desenvolvimento Integrado (IDE). Os IDEs possibilitam otimizações para o processo de desenvolvimento por meio de comentários instantâneos, mensagens e avisos de erros, ferramentas de auto completar, recursos que indicam possíveis correções e assistentes de padronização de endentação (HUY, 2012). No desenvolvimento deste trabalho foram utilizados IDEs como: *Android Studio*[31], *Visual Studio*[32] e *Visual Studio Code*[33].

O *Android Studio* é o IDE oficial do desenvolvimento para o sistema operacional *Android*, foi desenvolvido pelo *Google* e é atualmente baseado no software *IntelliJ IDEA* da *JetBrains* (GOOGLE, 2021a). O *Android Studio* conta com diversos recursos que aumentam a produtividade como: funções de inspeção de artefatos e componentes, coleta de dados e ferramentas de *benchmark*, além de uma ferramenta que gerencia emuladores *Android* (BIøRN-HANSEN *et al*., 2020).

Ambiente de desenvolvimento integrado da *Microsoft*, o *Visual Studio*, permite a edição, depuração, compilação e publicação de aplicativos (MICROSOFT, 2021a). O *Visual Studio* conta também com uma grande variedade de ferramentas e *plugins* que auxiliam os desenvolvedores nas atividades diárias (BELLMAN *et al*., 2018).

O *Visual Studio Code* é um IDE criado pela *Microsoft* com o intuito de possibilitar uma experiência simplificada mantendo os aspectos de otimização de desenvolvimento (MICROSOFT, 2021b). Este IDE conta com ferramentas de customização e de controle de versionamento integrado, é capaz de extrair as capacidades dos *frameworks* multiplataforma com o auxílio de pacotes tecnológicos agregadores (VIEIRA; FARIAS, 2020).

[31] https://developer.android.com/studio
[32] https://visualstudio.microsoft.com/pt-br/
[33] https://code.visualstudio.com

1.9.3 Emuladores

Para otimização do processo de desenvolvimento e com suporte nativo aos IDEs, foi utilizado um conjunto de emuladores com os sistemas *Android* e *iOS*. Este conjunto de emuladores permite uma maior rapidez no processo de depuração em tempo de projeto e maior facilidade no ciclo de desenvolvimento (RIEGER; MAJCHRZAK, 2019).

Para a emulação do sistema *Android* o *Android Emulator* foi utilizado, possibilitando a simulação de recursos de dispositivos reais, como é o caso de recebimento de chamadas telefônicas, mensagens de texto, sensores de reconhecimento digital e acelerômetro, diferentes velocidades de conexão e permite acesso à loja de aplicações *Play Store* (GOOGLE, 2021b). Além dos itens apresentados, o emulador ainda aumenta a compatibilidade com uma maior amplitude de dispositivos, possibilitando a escolha de diferentes modelos com distintas resoluções e versões do sistema operacional *Android*.

Utilizado para o sistema operacional *iOS* o *Simulator* presente no IDE do *Xcode* permite a simulação de dispositivos *Apple*. Este simulador proporciona o uso de todos os *iPhones* que possuem a última versão do *iOS*, além também de possibilitar a integração com outros dispositivos do ambiente *Apple* como o *iPad*, *Apple Watch* e *Apple TV* (APPLE, 2021). Esta compatibilidade proporciona a simulação de todos os dispositivos *Apple* com as versões mais recentes dos sistemas operacionais de cada um dos ambientes.

1.9.4 Ferramentas de prototipação

Com base nos aspectos essenciais das aplicações, a prototipação do design de interfaces possibilita a otimização na etapa de idealização dos aspectos visuais dos projetos (GAJJAR *et al*., 2021). No âmbito da prototipação de interfaces está presente a construção de *Wireframes* que são ilustrações semelhantes ao layout dos elementos fundamentais das interfaces. Os *Wireframes* sugerem a estrutura das interfaces e os relacionamentos entre elas. Para a prototipação de interfaces foram utilizadas as ferramentas *Adobe XD*[34] e *Figma*[35].

[34] https://www.adobe.com/br/products/xd.html
[35] https://www.figma.com

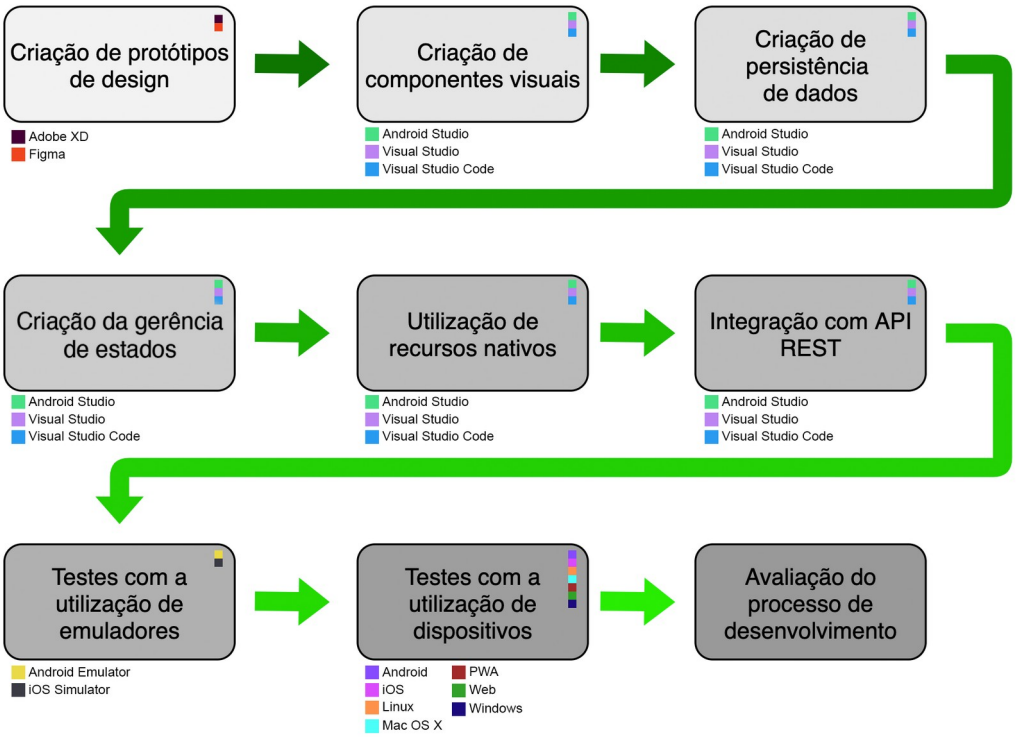
O software de prototipação *Adobe XD* possibilita a criação de designs de interfaces de alta fidelidade com ferramentas de componentização, possibilitando a demonstração de interações mínimas e conversão de artefatos para arquivos legíveis às aplicações criadas pelos *frameworks* multiplataforma (LI; ZHANG, 2021).

Figma é um editor gráfico de vetor e prototipagem de projetos de design de interface amplamente utilizado pelos designers, possibilitando grande flexibilidade e integração com ferramentas colaborativas (KIM; KIM, 2021). O *Figma* pode ser utilizado em ambientes *Desktop* e principalmente em navegadores por meio de ambientes *Web*.

1.10 Métodos

Nesta seção são apresentados os métodos utilizados nos procedimentos efetuados. A sequência aplicada para a realização do trabalho está descrita no diagrama contido na Figura 13, onde cada passo será detalhado na sequência assim como a maneira que serão executados.

Figura 13 - Diagrama com a sequência do trabalho



Fonte: Autoria própria

Primeiramente foi necessário criar os protótipos do design das interfaces desenvolvidas. Para a criação destes protótipos de interfaces foram utilizadas as ferramentas *Adobe XD* e *Figma*. Os protótipos foram elaborados em conjunto com experiências personalizadas para cada um dos ambientes, onde foram utilizadas técnicas de responsividade adequando as interfaces para os ambientes *Desktop*, *Web*, PWA e de dispositivos móveis. Estas interfaces foram aplicadas no desenvolvimento para cada um dos *frameworks* multiplataforma (*Flutter*, *React Native*, *Ionic*, *Vue Native* e *Xamarin*).

Com os protótipos de interfaces propostos foi iniciado o processo de desenvolvimento composto pela criação de componentes visuais, criação de persistência de dados, criação da gerência de estados, utilização dos recursos nativos e integração com API *Representational State Transfer* (REST). Estes processos utilizaram dos seguintes IDEs em conjunto com os respectivos *frameworks* multiplataforma:

- *Android Studio*: *Flutter* e *Ionic*;
- *Visual Studio*: *Xamarin*;
- *Visual Studio Code*: *Flutter*, *Ionic*, *React Native* e *Vue Native*.

A etapa de desenvolvimento foi iniciada pela criação de componentes visuais da aplicação. Esta etapa utilizou da metodologia de componentização dos artefatos, foram criadas as páginas e a navegação entre as rotas dos fluxos descritos no processo de prototipação. Após a criação dos componentes visuais foram criadas as metodologias de persistência de dados internos da aplicação, característica essencial para demonstrar as possibilidades criadas pelos *frameworks* multiplataforma. Em seguida foi iniciado o processo de criação da gerência de estados, que demonstra a escalabilidade e complexibilidade da aplicação devido aos múltiplos dados presentes nos componentes distribuídos pelas páginas.

Após a etapa de criação de gerência de estados foram exploradas as viabilidades de utilização dos recursos nativos dos dispositivos, aspecto necessário para a interação da aplicação junto as possibilidades beneficiadas pelos diferentes componentes de hardware, como sensores e acelerômetros. Em seguida foram aplicadas as metodologias de integração com serviços web (API REST), esta integração explora a interoperabilidade da aplicação com os demais sistemas conectados a Internet.

Para proporcionar testes das etapas de desenvolvimento, foram utilizados inicialmente os emuladores *Android Emulator* e *iOS Simulator*. Estes emuladores possibilitaram o tratamento de problemas em tempo de projeto, além da ampliação das variedades de configurações como resoluções e versões de sistemas operacionais. Após a finalização das etapas de testes iniciais com a utilização dos emuladores, as aplicações foram transferidas para os dispositivos com os sistemas operacionais *Android*, *iOS*, *Linux*, *Mac Os X*, *Web* e *Windows*.

