

Universidade Federal de São Carlos – UFSCar

Departamento de Computação

Programa de Pós-Graduação

Felipe Munhoz Afonso

# **Cr terios para Ado  o de Solu  es de Desenvolvimento Multiplataforma M vel na Perspectiva de Desenvolvedores de Software**

S o Carlos - SP

Abril/2020



Felipe Munhoz Afonso

**Cr terios para Ado o de Solu es de Desenvolvimento  
Multiplataforma M vel na Perspectiva de  
Desenvolvedores de Software**

Defesa de Disserta o apresentada ao Programa de P s-Gradua o em Ci ncia da Computa o da Universidade Federal de S o Carlos, como parte dos requisitos para a obten o do t tulo de Mestre em Ci ncia da Computa o,  rea de concentra o: Engenharia de Software.

Orientador: Prof. Dr. Fabiano Cutigi Ferrari

S o Carlos - SP

Abril/2020





## UNIVERSIDADE FEDERAL DE SÃO CARLOS

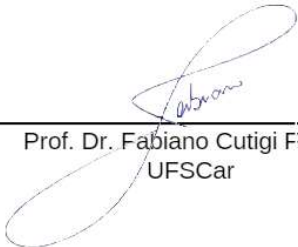
Centro de Ciências Exatas e de Tecnologia  
Programa de Pós-Graduação em Ciência da Computação

---

### Folha de Aprovação

---

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Felipe Munhoz Afonso, realizada em 03/04/2020:



---

Prof. Dr. Fabiano Cutigi Ferrari  
UFSCar

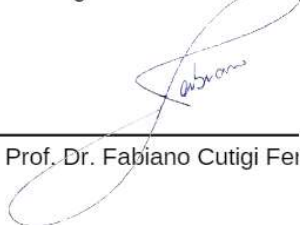
---

Profa. Dra. Vânia Paula de Almeida Neris  
UFSCar

---

Prof. Dr. Windson Viana de Carvalho  
UFC

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Vânia Paula de Almeida Neris, Windson Viana de Carvalho e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ão) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.



---

Prof. Dr. Fabiano Cutigi Ferrari



# Agradecimentos

Agradeço primeiramente às instituições e orientadores que me inseriram no meio acadêmico. Tudo começou no IFSP de Piracicaba, aonde tive meu primeiro contato com a ciência, primeiro através do Neto e em seguida o Juliano. E a partir deste interesse que nunca parou de crescer, cheguei até a UFSCar, aonde fui acolhido por meu orientador, Daniel, que me recebeu e guiou de maneira exímia. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Agradeço meus colegas do mestrado, Gabriel e Dennys por nossa amizade, independente do sucesso ou fracasso. Aos meus professores, tanto do IFSP quanto da UFSCar, por me iniciar na computação: Barreto, Carlos, Delano, Diego, Estevam, Fabiano, Heloísa, Luiz, Michel, Sérgio e Thiago. Aos meus amigos de minha cidade natal, por aventuras épicas e histórias descontraídas.

Agradeço à meu pai, por desde cedo me colocar na frente de um computador, e me ensinar a aprender, habilidade mais fundamental que considero possuir. À minha mãe, por acreditar em mim o tempo todo. E principalmente à minha namorada, Jacqueline, por ser minha motivação para me empenhar além do que eu jamais imaginei que poderia.





*"Eu posso não ter ido para onde eu pretendia ir, mas eu acho que acabei terminando onde eu pretendia estar"*

- Douglas Adams



# Resumo

Ferramentas de Desenvolvimento Multiplataforma são uma tecnologia que surgiram para suprir a demanda de equipes de desenvolvimento que operam com mais de uma plataforma móvel simultaneamente. Esta tecnologia beneficia principalmente pequenas e médias equipes que não possuíam capacidade de manter duas ou mais bases de códigos diferentes para o mesmo projeto. O uso destas ferramentas tem se tornado cada vez mais popular de acordo com a evolução da tecnologia empregada capaz de produzir aplicativos cada vez mais próximos a um desenvolvido nativamente. À medida que a popularidade de ferramentas multiplataforma cresce, também aumenta a necessidade de uma maneira concisa, flexível e duradoura de definir suas vantagens e desvantagens perante outras ferramentas. Portanto o objetivo deste trabalho é descobrir através de métodos mistos de coletas de dados como questionários, entrevistas e estudos práticos, o que realmente importa para os desenvolvedores, contribuindo com a evolução do estado da arte ao fornecer um conjunto de critérios que poderão ser utilizados para realizar comparações. Também é agregada ao estado da arte uma comparação entre as ferramentas de desenvolvimento mais utilizadas atualmente. Os resultados obtidos demonstram que Responsividade, Viabilidade a Longo Prazo e Manutenibilidade são os fatores de decisão mais importantes. Também foi verificado que as ferramentas mais completas são React Native, Flutter e Xamarin.

**Palavras-chaves:** Desenvolvimento multiplataforma, Desenvolvimento móvel, Questionário, Entrevista Intensiva, Diário de Bordo.



# Abstract

Cross-platform Development Tools are a technology that emerged to meet the demand of development teams that operate simultaneously with more than one mobile platform. This technology mainly benefits small and medium-sized teams that did not have the capacity to maintain two or more different code bases for the same project. The use of these tools has become more and more popular according to the evolution of the technology employed capable of producing applications increasingly closer to one developed natively. As the popularity of cross-platform tools grows, so does the need for a concise, flexible and long-lasting way of defining their advantages and disadvantages over other tools. So the objective of this work is to find out through mixed methods of data gathering such as questionnaires, interviews and practical studies what really matters to the developers, thus contributing to the evolution of the state of the art by providing a set of criteria that can be used to make comparisons. And enhancing the state of the art with a comparison between the development tools most used today. Results shows that the most important decision factors are Responsivity, Long Term Viability and Maintainability. It was also indentified that the most complete tools are React Native, Flutter and Xamarin.

**Key-words:** Cross-platform development, Mobile development, Questionnaire, Intensive Interview, Logbook.



# Lista de ilustrações

Figura 1 – Fatia de mercado de sistemas operacionais (STATISTA; IDC, 2018) . . .	21
Figura 2 – Número de usuários de <i>smartphones</i> (em bilhões) (STATISTA; EMARKETER, 2016) . . . . .	22
Figura 3 – Número de Aplicativos disponíveis para <i>iOS</i> e <i>Android</i> (STATISTA et al., 2017; STATISTA; APPLE, 2017) . . . . .	22
Figura 4 – Procura de algumas ferramentas no <i>site</i> de buscas <i>Google</i> , por interesse ao longo do tempo . . . . .	24
Figura 5 – Modelo da Abordagem <i>Web</i> (RAJ; TOLETY, 2012) . . . . .	30
Figura 6 – Modelo da Abordagem Híbrida (RAJ; TOLETY, 2012) . . . . .	32
Figura 7 – Modelo da Abordagem Interpretada (RAJ; TOLETY, 2012) . . . . .	33
Figura 8 – Modelo da Abordagem Generativa (RAJ; TOLETY, 2012) . . . . .	35
Figura 9 – Principais elementos do MDD (LUCRÉDIO, 2009) . . . . .	36
Figura 10 – Modelo geral da metodologia utilizada . . . . .	40
Figura 11 – Comparação de interfaces de usuário dos aplicativos produzidos. . . . .	62
Figura 12 – Distribuição de respostas sobre quais os critérios mais relevantes para o desenvolvimento multiplataforma. ( <b>n=95</b> ) . . . . .	67
Figura 13 – Distribuição de respostas de avaliação do Flutter. ( <b>n=12</b> ) . . . . .	75
Figura 14 – Distribuição de respostas de avaliação do Ionic. ( <b>n=7</b> ) . . . . .	76
Figura 15 – Distribuição de respostas de avaliação do Kotlin. ( <b>n=2</b> ) . . . . .	77
Figura 16 – Distribuição de respostas de avaliação do Qt. ( <b>n=6</b> ) . . . . .	78
Figura 17 – Distribuição de respostas de avaliação do React Native. ( <b>n=15</b> ) . . . . .	80
Figura 18 – Distribuição de respostas de avaliação do desenvolvimento web. ( <b>n=7</b> ) . . . . .	81
Figura 19 – Distribuição de respostas de avaliação do Xamarin. ( <b>n=11</b> ) . . . . .	82
Figura 20 – Mapa mental com a classificação dos critérios descobertos durante a codificação . . . . .	91





# Lista de tabelas

Tabela 1 – Fatores de Decisão de Dalmaso et al. (2013). . . . .	53
Tabela 2 – Identificadores atribuídos aos critérios . . . . .	65
Tabela 3 – Critérios ordenados por índice de concordância . . . . .	72
Tabela 4 – Identificadores atribuídos às questões pertinentes as ferramentas . . . . .	74
Tabela 5 – Qualidades e Défcits de cada ferramenta presente no questionário . . . . .	83
Tabela 6 – Artefatos utilizados na codificação e identificador atribuído . . . . .	87
Tabela 7 – Porcentagem de aparecimento de cada código de critério . . . . .	90
Tabela 8 – Quantidade de códigos encontrados para Flutter . . . . .	92
Tabela 9 – Quantidade de códigos encontrados para Ionic . . . . .	93
Tabela 10 – Quantidade de códigos encontrados para MD <sup>2</sup> . . . . .	93
Tabela 11 – Quantidade de códigos encontrados para desenvolvimento Nativo . . . . .	94
Tabela 12 – Quantidade de códigos encontrados para Phonegap . . . . .	94
Tabela 13 – Quantidade de códigos encontrados para Web Apps e PWA . . . . .	95
Tabela 14 – Quantidade de códigos encontrados para React Native . . . . .	95
Tabela 15 – Quantidade de códigos encontrados para Xamarin . . . . .	96
Tabela 16 – Qualidades e Défcits de cada ferramenta presente na codificação . . . . .	97
Tabela 17 – Resumo e conclusão dos critérios . . . . .	106
Tabela 18 – Resumo e conclusão das ferramentas . . . . .	109