A última etapa consistiu em avaliar os processos de desenvolvimento, bem como também as capacidades apresentadas pelos *frameworks* multiplataforma, possibilitando uma análise comparativa entre as ferramentas. Dentro deste aspecto, métricas foram definidas buscando aplicar artefatos relatados em trabalhos correlatos junto a experiências absorvidas nas etapas anteriores, são elas:

- **Suporte ao Desenvolvimento:** Possuir critérios sociais como ampla comunidade, utilização no mercado, documentação objetiva, constância de atualizações e disponibilidade de componentes oferecidos pela plataforma e por terceiros que possibilitem uma otimização do desenvolvimento (PALOMBA *et al*., 2018);
- **Ferramentas de Componentes Visuais:** Possuir ferramentas de componentes visuais que suportam a produtividade e qualidade do desenvolvimento, subsidiando a construção de interfaces fluídas e otimizadas (HUDLI *et al*., 2015);
- **Ferramentas para a Gerência de Estados:** Permitir a gerência de estados dos componentes presentes na interface do usuário da aplicação (EL-KASSAS *et al*., 2017);
- **Ferramentas para Persistência de Dados:** Dispor de ferramentas que possibilitam o desenvolvimento de aplicações com persistência e gerenciamento de dados alocados localmente (WILLOCX *et al*., 2016);
- **Consumo de Serviços Conectados:** Suportar o consumo de API providas da Internet por meio de ferramentas nativas ou de terceiros (BIøRN-HANSEN *et al*., 2019);
- **Acesso a Recursos Nativos:** Possibilitar a utilização de recursos nativos como acelerômetro e demais sensores, sendo por meio de bibliotecas nativas ou desenvolvidas pela comunidade (FERREIRA *et al*., 2018);
- **Responsividade:** Verificar recursos (nativos e/ou de terceiros) que subsidiem a flexibilização de interfaces, possibilitando o uso de pacotes que auxiliam a responsividade de diversos elementos em relação ao tamanho do dispositivo, resoluções de telas e diferentes métodos de entrada de dados, seja por meio de periféricos ou digitalizadores de toque (RIEGER; MAJCHRZAK, 2019);
- **Reuso de Habilidades:** Permitir o reuso de habilidades e conceitos de outras tecnologias, mantendo a constância de padrões aceitos pela comunidade de desenvolvedores buscando melhorar a curva de aprendizado (HUDLI *et al*., 2015);

- **Plataformas de Desenvolvimento para a Execução:** Quantificar as plataformas de desenvolvimento suportadas nativamente para a execução da aplicação (BIøRN-HANSEN *et al*., 2020);
- **Testes em Tempo de Projeto:** Possibilitar testes no momento do desenvolvimento por meio de *softwares* de emulação e programas nos respectivos ambientes de execução (RIEGER; MAJCHRZAK, 2019);
- **Processo para Geração de Aplicações:** Possuir um processo otimizado para a geração de *builds* para as respectivas plataformas de execução (BIøRN-HANSEN *et al*., 2019);
- **Tamanho do Arquivo da Aplicação:** Mensurar artefatos finais para instalação nas plataformas (AHTI *et al*., 2016).

Com os dados coletados e aplicados nas métricas descritas pelo processo de desenvolvimento, foram feitas as tabulações e os mesmos foram analisados. A análise dos resultados e discussões se baseou na escala de Likert, com as escalas: Excelente (5), Muito bom (4), Bom (3), Razoável (2) e Ruim (1).

Para o suporte ao desenvolvimento, os critérios de aplicação da escala de Likert descrevem: a quantidade de amostragem para documentação, quantidade de conteúdo compartilhado pelos desenvolvedores, o nível de atualização do conteúdo encontrado e a variedade de técnicas demonstradas para a resolução dos problemas.

Os critérios de avaliação para as ferramentas de componentes visuais, ferramentas para gerência de estados e ferramentas para persistência de dados descrevem: a variedade de recursos nativos ou de terceiros, a maturidade das ferramentas, a volatilidade das ferramentas e a aplicabilidade no desenvolvimento.

A aplicação da escala de Likert para as métricas de consumo de serviços conectados caracterizam: a versatilidade no consumo dos dados, a documentação da técnica consumidora e a flexibilização de cabeçalhos e parâmetros. O reuso de habilidades aplica a escala perante a características relacionadas ao uso de técnicas difundidas na comunidade de desenvolvimento, a linguagem utilizada pelo *framework* e a configuração do ambiente de desenvolvimento.

No âmbito de plataformas de execução, a avaliação é disposta por meio da variedade de ambientes de execução suportados nativamente, considerando a excelência com o suporte para todos os ambientes dispostos na Seção 2.3. Para a

responsividade, técnicas descritas na Seção 2.2 foram aplicadas, considerando a experiência do usuário na utilização da aplicação modelo.

As técnicas necessárias para a utilização e acesso a recursos nativos aos dispositivos foram avaliadas por meio da configuração necessária, parametrizando com experiências existentes no retrospecto tecnológico (2.1), de maneira a demonstrar a maturidade dos *frameworks*.

Para avaliar os testes em tempo de projeto, a escala de Likert foi disposta com a utilização de ferramentas que possibilitam a escalabilidade do desenvolvimento, viabilizando os testes com emuladores dos ambientes de execução suportados, diminuindo a disparidade do desenvolvimento com a experiência final.

No âmbito do processo de geração das aplicações, a escala foi disposta por meio da comodidade construída pelos *frameworks*, com ferramentas que facilitam o processo e preparam as aplicações para publicações posteriores. Por fim, a escala de Likert foi avaliada para o tamanho das aplicações geradas, com a métrica adotada por meio da média dos valores obtidos e a distância proporcional entre eles, sendo o menor tamanho obtido Excelente.

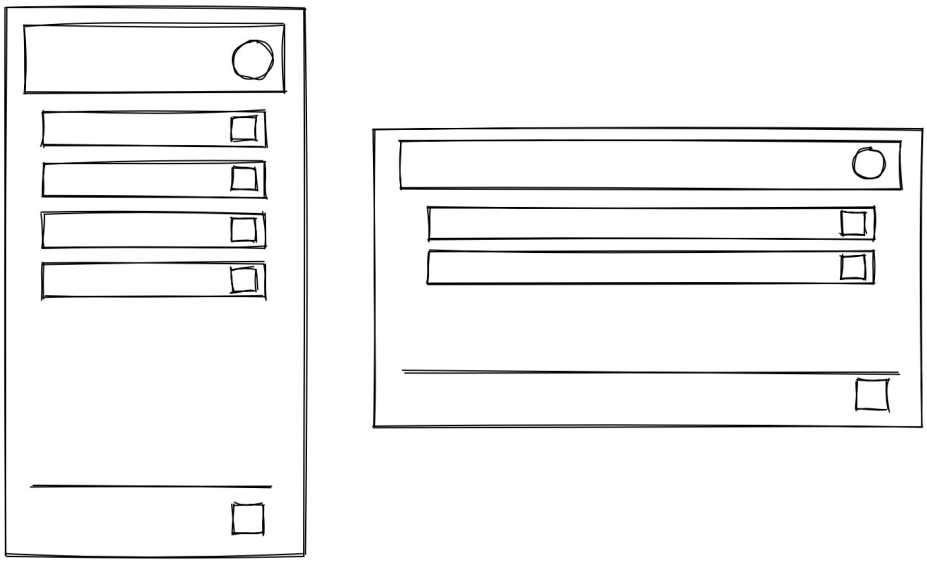
4 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados todos os resultados dos experimentos executados a partir do desenvolvimento com a utilização dos *frameworks* multiplataforma. Para cada um dos *frameworks*, foram feitos apontamentos a respeito das métricas e aspectos relevantes para auxiliar o entendimento das abordagens utilizadas.

1.11 Protótipo e design

Com o intuito de analisar a utilização dos *frameworks* multiplataforma, foi desenvolvido um protótipo modelo para a aplicação. A prototipação foi iniciada por meio de técnicas de design (Seção 2.2), com a criação de *wireframes* que facilitaram na transposição do protótipo de alta fidelidade conforme descrito na Figura 14.

Figura 14 - *Wireframes* da aplicação modelo

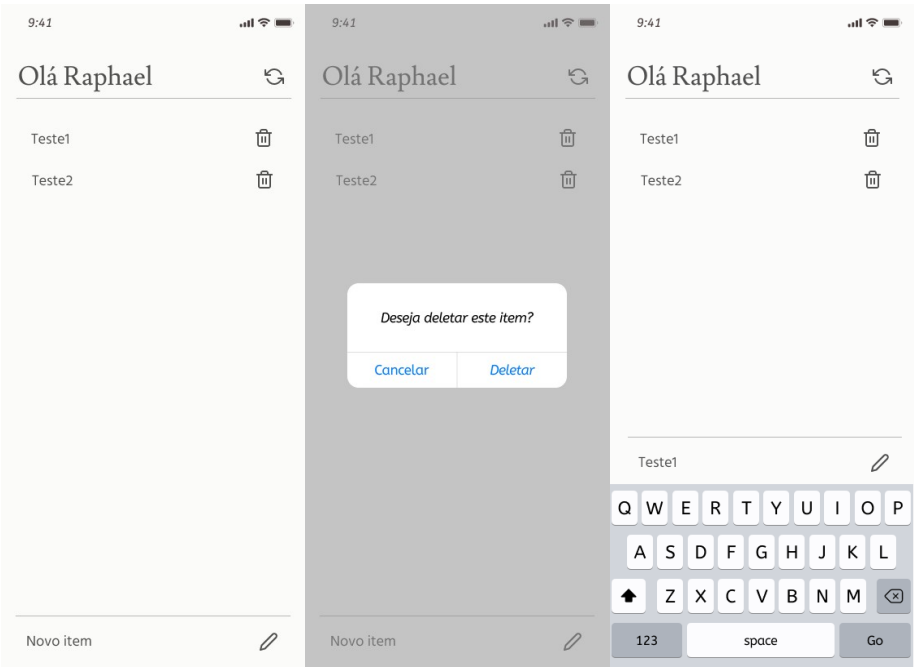


Fonte: Autoria própria

A aplicação foi escolhida unicamente para este trabalho como uma demonstração didática, dispondo de uma lista de afazeres (*TODO List*). Ela foi desenvolvida considerando tratamentos de possíveis erros e exceções dispostos por meio de caixas de alerta portáveis, de maneira a manter a identidade visual de cada ambiente de execução.

Na Figura 15 é demonstrada a tela principal da aplicação, no topo da interface é descrito um nome consumido por uma API[36] geradora de dados aleatórios. Na Figura 16 é descrito o modo de imagem para visão panorâmica, otimizando a utilização do espaço criado pela visualização da interface.

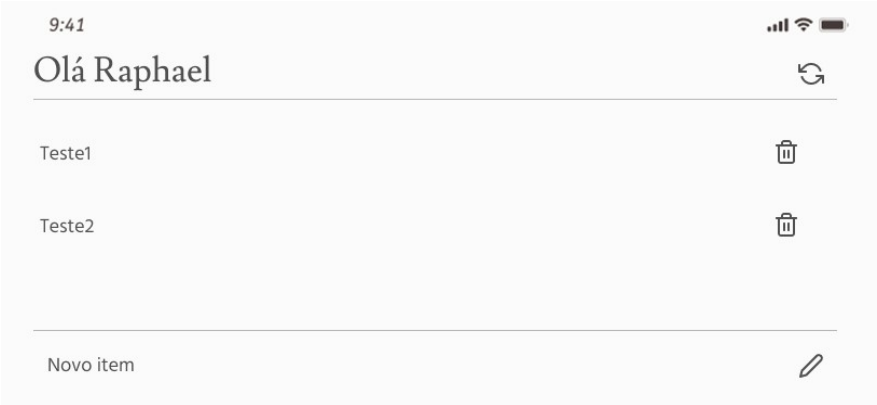
Figura 15 - Protótipos de alta fidelidade



(a) (b) (c)

Fonte: Autoria própria

Figura 16 - *Protótipo de alta fidelidade em visão panorâmica*



Fonte: Autoria própria

[36] https://randomuser.me

1.12 Processo de desenvolvimento

Com o protótipo construído, iniciou-se o processo de desenvolvimento do modelo da aplicação para cada um dos respectivos *frameworks* multiplataforma: *React Native*, *Ionic*, *Vue Native*, *Xamarin* e *Flutter*. Todos os *frameworks* contaram com a instalação inicial seguido pela preparação do ambiente de depuração e homologação. Para análise dos dados coletados foi aplicada a escala de Likert, conforme descrito na Seção 3.2.

1.12.1 React Native

Antes da instalação e configuração do ambiente de desenvolvimento com o uso do *React Native*, foi necessária a instalação do ambiente de execução *Node.js*[37]. O *Node.js* possibilita a execução *server-side* de aplicações *JavaScript*, possibilitando a depuração na máquina localmente (NODE.JS, 2021).

A instalação do *React Native* ocorre por meio do *Node.js*, seguido por duas possíveis abordagens: instalação nativa exclusiva para cada um dos sistemas operacionais *Android* e *iOS*, ou por meio da ferramenta *Expo*[38] que otimiza comandos específicos para o processo de desenvolvimento. A instalação é bem documentada pelo domínio oficial do *React Native*[39].

Iniciando o desenvolvimento, a depuração em tempo de projeto é simplificada com a utilização do conjunto de emuladores (*Android* e iOS). Como foi optado pela instalação com a ferramenta *Expo*, sempre que iniciada a aplicação é disposto um painel visível no navegador padrão do computador, também sendo instalada a aplicação no centralizador *Expo*.

As ferramentas para construção de componentes visuais trazem uma alta gama de possibilidades para a criação do design proposto, estas ferramentas são de fácil acesso pela comunidade de desenvolvedores do *React Native*. Possui também grande variedade de ferramentas para gerência de estados, tanto nativamente com o *useState* como com recursos de terceiros.

---

[37] https://nodejs.org/
[38] https://expo.dev
[39] https://reactnative.dev/docs/environment-setup

Para a persistência de dados, ferramentas como *AsyncStorage*[40], *SQLite 2*[41] e *Realm*[42] podem ser utilizadas. Sendo possível um grande reuso de habilidade dada a grande aderência da linguagem *JavaScript*, além do *HTML* e *CSS* que apoiam na criação dos componentes visuais. Ferramentas como *axios*[43] podem ser utilizadas para o consumo de APIs REST. A configuração para utilização de recursos nativos mantém o padrão, sendo único para cada um dos respectivos ambientes de execução.

O *React Native* possui um otimizado processo de geração de aplicativos. Com a utilização da ferramenta *Expo* um simples comando gera o instalador, possibilitando a escolha do ambiente de execução de destino, com o anexo das normativas de ambas as lojas de aplicativos (*AppStore* e *PlayStore*). O tamanho do arquivo da aplicação modelo gerada ficou com 50,2 MB para o *Android* e 122 MB para o *iOS*, possuindo responsividade nos dispositivos descritos na Seção 3.1. No Quadro 1 estão descritas as análises das métricas aplicadas ao *framework React Native*[44].

Quadro 1 – Análise das métricas para o *framework React Native*

Métricas	Escala de Likert
Suporte ao Desenvolvimento	Excelente
Ferramentas de Componentes Visuais	Excelente
Ferramentas para a Gerência de Estados	Excelente
Ferramentas para Persistência de Dados	Excelente
Consumo de Serviços Conectados	Excelente
Acesso a Recursos Nativos	Muito bom
Responsividade	Excelente
Reuso de Habilidades	Excelente
Plataformas de Desenvolvimento para Execução	Bom
Testes em Tempo de Projeto	Excelente
Processo para Geração de Aplicações	Excelente
Tamanho do Arquivo da Aplicação	Bom

Fonte: Autoria própria

[40] https://reactnative.dev/docs/asyncstorage
[41] https://github.com/craftzdog/react-native-sqlite-2
[42] https://docs.mongodb.com/realm/sdk/react-native/
[43] https://github.com/axios/axios
[44] https://github.com/raphaelbassani/ReactNative-RaphaelTCC

1.12.2 Ionic

Para instalação do *Ionic*, o *Node.js* mais uma vez se faz necessário. Após a instalação do *Node.js*, foi realizada a criação do projeto com a escolha do *framework* que será responsável pelos artefatos visuais, as opções são: *Angular.js*, *React.js* ou *Vue.js*. Para o desenvolvimento deste trabalho foi escolhido o *Angular.js*. Por fim, são dadas algumas opções de aplicações exemplo para a criação do novo projeto:

- *Tabs*: Um projeto exemplo com visão de interface com uso de guias;
- *Sidemenu*: Projeto inicial com um menu lateral e navegação na área de conteúdo;
- *Blank*: Projeto em branco;
- *My-first-app*: Explora o uso da câmera do dispositivo com integração na galeria de fotos;
- *Conference*: Um projeto exemplo com um menu lateral e navegação no rodapé, explora o uso de troca de temas e geolocalização.

Posterior a escolha do modelo de aplicação no novo projeto, foi iniciado o desenvolvimento da aplicação modelo (*TODO List*). Os testes em tempo de projeto são dispostos por meio do *localhost* via navegador. Para depuração com a utilização dos emuladores foi utilizado o *Android Studio* para o sistema *Android* e do *Xcode* para compilação no sistema *iOS*.

O conjunto de ferramentas para a gerência de estados é amplo, com opções bem documentadas pelo próprio *Ionic*[45]. Também possui opções de ferramentas para a persistência de dados, utilização de bibliotecas como *storage*[46] e *SQLite*[47]. De maneira geral o conteúdo disposto pela comunidade de desenvolvedores do *Ionic* possui materiais legados na sua utilização, com uma carência em conteúdos mais atualizados.

As ferramentas que suportam a construção de interfaces possuem opções dispostas pela documentação do *Ionic*. A reutilização de habilidades no processo de desenvolvimento é ampla, dado a utilização do ecossistema do *JavaScript*. O *Ionic*

[45] https://ionic.io/enterprise-guide/state-management
[46] https://ionicframework.com/docs/vue/storage
[47] https://ionicframework.com/docs/native/sqlite

possibilita também a utilização de pacotes como o *axios* para o consumo de serviços providos da Internet.

Para a utilização de recursos do dispositivo, a configuração se dá por meio do uso de bibliotecas de terceiros ou edição no código nativo do ambiente de execução em questão. O processo de geração das aplicações é dependente dos IDEs oficiais dos sistemas *Android* e *iOS* (*Android Studio* e *Xcode*). Sendo o espaço consumido em disco para os executáveis de cada uma das versões: 13 MB para o *Android* e 36,1 MB para o *iOS* respectivamente.

Nos testes das aplicações construídas com o *Ionic* foram notados alguns pontos de otimização comparados a execução em tempo de projeto, possuindo responsividade nos dispositivos utilizados para testes. Estão descritas as análises das métricas aplicadas ao *framework Ionic*[48] no Quadro 2.

Quadro 2 – Análise das métricas para o *framework Ionic*

Métricas	Escala de Likert
Suporte ao Desenvolvimento	Muito bom
Ferramentas de Componentes Visuais	Bom
Ferramentas para a Gerência de Estados	Excelente
Ferramentas para Persistência de Dados	Excelente
Consumo de Serviços Conectados	Excelente
Acesso a Recursos Nativos	Muito bom
Responsividade	Bom
Reuso de Habilidades	Excelente
Plataformas de Desenvolvimento para Execução	Muito bom
Testes em Tempo de Projeto	Bom
Processo para Geração de Aplicações	Bom
Tamanho do Arquivo da Aplicação	Excelente

Fonte: Autoria própria

1.12.3 Vue Native

Antes de iniciar o processo de desenvolvimento com o uso do *Vue Native*, é necessária a instalação do *Node.js*. No momento da configuração inicial do ambiente do *Vue Native* são dispostas algumas opções a serem seguidas: o uso da *Command*

[48] https://github.com/raphaelbassani/Ionic-RaphaelTCC

Line Interface (CLI) nativa do *framework Vue*[49], por meio do CLI do *framework React Native*[50] ou via *Expo*, para o desenvolvimento deste trabalho foi optado pela opção via *Expo*.

Após a configuração do ambiente, foi necessário a instalação das dependências do projeto com o uso do *React Native*. Estas dependências fazem parte da base do *Vue Native*, que conecta as possibilidades de execução do *React Native* aos artefatos visuais progressivos do *Vue.js*.

Iniciando o processo de desenvolvimento da aplicação didática do trabalho, o *Vue Native* possibilita o uso de diversas ferramentas para a construção de componentes visuais, característica herdada do *framework Vue.js*. O fato de ser um *framework* em fase de desenvolvimento limitou a aplicação da técnica de persistência de dados dada a falta de ferramentas disponíveis, entretanto as ferramentas para a gerência de estado demonstraram a progressividade do *Vue Native*.

O consumo de serviços providos da Internet foi aplicável por meio da ferramenta *axios*, com um alto grau de reuso de habilidades, fato aplicável em *frameworks* que utilizam o ecossistema da linguagem *JavaScript*. Com o processo para acessar os recursos nativos dos dispositivos de maneira configurável nativamente em cada ambiente.

As características relacionadas ao suporte ao desenvolvimento estão em construção, demonstrando uma grande capacidade de aderência da comunidade de desenvolvedores. Com o uso do *Expo*, os aspectos relacionados a geração das aplicações e testes em tempo de projeto são otimizados. O executável do sistema operacional *Android* foi gerado com o tamanho de 49,5 MB e para o *iOS* com 121,8 MB. No Quadro 3 são demonstrados os dados obtidos durante o processo de desenvolvimento[51].