## Acrónimos

**API** *Application Program Interface*

**CIM** *Computation Independent Model*

**CSS** *Cascading Style Sheets*

**DSL** *Domain Specific Language*

**ERP** *Enterprise Resource Planning*

**GT** *Grounded Theory*

**HTML** *Hyper-Text Markup Language*

**IDE** *Integrated Development Environment*

**MDD** *Model Driven Development*

**MVP** *Producto Viável Mínimo*

**PIM** *Platform Independent Model*

**PSM** *Platform Specific Model*

**PWA** *Progressive Web App*

**UI** *User Interface*

**UML** *Unified Modeling Language*

**UX** *User Experience*



# Sumário

<b>1</b>	<b>Introdução</b>	<b>21</b>
1.1	Motivação	23
1.2	Objetivos	26
1.3	Metodologia	27
1.4	Resultados e contribuições alcançadas	27
1.5	Organização da dissertação	28
<b>2</b>	<b>Fundamentação Teórica</b>	<b>29</b>
2.1	Abordagem Web	29
2.2	Abordagem Híbrida	31
2.3	Abordagem Interpretada	33
2.4	Abordagem Generativa	34
2.5	Abordagem Dirigida por Modelos	35
<b>3</b>	<b>Metodologia</b>	<b>39</b>
3.1	Ferramentas de Pesquisa	39
3.1.1	Entrevista Intensiva	40
3.1.2	Diário de Bordo	41
3.1.3	Codificação	41
3.1.4	Questionários	42
3.2	Revisão Bibliográfica	42
3.3	<i>Proof of Concept</i>	43
3.4	Entrevista Intensiva	43
3.5	Questionário	45
3.6	Literatura Cinza	49
3.7	Pesquisa de Campo	49
3.8	Análise Final	50
<b>4</b>	<b>Trabalhos Relacionados</b>	<b>51</b>
4.1	Metodologia	51
4.2	Comparação generalizada de ferramentas	51
4.3	Comparação específica de ferramentas	58
4.4	Estudo sobre ferramenta e estudos de caso	58
4.5	Discussão final	59
<b>5</b>	<b>Resultados</b>	<b>61</b>

5.1	<i>Proof of Concept</i>	61
5.2	Questionário	64
5.2.1	Flutter	73
5.2.2	Ionic	75
5.2.3	Kotlin	77
5.2.4	Qt	78
5.2.5	React Native	79
5.2.6	Desenvolvimento Web	80
5.2.7	Xamarin	81
5.3	Codificação	83
5.4	Pesquisa de Campo	97
5.4.1	Transformação para <i>Web App</i>	98
5.4.2	Transformação para <i>Progressive Web App</i> (PWA)	99
5.4.3	Criação do aplicativo multiplataforma	100
5.5	Análise de correlação entre os diferentes resultados	102
5.5.1	Sobre critérios e fatores de decisão	102
5.5.2	Sobre as ferramentas	106
<b>6</b>	<b>Conclusão</b>	<b>111</b>
6.1	Contribuições	111
6.2	Limitações	111
6.3	Ameaças à validade	112
6.3.1	<i>Proof of Concept</i>	112
6.3.2	Questionário	112
6.3.3	Codificação	112
6.3.4	Pesquisa de campo	113
6.4	Trabalhos Futuros	113
6.5	Considerações Finais	114
	<b>Referências</b>	<b>115</b>

# 1 Introdução

O mercado de dispositivos móveis sempre esteve em constante mudanças (RAJ; TOLETY, 2012). Tamanho volatilidade reflete nos desenvolvedores a necessidade de uma grande capacidade de adaptação para desenvolver seus aplicativos em variadas plataformas. Em contrapartida, desde 2011, certo favoritismo começou a prevalecer, e *iOS* e *Android* se tornaram o padrão para sistemas operacionais, eliminando cada vez mais a competição, fato que pode ser verificado na Figura 1. Segundo o estudo de Richter (2018), este favoritismo tornou o mercado um duopólio.

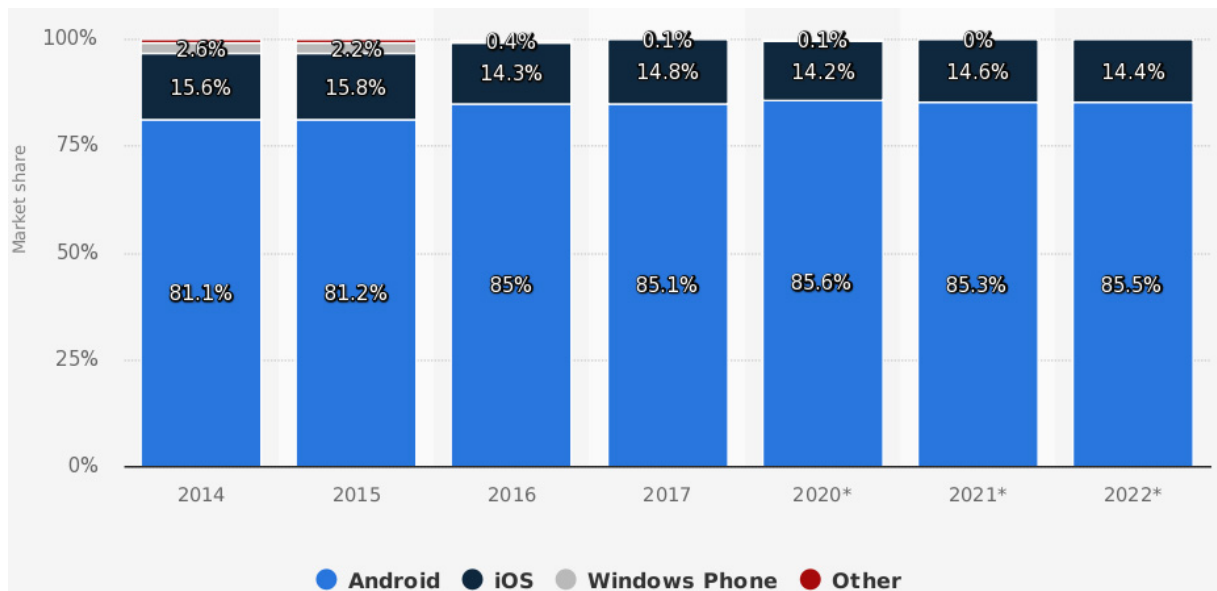


Figura 1 – Fatia de mercado de sistemas operacionais (STATISTA; IDC, 2018)

A Figura 1 apresenta a fatia de mercado entre os sistemas operacionais móveis mais populares. Ela representa a distribuição de sistemas operacionais utilizados em *smartphones* em atividade no período de 2014 a 2022, sendo de 2014 a 2017 baseados em dados reais e 2020 a 2022 uma estimativa.

Não foi apenas sua fatia de mercado que cresceu. Com a redução dos custos e avanços tecnológicos, a popularidade de *smartphones* também cresceu significativamente. As Figuras 2 e 3 demonstram a taxa de seus usuários e a quantidade de aplicativos disponíveis.

A partir destas estatísticas é possível perceber não apenas a crescente presença da tecnologia móvel, mas também a diferença entre essas duas principais concorrentes. Conforme mostra a Figura 3, ainda há uma margem de aplicativos que não são disponíveis para as duas maiores lojas de aplicativos. Os desenvolvedores de aplicativos que optam por permanecer em apenas uma plataforma acabam perdendo uma potencial fatia do mercado,

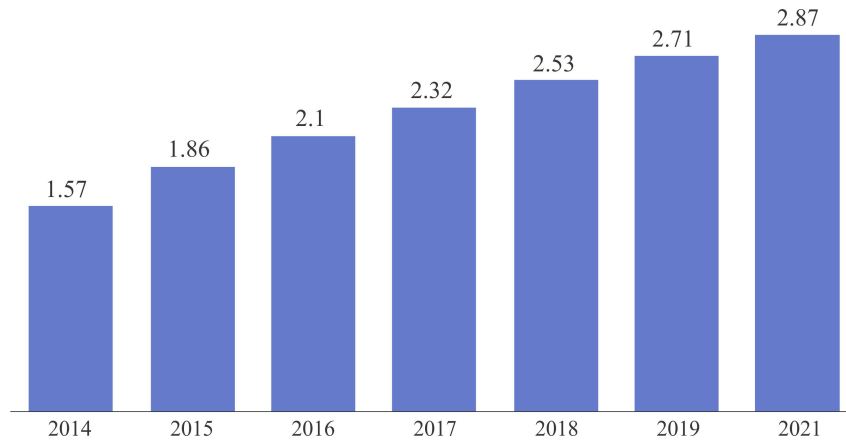


Figura 2 – Número de usuários de *smartphones* (em bilhões) (STATISTA; EMARKETER, 2016)

reduzindo seu alcance de usuários. Um dos fatores que influenciam neste resultado é a falta de disponibilidade de tempo para pequenas equipes de desenvolvimento criarem e atualizarem aplicativos em múltiplas plataformas, devido à necessidade de manter dois sistemas, com linguagens e peculiaridades diferentes.

O desenvolvimento de sistemas não trata mais apenas de desenvolver um software para apenas um dispositivo. Existem diversas outras formas de acessar um sistema virtual que não seja pelo tradicional computador de mesa. Uma das formas mais comuns para atingir um maior número de plataformas sem replicar o sistema para diferentes dispositivos é através da abordagem *web*. Nesta abordagem é criado um sistema que normalmente utiliza *HTML*, *CSS* e *JavaScript* para ser acessado em diferentes plataformas através de um navegador. No entanto, esta abordagem acaba apresentando algumas desvantagens, como o aplicativo não poder ser adquirido através da loja e não aproveitar 100%