[49] https://github.com/GeekyAnts/vue-native-cli
[50] https://www.npmjs.com/package/react-native-cli
[51] https://github.com/raphaelbassani/VueNative-RaphaelTCC

Quadro 3 – Análise das métricas para o *framework Vue Native*

Métricas	Escala de Likert
Suporte ao Desenvolvimento	Razoável
Ferramentas de Componentes Visuais	Excelente
Ferramentas para a Gerência de Estados	Muito bom
Ferramentas para Persistência de Dados	Razoável
Consumo de Serviços Conectados	Excelente
Acesso a Recursos Nativos	Muito bom
Responsividade	Excelente
Reuso de Habilidades	Excelente
Plataformas de Desenvolvimento para Execução	Bom
Testes em Tempo de Projeto	Excelente
Processo para Geração de Aplicações	Excelente
Tamanho do Arquivo da Aplicação	Bom

Fonte: Autoria própria

1.12.4 Xamarin

O processo de instalação do *Xamarin* é simplificado por meio da utilização do *Visual Studio*, requisito para o desenvolvimento com o uso deste *framework*. Após a instalação do IDE, foi iniciado o desenvolvimento com ferramentas facilitadoras integradas ao *Visual Studio* e abordagem direta para a depuração em tempo de projeto, utilizando os emuladores do *Android* e o simulador do *iOS*.

Existem diversas opções de ferramentas para o uso da metodologia da gerência de estados, possibilitando a aplicação de *Bindings* (ligação de dados) utilizando-se da arquitetura MVVM e uso da ferramenta *INotifyPropertyChanged*[52]. A técnica de persistência de dados foi aplicada com a ferramenta *SQLite*[53], que possui uma instalação simplificada via *Nuget*[54], um gerenciador de pacotes nativo ao *Visual Studio* projetado para otimizar a reutilização e o compartilhamento de códigos. Ferramentas como o *Entity Framework Core* também podem ser utilizadas para integrar o banco de dados *SQLite* no mapeamento relacional de dados, com configurações especificas para cada plataforma respectivamente.

[52]https://docs.microsoft.com/pt-br/dotnet/api/system.componentmodel.inotifypropertychanged?view=net-5.0
[53] https://docs.microsoft.com/pt-br/xamarin/get-started/quickstarts/database?pivots=windows
[54] https://www.nuget.org

Outras ferramentas integradas ao *Visual Studio* que auxiliam no processo de desenvolvimento com o *Xamarin* são as de geração de executáveis, ocorrendo diretamente pelo console do IDE. O *Xamarin* também suporta pacotes como o *Newtonsoft*[55], que desserializa as respostas vindas da API por meio de ferramentas de consumo de serviços da Internet. Para este projeto a ferramenta de consumo utilizada foi a padrão do *Xamarin*[56], permitindo a manipulação de campos e cabeçalhos via *Query String*, técnica que atribui valores específicos para a chamada.

No momento da criação de componentes visuais, o *Xamarin* apresentou bibliotecas que integram a utilização dos arquivos com a linguagem de marcação *XML*, linguagem utilizada para a criação de interfaces de usuário da *Microsoft* (MICROSOFT, 2021e). Comparado aos demais *frameworks* de estudo, o *Xamarin* utiliza-se das linguagens *XML* e *C#,* que diminuem o grau de reuso de habilidades em comparação com os demais *frameworks* multiplataforma, apesar da ótima gama de plataformas suportadas pelo *framework*.

Para acessar os recursos nativos dos dispositivos, é necessária a configuração manual para cada uma das plataformas suportadas, como na utilização de fontes customizadas e imagens que devem ser inseridas em todos os repositórios. Quanto ao suporte ao desenvolvimento, o *Xamarin* demonstrou uma comunidade participativa com conteúdos legados abordando todos os pontos explorados pela aplicação *TODO List*. Com um executável gerado de 50,8 MB para o sistema *iOS* e 23,6 MB para o *Android*. As métricas aplicadas ao *Xamarin*[57] estão descritas no Quadro 4.

[55] https://www.nuget.org/packages/Newtonsoft.Json/
[56] https://docs.microsoft.com/pt-br/xamarin/xamarin-forms/data-cloud/web-services/rest
[57] https://github.com/raphaelbassani/Xamarin-RaphaelTCC

Quadro 4 – Análise das métricas para o *framework Xamarin*

Métricas	Escala de Likert
Suporte ao Desenvolvimento	Muito bom
Ferramentas de Componentes Visuais	Bom
Ferramentas para a Gerência de Estados	Excelente
Ferramentas para Persistência de Dados	Excelente
Consumo de Serviços Conectados	Muito bom
Acesso a Recursos Nativos	Muito bom
Responsividade	Excelente
Reuso de Habilidades	Bom
Plataformas de Desenvolvimento para Execução	Muito bom
Testes em Tempo de Projeto	Muito Bom
Processo para Geração de Aplicações	Excelente
Tamanho do Arquivo da Aplicação	Excelente

Fonte: Autoria própria

1.12.5 Flutter

A instalação do *Flutter* se dá por meio do SDK[58] disposto na documentação oficial da ferramenta, seguido pela configuração de ambiente que permite a utilização dos comandos do *Flutter,* por meio do console de comandos nativo do sistema operacional, *Console* ou *Terminal*. Após a instalação e configuração do *Flutter*, foi iniciado o processo de desenvolvimento da aplicação *TODO List*.

A depuração em tempo de projeto é otimizada com o uso dos emuladores, navegadores e executáveis das plataformas suportadas pelo *Flutter*. O reuso de habilidades foi limitado em comparação com demais *frameworks*, dado que o *Flutter* utiliza-se da linguagem *Dart*. Para o acesso a recursos nativos, é necessária a configuração manual dentro de cada uma das respectivas plataformas de desenvolvimento.

Com a utilização de bibliotecas dispostas pela comunidade no gerenciador de pacotes *pub.dev*[59], foi possível a aplicação das técnicas de persistência de dados, a ferramenta utilizada no projeto foi o *sqflite*[60], que aplica um mecanismo de banco de dados nativamente a aplicação. Para a gerência de estados, são dispostas

[58] https://flutter.dev/docs/get-started/install/macos
[59] https://pub.dev
[60] https://pub.dev/packages/sqflite

ferramentas nativas como o *setState*, entretanto para o desenvolvimento da aplicação modelo foi utilizado o pacote *provider*[61] com auxílio do *ChangeNotifier*[62], que em conjunto monitoram os estados dos componentes reconstruindo-os a cada atualização.

Para o consumo de serviços conectados foi utilizado o cliente HTTP *dio*[63], com a tradução das respostas dos serviços feita com ferramentas nativas da linguagem *Dart*, aplicando a arquitetura MVC na organização do projeto. Para a criação de componentes visuais, o *Flutter* dispõe de ferramentas integradas com os pacotes *material*[64] e *cupertino*[65], que se utilizam dos componentes nativos dos sistemas *Android* e *iOS* respectivamente.

O *Flutter* também possui uma ampla comunidade ativa, suportando todos os pontos explorados no desenvolvimento do projeto, além de um processo de geração de executáveis otimizado via comandos, implementadas no SDK, com 32,2 MB para o executável do sistema *Android* e 113,4 MB para o *iOS*. No Quadro 5 são dispostas as métricas aplicadas ao *framework Flutter*[66].

Quadro 5 – Análise das métricas para o *framework Flutter*

Métricas	Escala de Likert
Suporte ao Desenvolvimento	Excelente
Ferramentas de Componentes Visuais	Excelente
Ferramentas para a Gerência de Estados	Excelente
Ferramentas para Persistência de Dados	Excelente
Consumo de Serviços Conectados	Muito bom
Acesso a Recursos Nativos	Muito bom
Responsividade	Excelente
Reuso de Habilidades	Bom
Plataformas de Desenvolvimento para Execução	Excelente
Testes em Tempo de Projeto	Excelente
Processo para Geração de Aplicações	Excelente
Tamanho do Arquivo da Aplicação	Muito bom

Fonte: Autoria própria

[61] https://pub.dev/packages/provider
[62] https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple
[63] https://pub.dev/packages/dio
[64] https://flutter.dev/docs/development/ui/widgets/material
[65] https://flutter.dev/docs/development/ui/widgets/cupertino
[66] https://github.com/raphaelbassani/Flutter-RaphaelTCC

1.13 Discussão dos resultados

O presente trabalho visou analisar a experiência de utilização de *frameworks* multiplataforma para o desenvolvimento de uma aplicação de código único portável, de maneira a otimizar os processos relacionados e melhorar as experiências dos usuários e principalmente dos desenvolvedores. Dada a crescente busca do mercado por ferramentas que possibilitem maior extração de valores agregadores ao produto, a aderência dos *frameworks* multiplataforma tem aumentado de maneira vertiginosa, o que comprova a maturidade adquirida por estas ferramentas.

Com um impacto cultural dentre todas as camadas da sociedade, se faz necessário que as aplicações possuam um alto grau de otimizações relacionadas a compatibilidade e suporte ao desenvolvimento. Dentre os principais impactos econômicos, os *frameworks* multiplataforma se destacam pela abstração técnica, sendo necessário a criação de um único código para as plataformas com a padronização classificada, extraindo componentes visuais próprios de cada ambiente de execução com o mantimento de recursos.

Após o desenvolvimento da aplicação modelo com a utilização dos *frameworks* multiplataforma, com as métricas dispostas extraídas, foi possível observar a viabilidade técnica e o grau de maturidade presente em cada uma das ferramentas. Isto possibilitou a comparação técnica para cada um dos aspectos presentes nos *frameworks*, com as respectivas singularidades das metodologias adotadas.

O *Vue Native* demonstrou boas perspectivas para evoluções, possibilitando a extração de progressividade do *Vue.js* em adição com a maturidade do *React Native*. Estes fatos corroboram para um futuro promissor desta ferramenta, que no momento da execução deste trabalho está no processo inicial de desenvolvimento.

Framework com um alto grau de compatibilidade com plataformas de execução, o *Ionic* demonstrou variedade nas ferramentas e bibliotecas de terceiros. Entretanto, foi notável a diferença para os demais *frameworks* em otimizações para processos relacionados ao desenvolvimento. O processo moroso para testes em tempo de projeto e principalmente para a geração de executáveis, pesaram na aplicação das métricas para o uso do *Ionic*.

Possuindo um ambiente de desenvolvimento otimizado, o *Xamarin* representou de maneira sólida as métricas aplicadas no trabalho. Fato este que

surpreende, dado o alto grau de acoplamento com o ambiente da *Microsoft*. Demonstrou também a capacidade de construção com o baixo acoplamento de lógicas, além da grande capacidade de orientação a objetos presente no *C#* em conjunto com o *XML*.

Com destaques para *React Native* e *Flutter*, que demonstraram o potencial no processo de desenvolvimento, principalmente pela presença ativa da comunidade no suporte ao desenvolvimento, o que evidencia a aderência de utilização no mercado. O *React Native* desponta dos demais *frameworks* que utilizam do ambiente de desenvolvimento *JavaScript*, possuindo otimizações em elementos relacionados a geração de aplicações e suporte ao desenvolvimento. Já o *Flutter,* com investimentos por parte do *Google* em uma ferramenta sólida para desenvolvimento multiplataforma. Estes fatos demonstram o alto grau de maturidade das empresas responsáveis pela manutenção destes *frameworks*. Disposto no Quadro 6 estão as métricas aplicadas a todos os *frameworks* e no Quadro 7 são descritas as médias obtidas por cada um dos *frameworks*, adotando cinco (5) para Excelente e um (1) para Ruim.

Quadro 6 – Análise das métricas para todos os *framework*

Métricas	***Flutter***	***Ionic***	***React Native***	***Vue Native***	***Xamarin***
Suporte ao Desenvolvimento	Excelente	Muito bom	Excelente	Razoável	Muito bom
Ferramentas de Componentes Visuais	Excelente	Bom	Excelente	Excelente	Bom
Ferramentas para a Gerência de Estados	Excelente	Excelente	Excelente	Muito bom	Excelente
Ferramentas para Persistência de Dados	Excelente	Excelente	Excelente	Razoável	Excelente
Consumo de Serviços Conectados	Muito bom	Excelente	Excelente	Excelente	Muito bom
Acesso a Recursos Nativos	Muito bom	Muito bom	Muito bom	Muito bom	Muito bom
Responsividade	Excelente	Bom	Excelente	Excelente	Excelente
Reuso de Habilidades	Bom	Excelente	Excelente	Excelente	Bom
Plataformas de Desenvolvimento para Execução	Excelente	Muito bom	Bom	Bom	Muito bom
Testes em Tempo de Projeto	Excelente	Bom	Excelente	Excelente	Muito Bom
Processo para Geração de Aplicações	Excelente	Bom	Excelente	Excelente	Excelente
Tamanho do Arquivo da Aplicação	Muito bom	Excelente	Bom	Bom	Excelente

Fonte: Autoria própria

Quadro 7 – Média das métricas para os *frameworks* multiplataforma

Frameworks	**Média**
Flutter	4,58
Ionic	4,08
React Native	4,58
Vue Native	4,00
Xamarin	4,25

Fonte: Autoria própria

5 CONCLUSÕES

Para a validação dos aspectos potencializados pelos *frameworks* multiplataforma, uma aplicação modelo foi desenvolvida com cada um dos *frameworks*: *Flutter*, *Ionic*, *React Native*, *Vue Native* e *Xamarin*. Com métricas extraídas a partir de uma variedade de referências, foi possível analisar pontos relacionados ao desenvolvimento como: criação de elementos visuais, gerência de estados, persistência de dados, suporte da comunidade de desenvolvedores, consumo de serviços conectados, reuso de habilidades, além também de aspectos ligados diretamente a geração de aplicações, tamanho e quantidade de plataformas suportadas.

A partir da extração das métricas no processo de desenvolvimento, realizou-se a análise dos dados, onde foram observadas as capacidades que os *frameworks* multiplataforma possuem. Cada um deles apresenta suas peculiaridades, sejam elas relacionadas a linguagem de programação utilizada, ambiente de desenvolvimento e/ou compatibilidade. Com o *Vue Native* dispondo de boas ideias a serem consolidadas, conforme o desenvolvimento da ferramenta de maneira mais sólida. O *Ionic* com possíveis pontos de otimização no processo de geração de aplicações e melhorias nos testes em tempo de projeto, dada a falta de compatibilidade na utilização de emuladores de maneira nativa ao *framework*.

Com o *Xamarin* demonstrando débitos relacionados ao suporte de desenvolvimento e ferramentas de componentes visuais, que calharam por diminuir a escala obtida. Sendo assim, percebe-se que o *Flutter* e o *React Native* dispõem de uma maior maturidade nos elementos explorados, com pontos positivos relacionados às ferramentas disponibilizadas e principalmente pelo suporte ao desenvolvimento da comunidade. Dentro deste contexto a utilização do *Flutter* e do *React Native* no desenvolvimento multiplataforma traz diversas otimizações para aplicações de maneira escalar, maior flexibilidade e menor curva de aprendizado com o *React Native* utilizando de tecnologias já consolidadas no mercado e com o *Flutter* possibilitando a evolução de aplicações continuas dentre todas as plataformas suportadas.

Os *frameworks* multiplataforma demonstram grande potencial, seja no ambiente acadêmico com a extração e evolução de habilidades aplicáveis a tarefas

cotidianas, ou no âmbito do mercado com o aumento no grau técnico dos desenvolvedores. Possibilitando a otimização nos processos relacionados a criação e manutenção de aplicativos, além de potencializar a compatibilidade dentre os ambientes de execução.

1.14 Trabalhos Futuros

Após o desenvolvimento deste trabalho, foram dispostos elementos relacionados a questões de interesse, que poderiam dar continuidade ao estudo da utilização de *frameworks* multiplataforma no desenvolvimento de aplicações de código único. Estes elementos são:

- A aplicação de questionários na comunidade de desenvolvedores, de modo a extrair os aspectos relacionados ao uso cotidiano de *frameworks* multiplataforma;
- O estudo no desenvolvimento aprofundado com utilização de um *framework* multiplataforma, aumentando o nível de aplicabilidade de recursos;
- Acrescentar atributos relacionados a componentes visuais e seus respectivos padrões de uso, de maneira a demonstrar as melhorias na experiência dos usuários;
- Desenvolver uma aplicação com técnicas de continuidade.

REFERÊNCIAS

AGHAJANI, E.; NAGY, C.; VEGA-MARQUEZ, O. L.; LINARES-VÁSQUEZ, M.; MORENO, L.; BAVOTA, G.; LANZA, M. Software documentation issues unveiled. In: **Proceedings of the 41st International Conference on Software Engineering**. IEEE Press, 2019. (ICSE '19), p. 1199–1210. Disponível em: https://doi.org/10.1109/ICSE.2019.00122.

AHTI, V.; HYRYNSALMI, S.; NEVALAINEN, O. An evaluation framework for cross-platform mobile app development tools: A case analysis of adobe phonegap framework. In: **Proceedings of the 17th International Conference on Computer Systems and Technologies 2016**. New York, NY, USA: Association for Computing Machinery, 2016. (CompSysTech '16), p. 41–48. ISBN 9781450341820. Disponível em: https://doi.org/10.1145/2983468.2983484.

ALI, M.; JOORABCHI, M. E.; MESBAH, A. Same app, different app stores: A comparative study. In: **Proceedings of the 4th International Conference on Mobile Software Engineering and Systems**. IEEE Press, 2017. (MOBILESoft '17), p. 79–90. ISBN 9781538626696. Disponível em: https://doi.org/10.1109/MOBILESoft.2017.3.

ALTAMIMI, F.; ASIF, W.; RAJARAJAN, M. Dads: Decentralized (mobile) applications deployment system using blockchain : Secured decentralized applications store. In: **2020 International Conference on Computer, Information and Telecommunication Systems (CITS)**. IEEE, 2020. p. 1–8. Disponível em: https://www.doi.org/10.1109/CITS49457.2020.9232506.

ANDROID. **Platform Architecture**. 2021. Disponível em: https://developer.android.com/guide/platform/?hl=pt-br. Acesso em: 28 de junho de 2021.

ANGULO, E.; FERRE, X. A case study on cross-platform development frameworks for mobile applications and ux. In: **Proceedings of the XV International Conference on Human Computer Interaction**. New York, NY, USA: Association for Computing Machinery, 2014. (Interaccio´n '14). ISBN 9781450328807. Disponível em: https://doi.org/10.1145/2662253.2662280.

ANVAARI, M.; JANSEN, S. Evaluating architectural openness in mobile software platforms. In: **Proceedings of the Fourth European Conference on Software Architecture: Companion Volume**. New York, NY, USA: Association for Computing Machinery, 2010. (ECSA '10), p. 85–92. ISBN 9781450301794. Disponível em: https://doi.org/10.1145/1842752.1842775.

APPLE. **Running Your App in the Simulator or on a Device**. 2021. Disponível em: https://developer.android.com/studio/run/emulator/. Acesso em: 22 de julho de 2021.

BAREA, A.; FERRE, X.; VILLARROEL, L. Android vs. ios interaction design study for a student multiplatform app. In: STEPHANIDIS, C. (Ed.). **HCI International 2013 - Posters' Extended Abstracts**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 8–12. ISBN 978-3-642-39476-8.

BELLMAN, C.; SEET, A.; BAYSAL, O. Studying developer build issues and debugger usage via timeline analysis in visual studio ide. In: **Proceedings of the 15th International Conference on Mining Software Repositories**. New York, NY, USA: Association for Computing Machinery, 2018. (MSR '18), p. 106–109. ISBN 9781450357166. Disponível em: https://doi.org/10.1145/3196398.3196463.

BIøRN-HANSEN, A.; GRøNLI, T. M.; GHINEA, G. A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. **ACM Computing Surveys**, Association for Computing Machinery, v. 51, 1 2019. ISSN 15577341.

BIøRN-HANSEN., A.; MAJCHRZAK., T. A.; GRøNLI., T. Progressive web apps: The possible web-native unifier for mobile development. In: INSTICC. **Proceedings of the 13th International Conference on Web Information Systems and Technologies - WEBIST**. SciTePress, 2017. p. 344–351. ISBN 978-989-758-246-2. ISSN 2184-3252. Disponível em: https://www.doi.org/10.5220/0006353703440351.

BIøRN-HANSEN, A.; RIEGER, C.; GRøNLI, T. M.; MAJCHRZAK, T. A.; GHINEA, G. An empirical investigation of performance overhead in cross-platform mobile development frameworks. **Empirical Software Engineering**, Springer, v. 25, p. 2997–3040, 7 2020. ISSN 15737616.

BLOM, S.; BOOK, M.; GRUHN, V.; HRUSHCHAK, R.; Ko ̈HLER, A. Write once, run anywhere a survey of mobile runtime environments. In: **2008 The 3rd International Conference on Grid and Pervasive Computing - Workshops**. IEEE Press, 2008. p. 132–137. Disponível em: https://www.doi.org/10.1109/GPC.WORKSHOPS.2008.19.

BOSNIC, S.; PAPP, I.; NOVAK, S. The development of hybrid mobile applications with apache cordova. In: **2016 24th Telecommunications Forum (TELFOR)**. IEEE, 2016. p. 1–4. Disponível em: https://www.doi.org/10.1109/TELFOR.2016.7818919.

BOUKHARY, S.; COLMENARES, E. A clean approach to flutter development through the flutter clean architecture package. In: **2019 International Conference on Computational Science and Computational Intelligence (CSCI)**. IEEE, 2019. p. 1115–1120. Disponível em: https://www.doi.org/10.1109/CSCI49370.2019.00211.