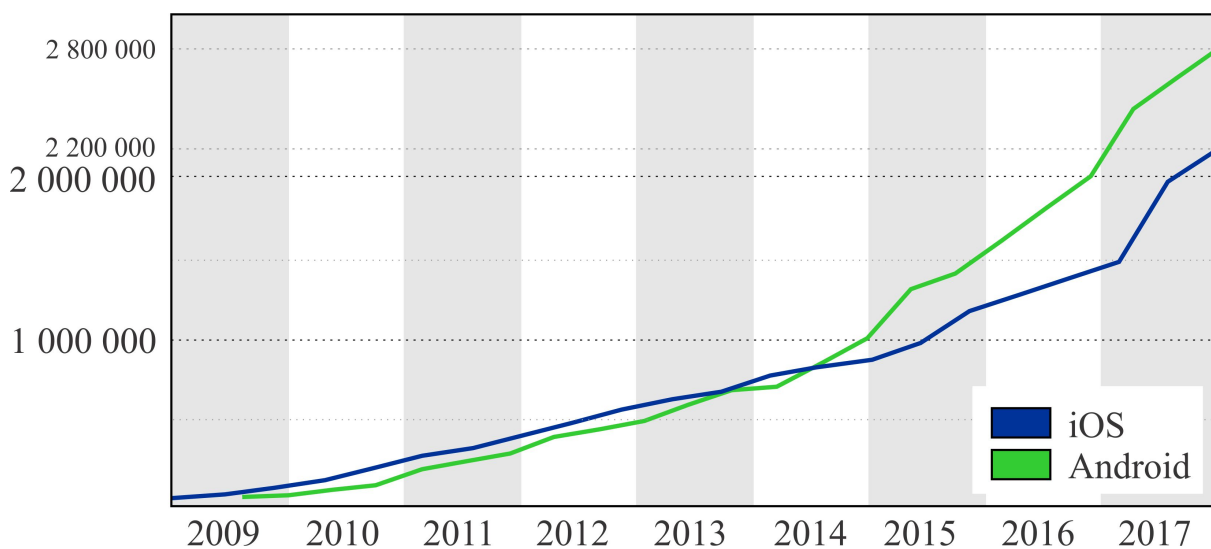


Figura 3 – Número de Aplicativos disponíveis para *iOS* e *Android* (STATISTA et al., 2017; STATISTA; APPLE, 2017)



das capacidades gráficas do dispositivo (BERNARDES; MIYAKE, 2016; HEITKOTTER; HANSCHKE; MAJCHRZAK, 2012; CHARKAOUI; ADRAOUI; Habib Benlahmar, 2014; RAJ; TOLETY, 2012). Uma discussão mais detalhada é apresentada na seção 2.1 deste trabalho.

Uma alternativa à abordagem *web* é o uso de ferramentas de desenvolvimento multiplataforma. Estas também permitem uma redução do tempo de desenvolvimento, mas mantêm acesso a recursos nativos, loja de aplicativos e desempenho independente de conexão. Porém, as ferramentas de desenvolvimento multiplataforma não estão livres de pontos negativos. Como Mercado, Munaiah e Meneely (2016) descrevem, usuários geralmente sentem certa perda de qualidade em aplicativos desenvolvidos por este tipo de ferramenta em relação às abordagens nativas. Uma descrição mais apurada sobre estas abordagens é realizada no Capítulo 2.

Com um mercado em constante expansão, as ferramentas de desenvolvimento multiplataforma se diversificaram, há diversas abordagens que produzem resultados diferentes. A literatura, predominantemente, lista como principais abordagens: híbrida, interpretada, generativa e dirigida por modelos (BERNARDES; MIYAKE, 2016; HEITKOTTER; HANSCHKE; MAJCHRZAK, 2012; DALMASSO et al., 2013; CHARKAOUI; ADRAOUI; Habib Benlahmar, 2014; MAJCHRZAK; BIORN-HANSEN; GRONLI, 2017; LATIF et al., 2016; PALMIERI; SINGH; CICCETTI, 2012; RIBEIRO; Da Silva, 2012). Com tantas opções para desenvolver um aplicativo, resta a dúvida: se todas têm suas vantagens, desvantagens e casos de sucesso, quando usar cada uma delas?

## 1.1 Motivação

Ferramentas de desenvolvimento multiplataforma estão se popularizando. Isto pode ser constatado de diversas maneiras. Em primeiro lugar, pelo maior número de aplicativos que são construídos utilizando estas ferramentas, conforme é divulgado nos próprios *sites* dos fabricantes. No entanto é necessário ter cuidado, pois esses dados podem ser enviesados, já que são fornecidos pelos principais interessados. Outro fator considerado é que a partir de uma simples consulta em *sites* de busca como o *Google* (observado na Figura 4) é encontrado um volume expressivo de procura por *frameworks* multiplataformas.

A Figura 4 mostra no eixo Y o interesse, este valor representa a popularidade dos termos relativos uns aos outros, sendo 100 o pico de popularidade entre todos os termos exibidos. A figura mostra apenas as ferramentas mais procuradas no momento da consulta: *React Native*, *Xamarin*, *Flutter* e *Ionic*. No entanto, existem dezenas de outras opções, que apesar de não serem tão populares como as quatro citadas, oferecem recursos para facilitar o desenvolvimento multiplataforma. Entre estas opções, pode-se citar *Applause*, *Codename One*, *Corona*, *DragonRad*, *Haxe*, *IBM Worklight*, *iPhonical*, *JMango*,

*Kotlin/Native, LiveCode, Marmalade, MD<sup>2</sup>, Mono, MoSync, Neomades, Qt, PhoneGap, Rhomobile, Sencha Touch, Titanium, Unity, Vue.js, XMLVM e Weex.* Algumas destas ferramentas tem propostas interessantes e podem vir a se tornar populares no futuro próximo.

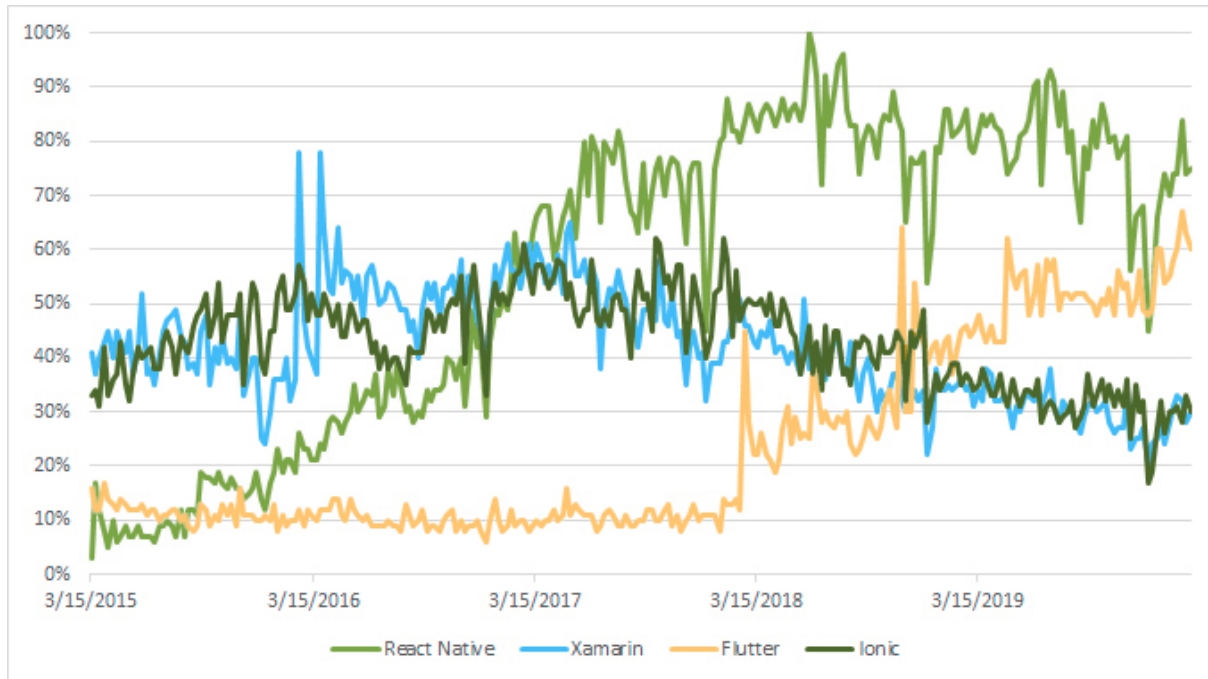


Figura 4 – Procura de algumas ferramentas no *site* de buscas *Google*, por interesse ao longo do tempo

Neste contexto, as opções são variadas. Cada opção traz vantagens e desvantagens que são inerentes à abordagem empregada, e mesmo entre as mesmas abordagens, ferramentas apresentam diferenças de funcionamento ou desenvolvimento. Dependendo das características do projeto, nem sempre a opção mais óbvia é a melhor. Existem diversos fatores envolvidos, desde os requisitos técnicos, que demandam peculiaridades do aplicativo multiplataforma, até não-técnicos, como restrições de equipe e tempo.

Para facilitar a compreensão deste problema, foram elaborados os possíveis cenários a seguir. As ferramentas sugeridas em cada cenário foram selecionadas a partir dos resultados observados durante a pesquisa.

**Cenário 1:** Uma equipe formando uma empresa *startup*, com pouca experiência, quer desenvolver um aplicativo que realize videoconferências. A empresa terá um ciclo de desenvolvimento longo, então demorará para lançar. Como os desenvolvedores não possuem conhecimento sobre o desenvolvimento multiplataforma, uma abordagem que tenha uma curva de aprendizado mais íngreme é o ideal. A possibilidade de customizar a câmera e exibir a imagem em tempo real dentro do aplicativo também não pode ser esquecida, pois não são todas as ferramentas que permitem tal liberdade. E como este projeto ficará congelado por certo tempo, a ferramenta deve possuir uma boa viabilidade

a longo termo. Neste caso específico, a ferramenta *React Native* pode ser uma boa opção, pois ela: tem o suporte à exibição da câmera em tempo real, um requisito técnico; tem documentação adequada e comunidade ativa, o que facilita o aprendizado; e já tem certa estabilidade, facilitando a longevidade do projeto.

**Cenário 2:** Uma empresa, que já possui um sistema *Enterprise Resource Planning* (ERP), utilizando de uma interface *web* em que um visual nativo não é prioridade, deseja transformar este *website* em um aplicativo que funcione sem conexão com a *Internet*. Por já ter um sistema *web*, evitar ter o retrabalho de começar o aplicativo do zero seria muito mais adequado. Portanto as opções híbridas se adequariam muito bem nesta situação. Neste caso, o uso do *Ionic* poderia realizar esta conversão exigindo pouco trabalho para os desenvolvedores. Além disso, como o visual nativo não é prioridade, as eventuais perdas de qualidade das abordagens híbridas não seriam tão relevantes.

**Cenário 3:** Um desenvolvedor solitário deseja criar um jogo, que não exija tanto da capacidade do dispositivo nem de sua bateria, também não requer gráficos que demandariam muito poder de processamento. O uso de *game engines*, como o *Unity*, pode aumentar o consumo de recursos de um jogo, portanto não seria a solução mais adequada. Neste caso o *Haxe* seria uma opção adequada, pois gera aplicativos nativos e possui suporte e uma comunidade ativa para desenvolvedores de jogos.