BOUSHEHRINEJADMORADI, N.; GANAPATHY, V.; NAGARAKATTE, S.; IFTODE, L. Testing cross-platform mobile app development frameworks. In: **Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering.** IEEE Press, 2015. (ASE '15), p. 441–451. ISBN 9781509000241. Disponível em: https://doi.org/10.1109/ASE.2015.21.

BRITO, H.; GOMES, A.; SANTOS, A.; BERNARDINO, J. Javascript in mobile applications: React native vs ionic vs nativescript vs native development. In: **Iberian Conference on Information Systems and Technologies, CISTI**. IEEE Computer Society, 2018. v. 2018-June, p. 1–6. ISBN 9789899843486. ISSN 21660735. Disponível em: https://www.doi.org/10.23919/CISTI.2018.8399283.

BUTLER, M. Android: Changing the mobile landscape. **IEEE Pervasive Computing**, v. 10, n. 1, p. 4–7, Jan 2011. ISSN 1558-2590.

CHARKAOUI, S.; ADRAOUI, Z.; BENLAHMAR, E. H. Cross-platform mobile development approaches. In: **Colloquium in Information Science and Technology, CIST**. Institute of Electrical and Electronics Engineers Inc., 2015. v. 2015- January, p. 188–191. ISBN 9781479959792. ISSN 23271884. Disponível em: https://www.doi.org/10.1109/CIST.2014.7016616.

CHEN, S.; WANG, R.; WANG, X.; ZHANG, K. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In: **2010 IEEE Symposium on Security and Privacy**. IEEE, 2010. p. 191–206. Disponível em: https://www.doi.org/10.1109/SP.2010.20.

CHEN, Y.; NI, J.; YU, D. Application developers' product offering strategies in multi-platform markets. **European Journal of Operational Research**, Elsevier B.V., v. 273, p. 320–333, 2 2019. ISSN 03772217.

CORBALAN, L.; FERNANDEZ, J.; CUITIO, A.; DELIA, L.; CASERES, G.; THOMAS, P.; PESADO, P. Development frameworks for mobile devices: A comparative study about energy consumption. In: **Proceedings - International Conference on Software Engineering**. IEEE Computer Society, 2018. p. 191–201. ISBN 9781450357128. ISSN 02705257. Disponível em: https://www.doi.org/10.1145/3197231.3197242.

COSTA, D. S.; COSTA, D. O.; BONIFACIO, B. A.; SOUZA, B. P. D.; FERNANDES, P. S. Using frameworks for rapid applications development as learning object for teaching web programming. In: **Proceedings - 13th Latin American Conference on Learning Technologies, LACLO 2018**. Institute of Electrical and Electronics Engineers Inc., 2018. p. 356–362. ISBN 9781728103822. Disponível em: https://www.doi.org/10.1109/LACLO.2018.00068.

DABROWSKI, J. R.; MUNSON, E. V. Is 100 milliseconds too fast? In: **CHI '01 Extended Abstracts on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2001. (CHI EA '01), p. 317–318. ISBN 1581133405. Disponível em: https://doi.org/10.1145/634067.634255.

DART. **Dart overview**. 2021. Disponível em: https://dart.dev/overview. Acesso em: 11 de julho de 2021.

DELIA, L.; GALDAMEZ, N.; THOMAS, P.; CORBALAN, L.; PESADO, P. Multi-platform mobile application development analysis. In: **Proceedings - International Conference on Research Challenges in Information Science**. IEEE Computer Society, 2015. v. 2015-June, p. 181–186. ISSN 21511357. Disponível em: https://www.doi.org/10.1109/RCIS.2015.7128878.

DIEHL, S.; GALL, H.; HASSAN, A. E.; LIBRARY., A. D. **Proceedings of the 2006 International Workshop on Mining Software Repositories : 2006, Shanghai, China, May 22-23, 2006**. New York, NY, USA: Association for Computing Machinery, 2006. 186 p. ISBN 159593085X.

EL-KASSAS, W. S.; ABDULLAH, B. A.; YOUSEF, A. H.; WAHBA, A. M. Taxonomy of cross-platform mobile applications development approaches. **Ain Shams Engineering Journal**, Ain Shams University, v. 8, p. 163–190, 6 2017. ISSN 20904479.

FERREIRA, C. M.; PEIXOTO, M. J.; DUARTE, P. A.; TORRES, A. B.; Ju´NIOR, M. L.; ROCHA, L. S.; VIANA, W. An evaluation of cross-platform frameworks for multimedia mobile applications development. **IEEE Latin America Transactions**, IEEE Computer Society, v. 16, p. 1206–1212, 4 2018. ISSN 15480992.

FLUTTER. **Flutter documentation**. 2021. Disponível em: https://flutter.dev/docs. Acesso em: 10 de julho de 2021.

FORTUNATO, D.; BERNARDINO, J. Progressive web apps: An alternative to the native mobile apps. In: **Iberian Conference on Information Systems and Technologies, CISTI**. IEEE Computer Society, 2018. v. 2018-June, p. 1–6. ISBN 9789899843486. ISSN 21660735. Disponível em: https://www.doi.org/10.23919/CISTI.2018.8399228.

GAJJAR, N.; PANDIAN, V. P. S.; SULERI, S.; JARKE, M. Akin: Generating ui wireframes from ui design patterns using deep learning. In: **International Conference on Intelligent User Interfaces, Proceedings IUI**. Association for Computing Machinery, 2021. p. 40–42. ISBN 9781450380188. Disponível em: https://www.doi.org/10.1145/3397482.3450727.

GIRI, N.; NANDGAONKAR, V.; GOSAVI, R. Virtual operating system for windows to linux migration. In: **2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)**. IEEE, 2017. p. 2125–2127. Disponível em: https://www.doi.org/10.1109/ICECDS.2017.8389825.

GITHUB. **The 2020 State of the OCTOVERSE**. 2020. Disponível em: https://octoverse.github.com. Acesso em: 17 de julho de 2021.

GIZAS, A.; CHRISTODOULOU, S.; PAPATHEODOROU, T. Comparative evaluation of javascript frameworks. In: **Proceedings of the 21st International Conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2012. (WWW '12 Companion), p. 513–514. ISBN 9781450312301. Disponível em: https://doi.org/10.1145/2187980.2188103.

GOADRICH, M. H.; ROGERS, M. P. Smart smartphone development: Ios versus android. In: **Proceedings of the 42nd ACM Technical Symposium on Computer Science Education**. New York, NY, USA: Association for Computing Machinery, 2011. (SIGCSE '11), p. 607–612. ISBN 9781450305006. Disponível em: https://doi.org/10.1145/1953163.1953330.

GOOGLE. **Conheça o Android Studio**. 2021a. Disponível em: https://developer.android.com/studio/intro. Acesso em: 20 de julho de 2021.

GOOGLE. **Executar apps no Android Emulator**. 2021b. Disponível em: https://developer.android.com/studio/run/emulator/. Acesso em: 22 de julho de 2021.

GOOGLE. **Interesse ao longo do tempo para os frameworks multiplataforma**. 2021c. Disponível em: https://trends.google.com/trends/explore?cat=31&date=2012-01- 01%202021-07-16&q=React%20Native,Flutter,Ionic,Xamarin,Vue.js. Acesso em: 16 de julho de 2021.

HARIS, N. A.; HASIM, N. Php frameworks usability in web application development. **International Journal of Recent Technology and Engineering**, Blue Eyes

Intelligence Engineering and Sciences Publication, v. 8, p. 109–116, 10 2019. ISSN 22773878.

HASSAN, H. B.; SARHAN, Q. I. Performance evaluation of graphical user interfaces in java and c. In: **2020 International Conference on Computer Science and Software Engineering (CSASE)**. IEEE, 2020. p. 290–295. Disponível em: https://www.doi.org/10.1109/CSASE48920.2020.9142075.

HEITKOTTER, H.; MAJCHRZAK, T. A.; KUCHEN, H. Cross-platform model-driven development of mobile applications with md2. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2013. (SAC '13), p. 526–533. ISBN 9781450316569. Disponível em: https://doi.org/10.1145/2480362.2480464.

HESS, R.; PAULSON, P. Linux kernel projects for an undergraduate operating systems course. In: **Proceedings of the 41st ACM Technical Symposium on Computer Science Education**. New York, NY, USA: Association for Computing Machinery, 2010. (SIGCSE '10), p. 485–489. ISBN 9781450300063. Disponível em: https://doi.org/10.1145/1734263.1734428.

HOLZER, A.; ONDRUS, J. Trends in mobile application development. In: HESSELMAN, C.; GIANNELLI, C. (Ed.). **Mobile Wireless Middleware, Operating Systems, and Applications - Workshops**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 55–64. ISBN 978-3- 642-03569-2.

HUANG, W.; MO, W.; SHEN, B.; YANG, Y.; LI, N. Automatically modeling developer programming ability and interest across software communities. In: **International Journal of Software Engineering and Knowledge Engineering**. World Scientific Publishing Co. Pte Ltd, 2016. v. 26, p. 1493–1510. ISSN 02181940. Disponível em: https://www.doi.org/10.1142/S0218194016400143.

HUANG, Y.-W.; YU, F.; HANG, C.; TSAI, C.-H.; LEE, D.-T.; KUO, S.-Y. Securing web application code by static analysis and runtime protection. In: **Proceedings of the 13th International Conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2004. (WWW '04), p. 40–52. ISBN 158113844X. Disponível em: https://doi.org/10.1145/988672.988679.

HUDLI, A.; HUDLI, S.; HUDLI, R. An evaluation framework for selection of mobile app development platform. In: **MobileDeLi 2015 - Proceedings of the 3rd International Workshop on Mobile Development Lifecycle**. Association for Computing Machinery, Inc, 2015. p. 13–16. ISBN 9781450339063. Disponível em: https://www.doi.org/10.1145/2846661.2846678.

HUY, N. P.; VANTHANH, D. Evaluation of mobile app paradigms. In: **Proceedings of the 10th International Conference on Advances in Mobile Computing amp; Multimedia**. New York, NY, USA: Association for Computing Machinery, 2012. (MoMM '12), p. 25–30. ISBN 9781450313070. Disponível em: https://doi.org/10.1145/2428955.2428968.

INTELLIPAAT. **Platform Architecture**. 2021. Disponível em: https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/. Acesso em: 28 de junho de 2021.

IONIC. **Ionic Framework**. 2021. Disponível em: https://ionicframework.com/docs/. Acesso em: 11 de julho de 2021.

ISRAELI, A.; FEITELSON, D. G. The linux kernel as a case study in software evolution. **Journal of Systems and Software**, v. 83, n. 3, p. 485–501, 2010. ISSN 0164-1212. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121209002519.

IVANOVA, S.; GEORGIEV, G. Using modern web frameworks when developing an education application: a practical approach. In: **2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. IEEE, 2019. p. 1485–1491. Disponível em: https://www.doi.org/10.23919/MIPRO.2019.8756914.