**Cenário 4:** Uma empresa possui um aplicativo para Android. No entanto, após o lançamento, percebeu que este não teve uma boa aderência de seu público alvo. Após uma análise, percebeu-se que muitos de seus clientes são usuários do iOS. Utilizar de uma abordagem multiplataforma é a melhor opção nesta situação, já que os desenvolvedores não querem manter duas bases de código, mas qual abordagem escolher? O fato de já possuir o código das *User Interfaces* (UIs) de uma das plataformas pode ser um incentivo para utilizar *frameworks* interpretados, que irá requerer apenas reescrever parte do projeto, mas continuará fornecendo uma experiência nativa. Também vale lembrar que nesta abordagem há varias ferramentas que podem ser empregadas, algumas usando linguagens de alto nível, que não exigirão tanto esforço de conversão a partir do código *Java*.

**Cenário 5:** Uma pequena equipe em uma grande empresa de desenvolvimento necessita entregar um Produto Viável Mínimo (MVP) para iniciar uma nova linha de produtos. A empresa já possui um ecossistema que possui uma *Application Program Interface* (API) interna que será reutilizada. Além disso o MVP servirá principalmente para prova de conceito, portanto pode vir a ser descartado no futuro próximo. Neste cenário, uma ferramenta de rápida prototipação, mesmo que não seja tão estável, pode ser o mais apropriado. Mais especificamente, uma ferramenta híbrida permite um desenvolvimento rápido da interface e também a integração com sua API, ou até mesmo um PWA.

Estes cenários exemplificam a diversidade de fatores envolvidos, ampla disponibilidade de opções, o que eleva a complexidade na tomada da decisão. Para tomar uma

decisão correta, o desenvolvedor precisa ser capaz de compreender os detalhes de cada solução para conseguir adotar uma que seja apropriada para cada projeto.

O que existe à disposição para o desenvolvedor, atualmente, é uma série de informações esparsas e nem sempre confiáveis.

- Existem blogs e sites voltados ao desenvolvimento, muitos deles apresentando guias e estudos comparativos entre as opções do mercado. Estes geralmente apresentam os pontos de vista de um especialista na área, que expõe sua opinião pessoal. Apesar de muitas vezes essas opiniões serem coerentes e úteis, são subjetivas e nem sempre consideram todos os fatores ou ferramentas existentes;
- Existem os *sites* dos fabricantes, porém as comparações e guias, se existirem, serão tendenciosas por natureza e não podem ser consideradas como um guia definitivo. Também é possível mergulhar na documentação oficial, realizar testes e estudos com a própria ferramenta. Porém isso demanda tempo e exige dedicação demorada, o que nem sempre é uma opção viável;
- Existem estudos na literatura acadêmica que fazem a comparação de uma forma mais rigorosa e científica. Um levantamento apresentado no Capítulo 4 ilustra estudos que seguem essa linha. Porém, a grande maioria adota um ponto de vista puramente acadêmico, sem considerar visões dos profissionais, e muitas vezes desprezando fatores importantes. Além disso, muitos possuem foco nas tecnologias e não nos conceitos, tornando os estudos rapidamente desatualizados. Por fim, alguns não adotam o rigor científico necessário para que a informação apresentada seja confiável.

Dadas essas dificuldades, a tomada de decisão sofre de incertezas e riscos que podem levar projetos ao fracasso. Também pode-se citar a dificuldade enfrentada por pesquisadores da área, que precisam ser capazes de filtrar conceitos e lacunas de pesquisa em meio a uma grande quantidade de conhecimento difuso e não estruturado.

## 1.2 Objetivos

**Considerando os conhecimentos da área de desenvolvimento multiplataforma móvel...** Já existe muita informação sobre o tema espalhada em forma de trabalhos científicos, artigos de blog e vídeos. No entanto nem todas informações são de fato relevantes para os praticantes da área. Portanto uma das preocupações deste trabalho foi levar em consideração a relevância das informações coletadas; e

**E considerando a visão da comunidade científica e dos profissionais...** Uma das demandas observadas durante a primeira iteração da revisão bibliográfica é

por uma maior participação dos profissionais. Com um tema tão íntimo ao mercado de trabalho a opinião dos que possuem experiências profissionais com as ferramentas são de extremo interesse para a academia e indústria.

A partir destas duas premissas, os objetivos deste trabalho são:

- **Estabelecer critérios relevantes para os desenvolvedores...** Ajudando pesquisadores interessados em métricas para realizar uma reavaliação de ferramentas. Mantendo o estado da arte mais orientado com o real interesse dos profissionais que utilizam de ferramentas de desenvolvimento multiplataforma; e
- **Comparar ferramentas utilizando dos critérios estabelecidos...** Ajudando os desenvolvedores a selecionar ferramentas mais adequadas ao seu projeto, através de comparações diretas com base em dados qualitativos e quantitativos de diversas fontes.

### 1.3 Metodologia

Para alcançar estes objetivos de forma que integra a opinião do profissional e do acadêmico foi necessário coletar dados diretamente dos praticantes da academia e indústria. Ao todo foi utilizado de dados de entrevistas, questionários, literatura cinza, *proofs of concept*, pesquisas de campos e artigos científicos.

Para unificar os diferentes resultados produzidos foi utilizado de Métodos Mistos para obter e analisar dados obtidos através de diversas fontes. Em suma os Métodos Mistos são uma combinação de esforços para adaptar técnicas de coleta e análise para produzir um resultado único combinando dados de diferentes origens e tipos (PARANHOS et al., 2016). Este processo é detalhado com mais profundidade no capítulo 3.

### 1.4 Resultados e contribuições alcançadas

Dos resultados obtidos com este trabalho destacam-se:

- **Identificação dos principais fatores de decisão.** Observou-se que, dentre os fatores, os mais importantes são Responsividade, Viabilidade a Longo Prazo e Manutenibilidade. Estes critérios indicam uma constante preocupação com as ferramentas de desenvolvimento multiplataforma, primeiramente quanto a qualidade do aplicativo resultante e depois com a efetividade e longevidade da ferramenta quando comparada ao desenvolvimento nativo.
- **Análise comparativa entre as ferramentas mais populares atualmente disponíveis.** Baseado nos dados obtidos durante as coletas de dados identificou-se que

as ferramentas mais completas são o React Native, Flutter e Xamarin. As três possuem as métricas mais elevadas dentre todas as ferramentas comparadas. As três utilizam de métodos generativos para produzir o aplicativo e são as maiores concorrentes no mercado de ferramentas de desenvolvimento multiplataforma.

- **Dados relevantes e atuais para a comunidade científica e profissional.** Ambos os resultados mencionados proporcionam para os acadêmicos uma forma renovada de abordar o desenvolvimento multiplataforma, e para os profissionais resultados confiáveis para guiar a seleção de sua ferramenta. Também é válido notar a revisão bibliográfica como um resultado importante para a academia, já que unifica os artigos mais citados e sintetiza o estado da arte.
- **Contribuições para trabalhos futuros.** Este trabalho também disponibiliza um formato sólido para reavaliar ferramentas de desenvolvimento multiplataforma conforme surgem novas demandas e ferramentas. Permitindo a novos pesquisadores reutilizarem os critérios descobertos para atualizar a classificação de ferramentas ou aproveitar a metodologia para reavaliar os critérios mais relevantes.

## 1.5 Organização da dissertação

Esta dissertação está organizada da seguinte maneira: No Capítulo 2 é aprofundado o conhecimento sobre as técnicas e tecnologias utilizadas durante o trabalho; o Capítulo 3 descreve como foram empregadas as técnicas de pesquisa utilizadas; o Capítulo 4 aborda o estado da arte e seu impacto neste trabalho; o Capítulo 5 lista os resultados obtidos na pesquisa; e o capítulo 6 combina os resultados em uma discussão sobre o trabalho.

## 2 Fundamentação Teórica

Este capítulo foca em introduzir os conceitos mais importantes que foram utilizados nesta pesquisa. Combinando o estado da arte do desenvolvimento multiplataforma móvel em um guia. A intenção é ilustrar como são classificadas as diferentes abordagens de desenvolvimento na literatura.

Para ser considerado um *software* multiplataforma, este deve ser capaz de funcionar em mais de uma arquitetura, sistema operacional ou dispositivo. Esta tarefa pode demandar muito tempo, já que plataformas diferentes implementam APIs diferentes, fazendo que o projeto crie diferenças para se adaptar às plataformas. Como alternativa comum, desenvolvedores optam por desenvolver seu projeto através de um aplicativo *web*. Esta elimina a necessidade de manter múltiplos projetos, mas ao custo de desempenho e naturalidade em relação à plataforma. Esta forma de desenvolvimento multiplataforma é conhecido como abordagem *web*.

A abordagem *web* não é a única com o objetivo de reduzir tempo de produção em projetos multiplataformas. Não há uma separação bem definida entre os tipos de abordagens, no entanto, há uma tendência para estabelecer cinco grandes grupos (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2012; XANTHOPOULOS; XINOGALOS, 2013; BERNARDES; MIYAKE, 2016): Abordagem *Web*, Abordagem Híbrida (*Hybrid*), Abordagem Interpretada (*Interpreted*), Abordagem Generativa (*Generated*) e Abordagem Dirigidas por modelos (*Model Driven*). As descrições das abordagens deste capítulo, são baseadas nos trabalhos de Xanthopoulos e Xinogalos (2013), Raj e Tolety (2012), Latif et al. (2016).

Para cada abordagem são listadas suas vantagens e desvantagens. Em maior parte, estas características foram obtidas através do trabalho de Bernardes e Miyake (2016). Isto acontece devido à natureza de seu trabalho, que por reunir as definições de dez outros autores traz resultados que representam claramente o consenso do estado da arte.

### 2.1 Abordagem Web

A abordagem *web* é a mais bem delineada por todos os autores que a descrevem. Esta certeza vem do fato de ser uma solução precedente ao reconhecimento da demanda por desenvolvimento multiplataforma. Definida como aplicações baseadas em navegadores, no qual o *software* é recebido pela *Internet*, são os sites e sistemas disponíveis na rede mundial de computadores. Utilizam de tecnologias bem difundidas como *Hyper-Text Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript*, e é muito popular

devido à sua facilidade de aprendizado e aplicação.