JAVASCRIPT. **JavaScript**. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 11 de julho de 2021.

JOVIC, M.; HAUSWIRTH, M. Listener latency profiling: Measuring the perceptible performance of interactive java applications. In: **Science of Computer Programming. Elsevier**, 2011. v. 76, p. 1054–1072. ISSN 01676423. Disponível em: https://www.doi.org/10.1016/j.scico.2010.04.009.

KAKAR, A. K. Why do users speak more positively about mac os x but are more loyal to windows 7? **Computers in Human Behavior**, Elsevier Ltd, v. 44, p. 166–173, 2015. ISSN 07475632.

KIM, T. S.; KIM, S. Winder: Linking speech and visual objects to support communication in asynchronous collaboration. In: **Conference on Human Factors in Computing Systems - Proceedings**. Association for Computing Machinery, 2021. ISBN 9781450380966. Disponível em: https://www.doi.org/10.1145/3411764.3445686.

KUMAR, S.; SURISETTY, S. Microsoft vs. apple: Resilience against distributed denial-of- service attacks. **IEEE Security Privacy**, v. 10, n. 2, p. 60–64, March 2012. ISSN 1558-4046.

LATIF, M.; LAKHRISSI, Y.; NFAOUI, E. H.; ES-SBAI, N. Review of mobile cross platform and research orientations. In: **2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems, WITS 2017**. Institute of Electrical and Electronics Engineers Inc., 2017. ISBN 9781509066810. Disponível em: https://www.doi.org/10.1109/WITS.2017.7934674.

LEE, J.; KIM, H.; PARK, J.; SHIN, I.; SON, S. Pride and prejudice in progressive web apps: Abusing native app-like features in web applications. In: **Proceedings of the ACM Conference on Computer and Communications Security**. Association for Computing Machinery, 2018. p. 1731–1746. ISBN 9781450356930. ISSN 15437221. Disponível em: https://www.doi.org/10.1145/3243734.3243867.

LI, N.; ZHANG, B. The research on single page application front-end development based on vue. In: **Journal of Physics: Conference Series**. IOP Publishing Ltd, 2021. v. 1883. ISSN 17426596. Disponível em: https://www.doi.org/10.1088/1742-6596/1883/1/012030.

LIN, F.; YE, W. Operating system battle in the ecosystem of smartphone industry. In: **Proceedings - 2009 International Symposium on Information Engineering and Electronic Commerce, IEEC 2009**. IEEE, 2009. p. 617–621. ISBN 9780769536866. Disponível em: https://www.doi.org/10.1109/IEEC.2009.136.

LIU, S. **Number of software developers worldwide in 2018 to 2024**. 2020. Disponível em: https://www.statista.com/statistics/627312/worldwide-developer-population/. Acesso em: 05 de outubro de 2021.

LIU, S. **Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021**. 2021. Disponível em: https://www.statista.com/statistics/869224/worldwide- software-developer-working-hours/. Acesso em: 05 de outubro de 2021.

LOWE, J. D. Mac os x tips and tricks in a lab environment. In: **Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services**. New York, NY, USA: Association for Computing Machinery, 2004. (SIGUCCS '04), p. 367–368. ISBN 1581138695. Disponível em: https://doi.org/10.1145/1027802.1027890.

MAHENDRA, M.; ANGGOROJATI, B. Evaluating the performance of android based cross- platform app development frameworks. In: **ACM International Conference Proceeding Series**. Association for Computing Machinery, 2020. p. 32–37. ISBN 9781450388092. Disponível em: https://www.doi.org/10.1145/3442555.3442561.

MARTEAU, L. Mac security: Mac os x amp; security - an overview. **Netw. Secur.**, Elsevier SciencePublishersB.V.,NLD,v.2005,n.5,p.11–13,maio2005.ISSN1353-4858.Disponível em: https://doi.org/10.1016/S1353-4858(05)70236-1.

MICROSOFT. **Bem-vindo ao IDE do Visual Studio**. 2021a. Disponível em: https://docs.microsoft.com/pt-br/visualstudio/get-started/visual-studio-ide?view=vs-2019. Acesso em: 20 de julho de 2021.

MICROSOFT. **Documentation of Visual Studio Code**. 2021b. Disponível em: https://code.visualstudio.com/docs. Acesso em: 20 de julho de 2021.

MICROSOFT. **Documentação do Xamarin.Forms**. 2021c. Disponível em: https://docs.microsoft.com/pt-br/xamarin/get-started/what-is-xamarin-forms. Acesso em: 15 de julho de 2021.

MICROSOFT. **Um tour pela linguagem C**. 2021d. Disponível em: https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/. Acesso em: 15 de julho de 2021.

MICROSOFT. **Visão geral do XAML (WPF .NET)**. 2021e. Disponível em: https://docs.microsoft.com/pt-br/dotnet/desktop/wpf/xaml/?view=netdesktop-5.0. Acesso em: 12 de outubro de 2021.

MICROSOFT. **Xamarin**. 2021f. Disponível em: https://dotnet.microsoft.com/apps/xamarin. Acesso em: 16 de julho de 2021.

NAYEEM, I.; WANT, R. Smartphones: Past, present, and future. **IEEE Pervasive Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 13, n. 04, p. 89–92, oct 2014. ISSN 1558- 2590.

NODE.JS. **About Node.js**. 2021. Disponível em: https://nodejs.org/en/about/. Acesso em: 30 de setembro de 2021.

OLHARDIGITAL. **As diferenças do modo escuro do WhatsApp para Android e iPhone**. 2021. Disponível em: https://olhardigital.com.br/2020/03/05/dicas-e-tutoriais/as-diferencas- do-modo-escuro-do-whatsapp-para-android-e-iphone/. Acesso em: 06 de julho de 2021.

OLSINA, L.; ROSSI, G. Measuring web application quality with webqem. **IEEE MultiMedia**, v. 9, n. 4, p. 20–29, Oct 2002. ISSN 1941-0166. Disponível em: https://www.doi.org/10.1109/MMUL.2002.1041945.

OSMANI, A. **Getting started with Progressive Web Apps**. 2021. Disponível em: https://addyosmani.com/blog/getting-started-with-progressive-web-apps/. Acesso em: 08 de julho de 2021.

PALOMBA, F.; TAMBURRI, D. A.; FONTANA, F. A.; OLIVETO, R.; ZAIDMAN, A.; SEREBRENIK, A. Beyond technical aspects: How do community smells influence the intensity of code smells? **IEEE Transactions on Software Engineering**, v. 47, n. 1, p. 108–129, 2021.

PERCHAT, J.; DESERTOT, M.; LECOMTE, S. Component based framework to create mobile cross-platform applications. In: **Procedia Computer Science**. Elsevier B.V., 2013. v. 19, p. 1004–1011. ISSN 18770509. Disponível em: https://www.doi.org/10.1016/j.procs.2013.06.140.

PINTO, C. M.; COUTINHO, C. From native to cross-platform hybrid development. In: JARDIM-GONC¸ALVES, R.; MENDONC¸A, J. P.; JOTSOV, V.; MARQUES, M.; MARTINS, J.; BIERWOLF, R. E. (Ed.). **9th IEEE International Conference on Intelligent Systems, IS 2018, Funchal, Madeira, Portugal, September 25-27, 2018**. IEEE, 2018. p. 669–676. Disponível em: https://doi.org/10.1109/IS.2018.8710545.

PUDER, A. Cross-compiling android applications to the iphone. In: **Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java**. New York, NY, USA: Association for Computing Machinery, 2010. (PPPJ '10), p. 69–77. ISBN 9781450302692. Disponível em: https://doi.org/10.1145/1852761.1852772.

PUDER, A.; TILLMANN, N.; MOSKAL, M. Exposing native device apis to web apps. In: **1st International Conference on Mobile Software Engineering and Systems, MOBILESoft 2014 - Proceedings**. Association for Computing Machinery, 2014. p. 18–26. ISBN 9781450328784. Disponível em: https://www.doi.org/10.1145/2593902.2593908.

QUE, P.; GUO, X.; ZHU, M. A comprehensive comparison between hybrid and native app paradigms. In: **Proceedings - 2016 8th International Conference on Computational Intelligence and Communication Networks, CICN 2016**. Institute of Electrical and Electronics Engineers Inc., 2017. p. 611–614. ISBN 9781509011445. Disponível em: https://www.doi.org/10.1109/CICN.2016.125.

RAJ, C. P. R.; TOLETY, S. B. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: **2012 Annual IEEE India**

Conference, INDICON 2012. IEEE, 2012. p. 625–629. ISBN 9781467322720. Disponível em: https://www.doi.org/10.1109/INDCON.2012.6420693.

REACTNATIVE. **ReactNative Docs.** 2021. Disponível em: https://reactnative.dev/docs/getting-started. Acesso em: 13 de julho de 2021.

REDDI, V. J.; YOON, H.; KNIES, A. Two billion devices and counting. **IEEE Micro**, v. 38, n. 1, p. 6–21, January 2018. ISSN 1937-4143. Disponível em: https://www.doi.org/10.1109/MM.2018.011441560.

RIEGER, C.; MAJCHRZAK, T. A. Towards the definitive evaluation framework for cross-platform app development approaches. **Journal of Systems and Software**, Elsevier Inc., v. 153, p. 175–199, 7 2019. ISSN 01641212. Disponível em: https://www.doi.org/10.1016/j.jss.2019.04.001.

SAMBASIVAN, D.; JOHN, N.; UDAYAKUMAR, S.; GUPTA, R. Generic framework for mobile application development. In: **Asian Himalayas International Conference on Internet**. IEEE, 2011. ISBN 9781457710872. ISSN 21570647. Disponível em: https://www.doi.org/10.1109/AHICI.2011.6113938.

SHAH, K.; SINHA, H.; MISHRA, P. Analysis of cross-platform mobile app development tools. **2019 IEEE 5th International Conference for Convergence in Technology (I2CT)**, p. 1–7, 2019. Disponível em: https://www.doi.org/10.1109/I2CT45611.2019.9033872.

STACKOVERFLOW. **Most Loved, Dreaded, and Wanted Other Frameworks, Libraries, and Tools**. 2020. Disponível em: https://insights.stackoverflow.com/survey/2020technology- most-loved-dreaded-and-wanted-other-frameworks-libraries-and-tools-loved3. Acesso em: 16 de julho de 2021.

STALLINGS, W. **Operating systems : internals and design principles**. Pearson/Prentice Hall, 2009. 822 p. ISBN 9780136006329. Disponível em: https://www.amazon.com.br/Operating-Systems-Internals-Design-Principles/dp/0133805913.