O funcionamento da abordagem *web* acontece da seguinte forma: O aplicativo é hospedado em um servidor, no servidor será inserido o projeto do aplicativo, que pode ser desenvolvido de diversas formas diferentes, como HTML, CSS, *JavaScript* e *PHP* ou através de *Web Forms* com *C#* na plataforma *Visual Studio*; O usuário utiliza de um navegador em seu dispositivo para acessar o servidor, ao fornecer o código HTML da página o navegador transforma este código que recebeu do servidor em elementos visuais interativos para o usuário, o ciclo se repete toda vez que uma ação do usuário acessa outra página. Este ciclo de vida é resumido em um modelo observado na Figura 5.

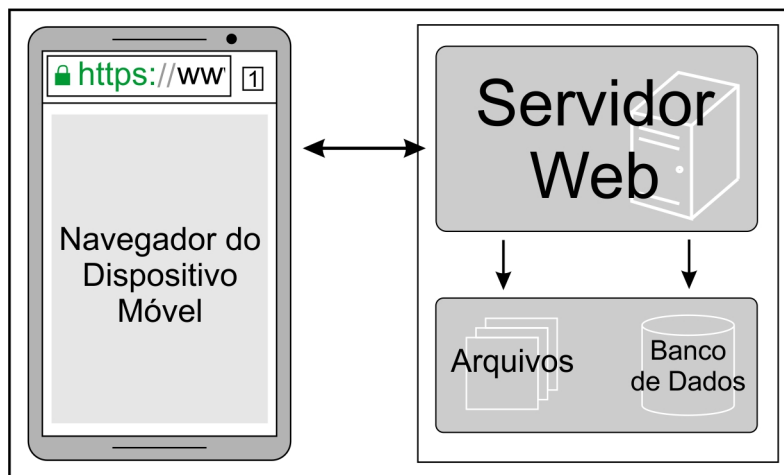


Figura 5 – Modelo da Abordagem *Web* (RAJ; TOLETY, 2012)

O trabalho de Bernardes e Miyake (2016) coloca as seguintes vantagens e desvantagens da abordagem *web*:

- Vantagens:
  - Fácil distribuição e não requer instalações ou atualizações de aplicações;
  - Utilizam de linguagens de programação largamente conhecidas, consequentemente possuem maior leque de ferramentas de desenvolvimento, e novos desenvolvedores terão mais facilidade para se adaptar; e
  - Rodam em qualquer plataforma, desde de que esta possua acesso à *Internet* e execute um navegador.
- Desvantagens:
  - Não são disponibilizadas nas lojas de aplicações de dispositivos móveis, que é a principal fonte de procura de aplicativos;
  - Dependem de conexão com a *Internet*, portanto ficam sujeitas à perda de desempenho por conta de conexões lentas;



- Não utilizam 100% das capacidades gráficas do dispositivo, e acabam perdendo a naturalidade da plataforma;
- Dificuldade de acessar APIs nativas, impossibilitando sistemas de usar de todas as capacidades do dispositivo que possam ser úteis para seu objetivo.

As seguintes inconsistências com o cenário atual e o trabalho de Bernardes e Miyake (2016) (e conseqüentemente com outros trabalhos citados em sua pesquisa) foram encontradas: o desenvolvimento de aplicações *web* se tornou muito mais flexível quando a quinta versão do HTML foi lançada. Nesta, o desenvolvedor tem acesso a um número de APIs nativas que podem ser muito úteis. Além de APIs, no *HTML5*, foram adicionado recursos correspondentes à uma padronização mais moderna, assim como elementos antigos e inapropriados removidos.

O quesito de funcionamento dependente da conexão com a *Internet* também se encontra desatualizado. Técnicas de *caching* permitem que aplicativos *web* reduzissem o consumo de dados e conseqüentemente melhorassem seu desempenho já existem desde o final da década de 90 (WANG, 1999). Com o aperfeiçoamento destas foram criadas técnicas que permitem a operação de um aplicativo *web* de forma independente da conexão, bastando apenas o usuário acessar a página uma primeira vez. Uma destas técnicas é a do *Service Worker*, um *script* executado pelo navegador que contém os dados que os desenvolvedores consideram essenciais para o funcionamento de seu aplicativo. Normalmente são armazenados elementos estáticos e instruções de reação à instabilidade e falta de conexão. Os dados contidos neste *script* são armazenados no *cache* do navegador e exibidos quando não for possível de obter os elementos originais (GAUNT, 2018).

Fazendo uso dos *Service Workers*, outra possibilidade de desenvolvimento *web* foi criada. São os PWAs, que também utilizam do *HTML5* para acessar APIs. Os PWAs utilizam de uma tecnologia do navegador que ao acessar a primeira vez a página do aplicativo, é perguntado ao usuário se ele deseja instalar este aplicativo. Ao instalar, a página pode ser acessada como um aplicativo nativo, sendo em tela cheia e com um ícone disponível na tela inicial de seu dispositivo. E tanto a versão em navegador quanto o aplicativo podem ser acessados sem conexão com a *Internet*. No entanto, quanto a naturalidade da plataforma, será necessário um esforço extra da equipe para adequar aos padrões das plataformas.

## 2.2 Abordagem Híbrida

Aplicativos híbridos tentam combinar as vantagens de aplicativos nativos e *web*. São construídos majoritariamente com *HTML5* e *JavaScript*, não havendo necessidade de conhecer as plataformas alvo. Exemplos de ferramentas que usam desta abordagem são: *PhoneGap*, *IBM Worklight* e *AppMobi*.

A abordagem híbrida é muito próxima à abordagem *web*, onde há um servidor que irá fornecer o acesso ao sistema. O diferencial é a possibilidade de gerar um aplicativo que funciona utilizando o *browser engine* do dispositivo, renderizando o conteúdo HTML em tela cheia. O *browser engine* é uma API criada a partir do navegador que permite embutir a função de navegação em um aplicativo. Também é possível acessar recursos nativos do dispositivo através da camada de abstração. Nesta abordagem é possível criar aplicativos que dependam ou não de conexão com a *Internet*. No caso de uma aplicação *offline* o conteúdo HTML fica alojado no dispositivo e é acessado através do aplicativo.

Na Figura 6 está descrito o modelo de funcionamento desta abordagem. Nesta é observado que o aplicativo faz uso do servidor *web* da mesma forma que a abordagem *web*. Quando não há conexão com a *Internet* disponível, o aplicativo utiliza do código HTML armazenado para disponibilizar funções *offline* ao usuário. No entanto, o desenvolvedor conta com uma camada de abstração responsável por regular o acesso aos recursos nativos. Esta camada de abstração é disponibilizada em código (normalmente *JavaScript*) por bibliotecas fornecidas pela ferramenta escolhida.

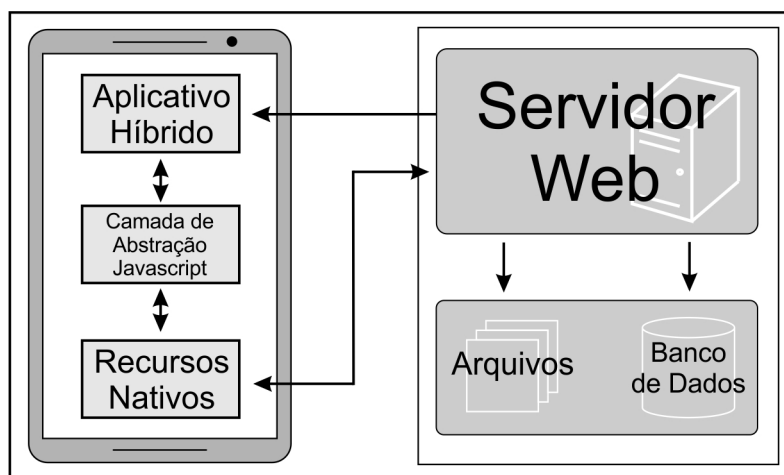


Figura 6 – Modelo da Abordagem Híbrida (RAJ; TOLETY, 2012)

Devido a abordagem híbrida haver pouca evolução desde que o trabalho de Bernardes e Miyake (2016) foi publicado, não há inconsistências com o estado atual. Os autores colocam as seguintes vantagens e desvantagens da abordagem híbrida:

- Vantagens:
  - Permitem acesso às APIs nativas, e à loja de aplicações;
  - Não dependem sempre de *Internet* para o uso; e
  - Utilizam de linguagens de programação largamente conhecidas, consequentemente possuem maior leque de ferramentas de desenvolvimento, e novos desenvolvedores terão mais facilidade para se adaptar.

- Desvantagens:
  - Necessita esforço extra para atingir a natureza visual da plataforma;
  - Apresentam desempenho inferior devido à sua execução ocorrer em um *browser engine*.

## 2.3 Abordagem Interpretada

A abordagem interpretada dá um passo além do que é proposto pela abordagem híbrida. Ao invés de utilizar o *browser engine* é utilizado um interpretador de código, deixando de lado o formato *web* de desenvolvimento e partindo para uma forma mais próxima à nativa. Uma das características mais marcantes da abordagem interpretada é a quantidade de linguagens que podem ser utilizadas para desenvolver. Há um grande número ferramentas que utilizam de diferentes linguagens. As ferramentas mais populares são: *Appcelerator Titanium Mobile*, *LiveCode* e *Rhobile*.

O funcionamento da abordagem interpretada acontece a partir de um interpretador que executa o código da aplicação em tempo de execução. O interpretador é instalado junto ao aplicativo, e este é específico para cada plataforma. Para o desenvolvedor, o interpretador é quem fornece acesso aos recursos nativos através de uma camada de abstração. Normalmente sua capacidade de gerar interfaces de usuário nativas é totalmente dependente do ambiente de desenvolvimento, ficando vinculado ao que os criadores da ferramenta disponibilizam. No entanto, há ferramentas que utilizam de código de interface nativo para cada plataforma. Na Figura 7 está descrito o modelo de funcionamento desta abordagem.