STATCOUNTER. **Desktop Operating System Market Share Worldwide**. 2021a. Disponível em: https://gs.statcounter.com/os-market-share/desktop/worldwide. Acesso em: 05 de julho de 2021.

STATCOUNTER. **Mobile Operating System Market Share Worldwide**. 2021b. Disponível em: https://gs.statcounter.com/os-market-share/mobile/worldwide. Acesso em: 23 de junho de 2021.

STEINER, T. What is in a web view: An analysis of progressive web app features when the means of web access is not a web browser. In: **The Web Conference 2018 - Companion of the World Wide Web Conference, WWW 2018**. Association for Computing Machinery, Inc, 2018. p. 789–796. ISBN 9781450356404. Disponível em: https://www.doi.org/10.1145/3184558.3188742.

SZCZEPANIK, M.; KEDZIORA, M. State management and software architecture approaches in cross-platform flutter applications. In: **ENASE 2020 - Proceedings of the 15th International Conference on Evaluation of Novel Approaches to**

Software Engineering. SciTePress, 2020. p. 407–414. ISBN 9789897584213. Disponível em: https://www.doi.org/10.5220/0009411604070414.

TAMBURRI, D. A.; KAZMAN, R.; FAHIMI, H. The architect's role in community shepherding. **IEEE Software**, v. 33, n. 6, p. 70–79, 2016. Disponível em: https://www.doi.org/10.1109/MS.2016.144.

TWITTER, I. **Twitter**. 2021a. Disponível em: https://play.google.com/store/apps/details?id=com.twitter.android&hl=pt BR&gl=US. Acesso em: 06 de julho de 2021.

TWITTER, I. **Twitter**. 2021b. Disponível em: https://apps.apple.com/br/app/twitter/id333903271. Acesso em: 06 de julho de 2021.

VDOVJAK, K.; BALEN, J.; NENADIC, K. Experimental evaluation of desktop operating systems networking performance. **International Journal of Electrical and Computer Engineering**, Ferit, v. 11, p. 67–76, 2020. Disponível em: https://www.doi.org/10.32985/ijeces.11.2.2.

VENIGALLA, A. S. M.; CHIMALAKONDA, S. Understanding emotions of developer community towards software documentation. **Cornell University**, 3 2021. Disponível em: http://arxiv.org/abs/2103.00881.

VIEIRA, R. D.; FARIAS, K. Cognide: A psychophysiological data integrator approach for visual studio code. In: **ACM International Conference Proceeding Series**. Association for Computing Machinery, 2020. p. 393–398. ISBN 9781450387538. Disponível em: https://www.doi.org/10.1145/3422392.3422453.

VITOLS, G.; SMITS, I.; BOGDANOV, O. Cross-platform solution for development of mobile applications. In: **ICEIS 2013 - Proceedings of the 15th International Conference on Enterprise Information Systems**. Scitepress, 2013. v. 2, p. 273–277. ISBN 9789898565600. Disponível em: https://www.doi.org/10.5220/0004448302730277.

VUENATIVE. **Vue Native Guide**. 2021. Disponível em: https://vue-native.io/docs/. Acesso em: 14 de julho de 2021.

WALJAS, M.; SEGERSTAHL, K.; VANANEN-VAINIO-MATTILA, K.; OINAS-KUKKONEN, H. Cross-platform service user experience: A field study and an initial framework. In: **Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services**. New York, NY, USA: Association for Computing Machinery, 2010. (MobileHCI '10), p. 219–228. ISBN 9781605588353. Disponível em: https://doi.org/10.1145/1851600.1851637.

WANT, R. iphone: Smarter than the average phone. **IEEE Pervasive Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 9, n. 03, p. 6–9, jul 2010. ISSN 1558-2590. Disponível em: https://www.doi.org/10.1109/MPRV.2010.62.

WAREN, J. Cross-platform mobile software development with react native. **Theseus**, 2016. Disponível em: https://urn.fi/URN:NBN:fi:amk-2016120919690.

WASILEWSKI, K.; ZABIEROWSKI, W. A comparison of java, flutter and kotlin/native technologies for sensor data-driven applications. **Sensors**, MDPI AG, v. 21, 5 2021. ISSN 14248220. Disponível em: https://www.doi.org/10.3390/s21103324.

WASSERMAN, A. I. Software engineering issues for mobile application development. In: **Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research**. New York, NY, USA: Association for Computing Machinery, 2010. (FoSER '10), p. 397–400. ISBN 9781450304276. Disponível em: https://doi.org/10.1145/1882362.1882443.

WILLOCX, M.; VOSSAERT, J.; NAESSENS, V. A quantitative assessment of performance in mobile app development tools. In: **2015 IEEE International Conference on Mobile Services**. IEEE, 2015. p. 454–461. Disponível em: https://www.doi.org/10.1109/MobServ.2015.68.

WILLOCX, M.; VOSSAERT, J.; NAESSENS, V. Comparing performance parameters of mobile app development strategies. In: **Proceedings - International Conference on Mobile Software Engineering and Systems, MOBILESoft 2016**. Association for Computing Machinery, Inc, 2016. p. 38–47. ISBN 9781450341783. Disponível em: https://www.doi.org/10.1145/2897073.2897092.

WU, W. React native vs flutter, cross-platform mobile application frameworks. **Theseus**, 2018. Disponível em: https://urn.fi/URN:NBN:fi:amk-201805158156.

XIAO, G.; ZHENG, Z.; WANG, H. Evolution of linux operating system network. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 466, p. 249–258, 1 2017. ISSN 03784371. Disponível em: https://www.doi.org/10.1016/j.physa.2016.09.021.

YANG, S.; YAN, D.; ROUNTEV, A. Testing for poor responsiveness in android applications. In: **2013 1st International Workshop on the Engineering of Mobile-Enabled Systems, MOBS 2013 - Proceedings**. IEEE Computer Society, 2013. p. 1–6. ISBN 9781467363334. Disponível em: https://www.doi.org/10.1109/MOBS.2013.6614215.

ZHADCHENKO, A. V.; MAMROSENKO, K. A.; GIATSINTOV, A. M. Porting x windows system to operating system compliant with portable operating system interface. **International Journal of Advanced Computer Science and Applications**, The Science and Information Organization, v. 11, n. 7, 2020. Disponível em: http://dx.doi.org/10.14569/IJACSA.2020.0110703.

ZHOU, X.; HU, W.; LIU, G.-P. React-native based mobile app for online experimentation. In: **2020 39th Chinese Control Conference (CCC)**. IEEE, 2020. p. 4400–4405. Disponível em: https://www.doi.org/10.23919/CCC50068.2020.9189636.

anexo A - Direitos autorais - Lei n. 9.610, de 19 de fevereiro de 1998

Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos

LEI Nº 9.610, DE 19 DE FEVEREIRO DE 1998[67].

Mensagem de veto

Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.

O PRESIDENTE DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Título I - Disposições Preliminares

Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

Art. 2º Os estrangeiros domiciliados no exterior gozarão da proteção assegurada nos acordos, convenções e tratados em vigor no Brasil.

Parágrafo único. Aplica-se o disposto nesta Lei aos nacionais ou pessoas domiciliadas em país que assegure aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes.

Art. 3º Os direitos autorais reputam-se, para os efeitos legais, bens móveis.

Art. 4º Interpretam-se restritivamente os negócios jurídicos sobre os direitos autorais.

Art. 5º Para os efeitos desta Lei, considera-se:

I - publicação - o oferecimento de obra literária, artística ou científica ao conhecimento do público, com o consentimento do autor, ou de qualquer outro titular de direito de autor, por qualquer forma ou processo;

II - transmissão ou emissão - a difusão de sons ou de sons e imagens, por meio de ondas radioelétricas; sinais de satélite; fio, cabo ou outro condutor; meios óticos ou qualquer outro processo eletromagnético;

III - retransmissão - a emissão simultânea da transmissão de uma empresa por outra;

IV - distribuição - a colocação à disposição do público do original ou cópia de obras literárias, artísticas ou científicas, interpretações ou execuções fixadas e fonogramas, mediante a venda, locação ou qualquer outra forma de transferência de propriedade ou posse;

V - comunicação ao público - ato mediante o qual a obra é colocada ao alcance do público, por qualquer meio ou procedimento e que não consista na distribuição de exemplares;

VI - reprodução - a cópia de um ou vários exemplares de uma obra literária, artística ou científica ou de um fonograma, de qualquer forma tangível, incluindo qualquer armazenamento permanente ou temporário por meios eletrônicos ou qualquer outro meio de fixação que venha a ser desenvolvido;

VII - contrafação - a reprodução não autorizada;

VIII - obra:

a) em co-autoria - quando é criada em comum, por dois ou mais autores;

b) anônima - quando não se indica o nome do autor, por sua vontade ou por ser desconhecido;

c) pseudônima - quando o autor se oculta sob nome suposto;

d) inédita - a que não haja sido objeto de publicação;

e) póstuma - a que se publique após a morte do autor;

f) originária - a criação primígena;

g) derivada - a que, constituindo criação intelectual nova, resulta da transformação de obra originária;

h) coletiva - a criada por iniciativa, organização e responsabilidade de uma pessoa física ou jurídica, que a publica sob seu nome ou marca e que é constituída pela participação de diferentes autores, cujas contribuições se fundem numa criação autônoma;

i) audiovisual - a que resulta da fixação de imagens com ou sem som, que tenha a finalidade de criar, por meio de sua reprodução, a impressão de movimento, independentemente dos processos de sua captação, do suporte usado inicial ou posteriormente para fixá-lo, bem como dos meios utilizados para sua veiculação;

IX - fonograma - toda fixação de sons de uma execução ou interpretação ou de outros sons, ou de uma representação de sons que não seja uma fixação incluída em uma obra audiovisual;

X - editor - a pessoa física ou jurídica à qual se atribui o direito exclusivo de reprodução da obra e o dever de divulgá-la, nos limites previstos no contrato de edição;

XI - produtor - a pessoa física ou jurídica que toma a iniciativa e tem a responsabilidade econômica da primeira fixação do fonograma ou da obra audiovisual, qualquer que seja a natureza do suporte utilizado;

XII - radiodifusão - a transmissão sem fio, inclusive por satélites, de sons ou imagens e sons ou das representações desses, para recepção ao público e a transmissão de sinais codificados, quando os meios de decodificação sejam oferecidos ao público pelo organismo de radiodifusão ou com seu consentimento;

XIII - artistas intérpretes ou executantes - todos os atores, cantores, músicos, bailarinos ou outras pessoas que representem um papel, cantem, recitem, declamem, interpretem ou executem em qualquer forma obras literárias ou artísticas ou expressões do folclore.

Art. 6º Não serão de domínio da União, dos Estados, do Distrito Federal ou dos Municípios as obras por eles simplesmente subvencionadas.

[67] Disponível em: http://www.planalto.gov.br/ccivil_03/leis/l9610.htm.