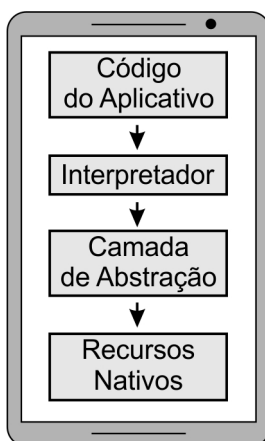


Figura 7 – Modelo da Abordagem Interpretada (RAJ; TOLETY, 2012)

O trabalho de Bernardes e Miyake (2016) coloca as vantagens e desvantagens das abordagens interpretadas como a seguir:

- Vantagens:

- Permitem acesso às APIs nativas, e à loja de aplicações; e
  - Apresentam natureza visual em conformidade com a plataforma.
- Desvantagens:
    - Requerem conhecimento mais específico, tanto da linguagem quanto da plataforma, no desenvolvimento;
    - Perdem um pouco de desempenho em relação aos aplicativos nativos devido ao tempo de interpretação.

É notável a semelhança entre as abordagens híbrida e interpretada, portanto há a necessidade de um aprofundamento neste aspecto. A abordagem híbrida, por utilizar do *browser engine*, necessita que o código se transforme em um *website*, portanto quando ocorre uma ação, o código é executado em *JavaScript* que irá disparar um ação através da camada de abstração, e só então será interpretado pelo dispositivo. Já o interpretador transforma o código diretamente em chamadas internas em tempo de execução, tornando-se uma forma mais ágil do que a híbrida, como pode ser visto no estudo de Dalmasso et al. (2013).

## 2.4 Abordagem Generativa

Aplicativos gerados são caracterizados por serem compilados como um aplicativo nativo. Em seu desenvolvimento um aplicativo é criado para cada plataforma alvo. Há diversas formas de uma ferramenta gerar versões nativas, uma das formas é transformando o código através de um conector, como o *React Native*, ou transpilando-o (transformação de código de uma linguagem em outra linguagem de mesmo nível), como o *Haxe*. Muitas destas ferramentas ainda permitem que o desenvolvedor acesse diretamente o aplicativo gerado, tornando possível editar o código gerado para implementar peculiaridades do sistema e que não são atendidas pelo gerador.

Mesmo com cada ferramenta utilizando de formas diferentes de produzir o código nativo, elas tem o funcionamento geral parecido. O desenvolvedor deve se preocupar apenas com um único projeto, que é transformado em projetos nativos para cada plataforma alvo. Após este processo, algumas plataformas disponibilizam o projeto depois de ser adaptado para a plataforma, podendo ser editado nativamente. Na Figura 8 se encontra o modelo de funcionamento da abordagem descrito neste parágrafo.

O trabalho de Bernardes e Miyake (2016) coloca as vantagens e desvantagens das abordagens generativas como a seguir:

- Vantagens:

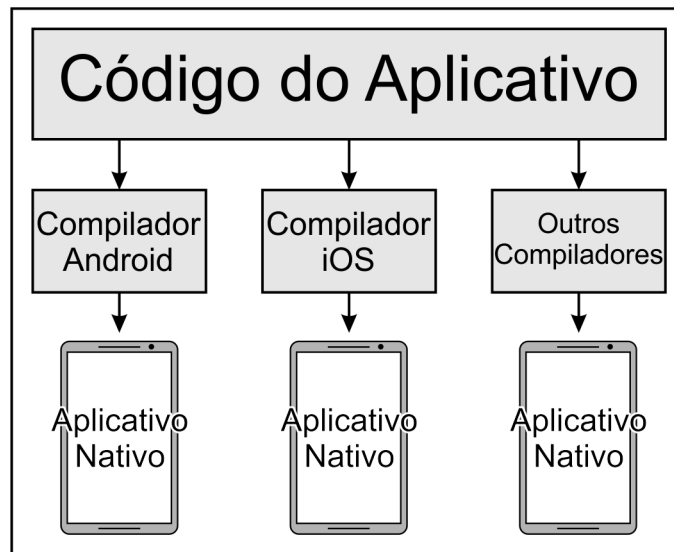


Figura 8 – Modelo da Abordagem Generativa (RAJ; TOLETY, 2012)

- Permitem acesso às APIs nativas, e à loja de aplicações;
  - Apresentam natureza visual em conformidade com a plataforma; e
  - Mesmo desempenho que um aplicativo nativo.
- Desvantagens:
    - Segundo os autores, a interface com o usuário não pode ser reutilizada para outras plataformas.

Novamente, no trabalho de Bernardes e Miyake (2016) há inconsistências com o cenário atual. A desvantagem proposta para esta abordagem já foi superada por ferramentas mais modernas e a modernização de algumas das ferramentas citadas em seu trabalho. O *React Native*, *Xamarin* e *MoSync* são exemplos disto. Nestas ferramentas o desenvolvimento da UI ocorre de forma unificada para todas as plataformas alvo. O *Xamarin* leva isto ainda mais adiante, já que utiliza de código *XAML* que também pode ser reutilizado em aplicativos *desktop*.

## 2.5 Abordagem Dirigida por Modelos

É a abordagem baseada no *Model Driven Development* (MDD), a qual o desenvolvedor pode desenvolver o aplicativo em um alto nível de abstração, normalmente através de modelos *Unified Modeling Language* (UML). Estes modelos podem ser categorizados em três níveis diferentes de abstração:

- ***Computation Independent Model* (CIM)**: Modelo mais abstrato, onde é inserido o problema a ser resolvido;

- **Platform Independent Model (PIM):** Modelo intermediário que é responsável por tratar da solução do problema;
- **Platform Specific Model (PSM):** Modelo de nível menos abstrato que é responsável pela implementação em uma plataforma.

Nestes modelos são aplicados transformadores que geram outros artefatos, que podem ser textos ou outros modelos. A partir destas transformações um código fonte final é produzido. Dada a natureza generativa desta abordagem, Xanthopoulos e Xinogalos (2013) apresentam uma tendência em agrupar a abordagem dirigida por modelos e a abordagem generativa. Contudo, os processos e a forma de construção destas abordagens são significativamente diferentes, principalmente para o ponto de vista dos desenvolvedores. A Figura 9 representa uma visão geral sobre o funcionamento do MDD.

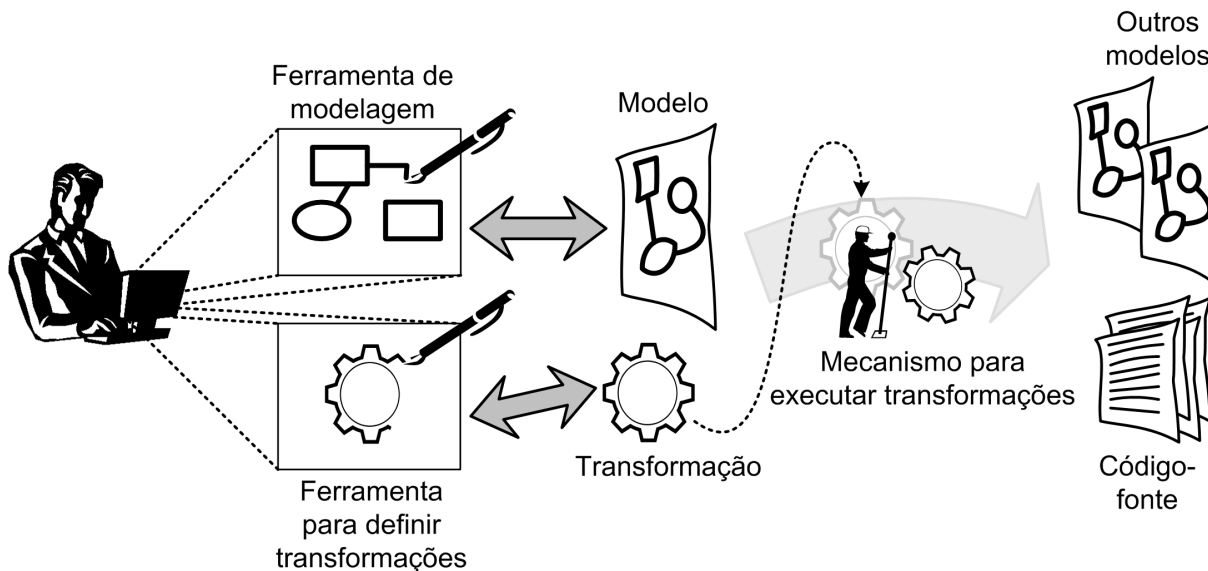


Figura 9 – Principais elementos do MDD (LUCRÉDIO, 2009)

Os trabalhos de Latif et al. (2016) e Gaouar, Benamar e Bendimerad (2015) colocam as vantagens e desvantagens das abordagens dirigidas por modelo como a seguir:

- Vantagens:
  - Criação de aplicações nativas, já que os artefatos resultantes do processo formam projetos nativos.
- Desvantagens:
  - Limita-se ao escopo da linguagem de modelagem utilizada. Apenas aplicativos sob o escopo suportado podem ser modelados; e

- Necessidade de integrar manualmente o código. Isto ocorre devido às ferramentas atuais produzirem o código de forma incompleta. No entanto esta é uma desvantagem que precisa ser melhor estudada, já que é um problema que pode ser resolvido no futuro.





## 3 Metodologia

Este capítulo trata sobre a metodologia geral desta pesquisa. E aqui é descrita a composição de cada processo e artefato utilizados. E também sobre como funcionam as ferramentas de pesquisa utilizadas. A Figura 10 fornece uma visão geral sobre a estrutura, processos e artefatos desta metodologia. Esta figura usa notação baseada em fluxogramas, portanto: o item hexagonal representa o início da pesquisa; os retângulos são processos; os paralelogramos são dados produzidos por processos; o círculo é uma atribuição; e o triângulo um processo de junção dos dados de outros processos.

A filosofia geral da pesquisa foi baseada no conceito de Métodos Mistos por reunir dados tanto quantitativos quanto qualitativos, de diversas fontes e de diferentes naturezas. A compreensão e discussão das ferramentas de pesquisa utilizadas não poderia ser mais vital. Portanto foram empregadas técnicas mistas de análise para permitir a combinação dos resultados em único. Não apenas por questão de assimilar a construção do trabalho, mas também para verificar a confiabilidade dos resultados produzidos por este.

A decisão sobre a metodologia ocorreu de forma reativa após a etapa de revisão inicial descrita ainda neste capítulo. Devido a variedade de tipo de dados que seriam utilizados foi decidido o uso da *Grounded Theory* (GT) justamente por sua flexibilidade. No entanto, durante as coleta de dados, essa necessidade foi reavaliada. A GT é uma opção muito completa para os requisitos, no entanto algumas de suas premissas não eram compatíveis com o objetivo deste trabalho. O maior empecilho é a necessidade de encontrar uma teoria fundamentada nos dados, enquanto este trabalho não necessita elaborar uma resposta para o estado da arte, e sim orientar desenvolvedores e pesquisadores sobre o que é relevante para a área.

### 3.1 Ferramentas de Pesquisa

Nesta seção é apresentado como as ferramentas de pesquisa utilizadas funcionam e como contribuem para o trabalho. Grande parte destas ferramentas são frequentemente utilizadas sob a metodologia GT, a qual inspirou alguns aspectos deste trabalho. A GT é baseada em evoluir uma teoria através dos dados coletados com o objetivo de responder uma questão de pesquisa. Mais informações sobre a seleção da metodologia de pesquisa são discutidas no Capítulo 3.

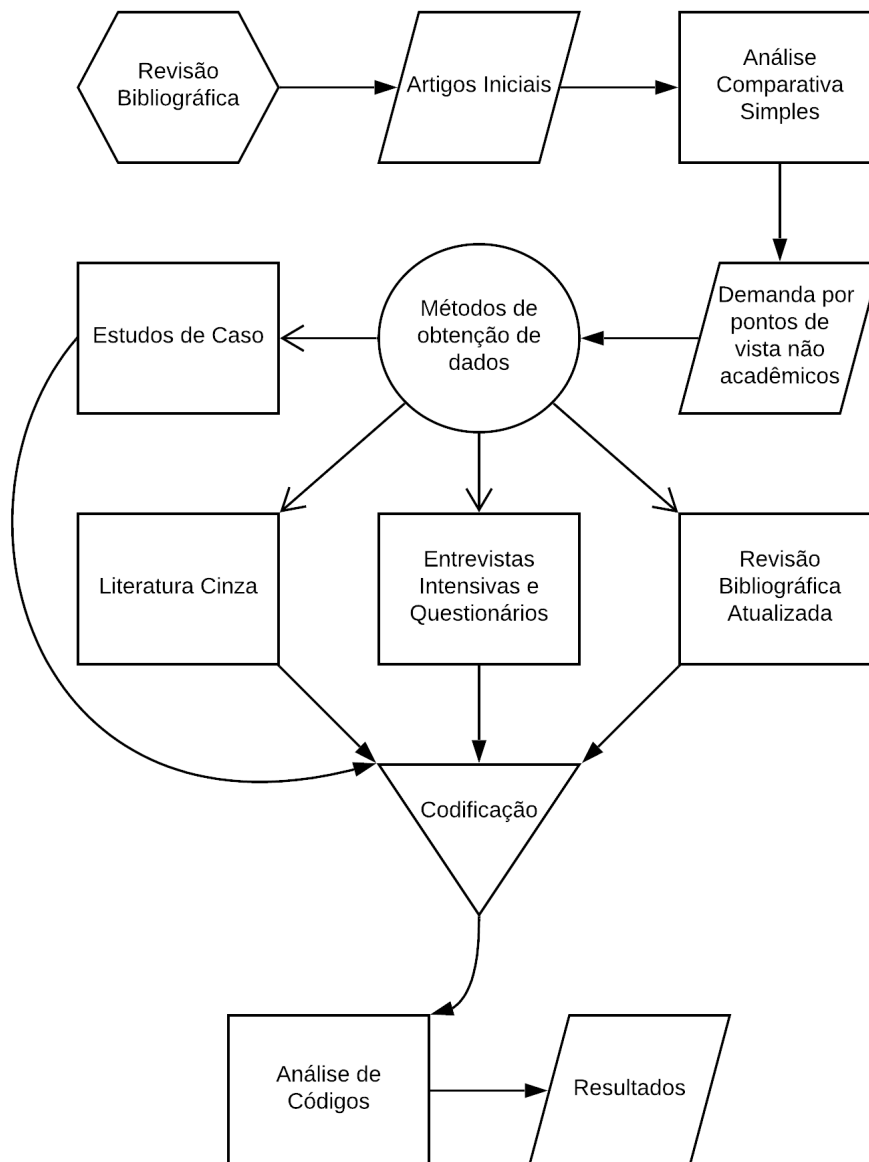


Figura 10 – Modelo geral da metodologia utilizada

### 3.1.1 Entrevista Intensiva

Diferentemente de uma típica entrevista, as entrevistas intensivas tem como intenção permitir o pesquisador direcionar de acordo com as experiências do entrevistado (CHARMAZ, 2014). Esta flexibilidade diminui a quantidade de resultados fora do contexto do entrevistado, mas enriquece a qualidade dos dados dentro de sua experiência, o que no final acaba por produzir resultados com mais confiabilidade e mais atrelados ao estado da prática.

A condução de uma entrevista intensiva envolve mais do que apenas seguir perguntas ou pedir por mais detalhes ao entrevistado. O pesquisador, por possuir mais controle,

pode, dependendo do contexto:

- Manter o entrevistado no assunto;
- Voltar a um ponto anterior;
- Reafirmar o entrevistado para confirmar seu ponto de vista;
- Acelerar ou reduzir o ritmo;
- Mudar de tópico; e
- Validar a ação ou perspectiva do entrevistado.

Estas ações são a essência das entrevistas intensivas. Dessa forma a entrevista pode fluir como uma conversa mais amigável, capaz de atingir uma opinião íntima que uma entrevista direta não atingiria, sendo assim uma ótima técnica para se aplicar quando se busca por resultados qualitativos (CHARMAZ, 2014).

### 3.1.2 Diário de Bordo

Um diário de bordo é uma ferramenta de coleta de dados qualitativos longitudinal auto-reportada pelos participantes de uma pesquisa (FLAHERTY, 2016). É considerada longitudinal por ocorrer durante um período de tempo, seja de alguns dias até alguns meses. É autorreportada por permitir que o próprio participante, no momento que preferir, escreva suas ideias espontaneamente, permitindo assim acompanhar o progresso de uma evolução de forma pouco intrusiva e bastante detalhada.

Um dos usos comuns desta ferramenta é para os próprios autores da pesquisa gerarem dados qualitativos (SOUZA; STEINMACHER; PRIKLADNICKI, 2018) . Isto se torna ainda mais útil quando combinado com uma nova experiência prática, gerando dados sob a visão de um conhecedor da área. Claro que esta abordagem pode gerar certo viés, no entanto existem técnicas de análise que podem reduzi-lo. Uma destas técnicas é a análise utilizando código, que é explicado na próxima Seção.

### 3.1.3 Codificação

Codificação é um processo analítico de dados quantitativos e qualitativos que os categoriza para facilitar sua análise (CHARMAZ, 2014). A base do processo são os códigos, trechos de texto, áudio ou imagens que são demarcados com um nome dado pelo pesquisador, que geralmente é uma breve descrição do fenômeno marcado. A codificação começa com a seleção dos itens que serão analisados. Estes são lidos atentamente pelo pesquisador e para cada trecho que chamar sua atenção o pesquisador o marca com um

código. Códigos podem ser agrupados com a maneira que o pesquisador melhor entender, categorizando-os.

Com as categorias definidas começa o processo de análise. Existem diversas maneiras que a análise desses códigos pode ser realizada. Uma das maneiras quantitativas de análise acontece utilizando softwares de planilhas para gerar um resultado estatístico dos dados. Esta técnica foi empregada nesta pesquisa e se encontra na Seção 5.3. Também pode ser feita uma análise qualitativa utilizando softwares específicos para gerar modelos teóricos, estabelecendo relações entre os temas descobertos. Outra prática adotada foi a realização da codificação por mais de um pesquisador com o intuito de reduzir o viés através do cruzamento de dados.

### 3.1.4 Questionários

Questionário é um instrumento de pesquisa estático e de fácil distribuição. Ele é composto de diversas questões direcionadas que podem ser desenvolvidas para respostas quantitativas e qualitativas. Por não exigir interação ativa do pesquisador o instrumento pode ser amplamente divulgado através da internet, aumentando o alcance da pesquisa.

Caso o questionário use de questões quantitativas, frequentemente é empregada a escala de Likert para sua elaboração. A escala tem como objetivo obter a opinião do colaborador através de respostas em um formato típico que escala de “discordo totalmente” a “concordo totalmente”. Comumente ela possui um número ímpar de respostas, com a resposta no centro sendo “indiferente”, mas também é possível obrigar o colaborador a tomar um lado utilizando um número par de respostas.

## 3.2 Revisão Bibliográfica

A primeira etapa de coleta de dados foi a pesquisa bibliográfica na literatura científica. Esta etapa é descrita neste capítulo. Os alvos do estudo foram bases de dados da ciência da computação e conferências. Os termos de busca empregados foram “cross-platform development”, “multiplatform development” e “multiple platforms systems”. As bases de dados pesquisadas foram:

- *IEEE*;
- *ACM*;
- *Springer*; e
- *Scopus*.

Também há artigos que foram encontrados através de uma revisão dos últimos 4 anos do *MODELS (Model Driven Engineering Languages and Systems)*, que é a principal conferência de engenharia, linguagens e sistemas orientados a modelos, assim como da *GPCE*, importante conferência sobre programação generativa. Estas bases e conferências foram escolhidas por sua importância no contexto acadêmico atual.

Este primeiro levantamento bibliográfico foi utilizado para explorar o estado da arte do desenvolvimento multiplataforma quando esta pesquisa ainda possuía outros objetivos. E imediatamente foi observada uma demanda por um trabalho mais cientificamente orientado e menos preso aos pensamentos dos autores. Como é explicado no Capítulo 4, muitos dos artigos não levam em consideração a opinião dos profissionais da área e baseiam-se em dados fornecidos pelos produtores destas ferramentas. E isto pode ser um problema, já que os produtores são os principais interessados no sucesso da ferramenta, existindo a possibilidade de divulgarem informações nem sempre tão apuradas. As discussões sobre os artigos encontrados nesta etapa localizam-se no Capítulo 4.

A revisão bibliográfica final ocorreu em outubro de 2019 com o objetivo obter novamente o estado da arte. Esta revisão utilizou das mesmas bases e termos de busca que a revisão inicial. Os artigos encontrados assim como uma breve discussão deles se encontram no Capítulo 4.

### 3.3 *Proof of Concept*

Ao mesmo tempo do levantamento inicial, também foi realizado uma primeira *proof of concept*. Este possuía como objetivo buscar uma melhor compreensão do estado da prática e um primeiro contato com as principais ferramentas de desenvolvimento multiplataforma da época. A *proof of concept* consistiu de elaborar um aplicativo capaz de acessar recursos nativos, e avaliar como seria a naturalidade resultante. Para isto, foi desenvolvido um aplicativo que tira uma foto e fornece a localização atual para o usuário.

Após o desenvolvimento dos aplicativos foi feita uma comparação rápida entre alguns aspectos observados pelos autores. Os resultados desta *proof of concept* encontram-se na Seção 5.1.

### 3.4 Entrevista Intensiva

As entrevistas foram realizadas entre dezembro de 2018 e fevereiro de 2019, logo após aprovação pelo Comitê de Ética e Pesquisa da UFSCar. Foram entrevistados um total de 7 pessoas, sendo 6 desenvolvedores profissionais e 1 acadêmico da área. Os cargos dos profissionais variam entre desenvolvedores *front-end*, *full-stack*, engenheiros de *software*

e CEOs de consultoria de desenvolvimento. A escolha destes participantes ocorreu por conveniência, através de listas de emails, contatos próximos e indicações destes.

Como indicado por Charmaz (2014) foi utilizado de entrevistas intensivas como instrumento de coleta de dados. Este tipo de entrevista utiliza de perguntas abertas e flexíveis para explorar a perspectiva do entrevistado. A seguir são listadas as questões abertas iniciais para contextualização do entrevistado e entrevistador:

1. Antes de empregarem o desenvolvimento multiplataforma, como vocês desenvolviam os projetos?
2. Conte-me como começou a utilizar métodos de desenvolvimento multiplataforma.
3. Vocês utilizam apenas uma ferramenta de desenvolvimento multiplataforma?
4. Poderia falar sobre a(s) ferramenta(s) que tem utilizado atualmente ou que já utilizou? Houve alguma transição de ferramentas algum momento?
5. O que contribuiu para utilizá-la(s)?
6. [Se houve,] como foi(é) o processo de decisão para selecionar a ferramenta mais adequada? Quais os principais fatores decisivos? Houve o uso de um guia ou estrutura para tomar esta decisão?

Após esta contextualização se inicia as questões intermediárias, que são as principais ferramentas para obtenção dos dados mais importantes:

1. O que veio depois de começarem a utilizar esta(s) ferramenta(s)?
2. Como foi a adaptação? - Se o participante não falar, perguntar: teve treinamento formal? A documentação é consultada sempre?
3. Seus pensamentos e sentimentos sobre o desenvolvimento multiplataforma mudaram desde que começou a utilizar? Como?
4. Quais foram os principais pontos positivos que percebeu?
5. Qual sua opinião sobre a produtividade utilizando estas ferramentas?
6. Existe algum ponto que foi prejudicado? Poderia me dar exemplos?
7. Quais são as principais reclamações, sua ou de sua equipe, sobre a ferramenta?
8. Qual o suporte que possuem em caso de encontrar problemas?
9. [Caso não utilizem outras ferramentas] o que pensa sobre outras ferramentas? Pensam em algum dia fazer uma transição?

10. Olhando para trás, ocorreu algum evento que se destaca em sua mente? Poderia descrevê-lo(s)? Como este evento afetou o que aconteceu? Como você respondeu às situações resultantes?
11. Quais são as principais lições que aprendeu utilizando o desenvolvimento multiplataforma?
12. Você acredita que em certo momento deverá voltar a desenvolver nativamente? Ou que poderá utilizar do desenvolvimento web?

As questões de finalização têm o propósito de motivar o entrevistado a falar algo que julgue importante mas que não tenha sido perguntado, ao mesmo tempo que permita que o entrevistador volte para algum ponto que possa não ter sido bem explorado:

1. Poderia me dizer como sua visão [de planejamento de projetos] pode ter mudado desde que começou a utilizar estas ferramentas?
2. Com suas experiências, que conselho daria para alguém que acaba de começar a utilizar o desenvolvimento multiplataforma?
3. Há algo que você poderia não ter pensado antes e que tenha lhe ocorrido durante esta entrevista?
4. Há algo que você considere que eu deva saber para compreender melhor sobre [algo que julgar incompleto durante a entrevista]?
5. Há algo que gostaria de me perguntar?

O objetivo destas questões é ser genéricas o suficiente para que o entrevistado demonstre afinidade com algum dos temas. E esta afinidade é utilizada como uma brecha para compreender melhor os pensamentos dele perante o tema. Obviamente as perguntas podem não ser suficientes para achar um ponto em aberto para o entrevistador, no entanto, neste caso cabe ao entrevistador improvisar. De qualquer forma, em nenhuma entrevista houve esta necessidade.

### 3.5 Questionário

O questionário foi aplicado entre novembro de 2019 e janeiro de 2020. A divulgação foi feita através de listas de emails e publicação em fóruns na internet (Comunidades do Ionic, Qt, Flutter, JavaScript, Desenvolvimento Web, PWA, Xamarin e Kotlin no site Reddit). Ele foi divulgado em português quando enviado diretamente para públicos brasileiros, e em inglês quando apresentado em fóruns. A primeira parte do questionário

trata sobre esclarecer as demográficas e o contexto do colaborador. Esta parte conta com as questões:

1. De forma livre e resumida, como intitularia seu trabalho atual?
2. Você atua principalmente na indústria ou academia?
3. Qual o tamanho da empresa/instituição em que trabalha? Micro (até 9 empregados), Pequena (de 10 a 49 empregados), Média (de 50 a 99 empregados) e Grande (mais de 100 empregados)
4. Experiência em meses com cada Ferramenta (*React Native, Xamarin, Phonegap/-Cordova, Flutter, Vue, Ionic, Desenvolvimento Web* e Outras Ferramentas)

Em seguida inicia a primeira seção de questões sobre quais fatores de decisão são mais importantes na escolha de uma ferramenta. As questões são respondidas com uma escala Likert variando de 1 (Nada Importante) a 5 (Fundamental). Estas foram as questões referentes a cada critério de decisão:

1. **Acesso aos recursos nativos** O aplicativo pode ser construído de forma a utilizar de APIs nativas de forma facilmente acessível para o programador
2. **Aplicativo Resultante (Detalhes técnicos do aplicativo gerado)** O sistema desenvolvido pode ser editado nativamente depois de gerado, permitindo refinar detalhes específicos que só seriam possíveis nativamente
3. **Desenvolvimento de Interface de Usuário (Facilidade de implementação)** Ser possível acompanhar o desenvolvimento da interface de usuário sem ter de construir e inicializar o aplicativo, e facilidade de uso da linguagem utilizada para construção da UI
4. **Distribuição (Estar disponível em lojas de aplicativos)** Ser possível lançar o produto na loja de aplicativos, ou distribuí-lo de outras formas
5. **Escalabilidade (O aplicativo poder crescer sem retrabalho)** Capacidade de expansão de forma fluída e intuitiva
6. **Facilidade de Desenvolvimento (A linguagem e ferramentas devem ser intuitivas e eficientes)** A forma de programar e APIs utiliza de conceitos intuitivos
7. **IDE (Possuir um ambiente integrado)** Fornecer ao desenvolvedor ferramentas necessárias para um desenvolvimento pleno
8. **Licença e Custos (O preço para usar a ferramenta)** O *framework* possuir custos, seja para adquiri-lo ou para criar software comercial



9. **Linguagem de Programação** A linguagem de programação ser intuitiva e funcional para o desenvolvedor
10. **Manutenibilidade (Facilidade de compreender e modificar o funcionamento)** Um código com alta legibilidade e possibilidade de organizar o código modularmente de forma a facilitar a manutenção posterior
11. **Naturalidade (O aplicativo parecer natural à plataforma sem trabalho extra)** O aplicativo se assemelhar ao estilo da plataforma, tanto em aparência quanto em funcionamento
12. **Oportunidade para desenvolvimento futuro (Ser possível transferir o projeto para outro *framework* ou plataforma de forma facilitada)** Através de reuso, ser possível transferir o projeto para outro *framework* ou plataforma de forma que caso haja necessidade, ter uma rota de fuga do *framework*
13. **Plataformas Suportadas (Portabilidade para além de *Android* e *iOS*)** Se é importante a ferramenta produzir aplicativos para plataformas além de *iOS* e *Android*
14. **Responsividade (O aplicativo possuir uma navegação fluída)** Ao inicializar e ao interagir com o usuário, o aplicativo responder imediatamente de forma fluída
15. **Tecnologia Difundida (Utilizar de ferramentas ou linguagens populares para desenvolvimento)** Utilizar de uma tecnologia que ocorre frequentemente em outras ferramentas, como o *JavaScript*
16. **Velocidade e Custo de Desenvolvimento (Haver atalhos ou complicações técnicas para o desenvolvimento)** O processo de desenvolvimento apresentar fatores que facilitam ou dificultam o desenvolvimento de alguma maneira
17. **Viabilidade a Longo Termo (A ferramenta continuar evoluindo)** A ferramenta lança atualizações frequentes, possui comunidade ativa e suporte para plataformas mais novas
18. **Comunidade (A possibilidade de tirar dúvidas complexas quanto precisar)** A ferramenta possui um grupo relativamente grande de usuários, que estão aptos a responder perguntas e interagir sobre assuntos relacionados a ferramenta
19. **Documentação (Facilidade de iniciar o aprendizado)** A documentação ser capaz de introduzir um iniciante de forma rápida e eficiente
20. **Desempenho (O aplicativo precisa processar dados rapidamente)** O aplicativo resultante ter a capacidade de utilizar dos recursos internos do dispositivo de forma tão eficiente quanto um nativo

A segunda seção do questionário é opcional para o sujeito. Ela é direcionada a uma ferramenta específica que o colaborador escolher responder sobre. As questões também utilizam uma escala Likert, só que variando de 1 (Discordo Totalmente) a 5 (Concordo Totalmente). As questões são as seguintes:

1. Quanto você recomenda esta ferramenta para outras pessoas interessadas em começarem com desenvolvimento multiplataforma?
2. Minha ferramenta é a melhor escolha disponível atualmente
3. Tenho facilidade em encontrar tudo o que preciso sobre minha ferramenta
4. O resultado final da minha ferramenta cumpre as expectativas do meu cliente
5. Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta
6. Consigo desenvolver interfaces de usuário rapidamente ou do jeito que eu desejar
7. Posso publicar meus aplicativos aonde quiser
8. Tenho facilidade em implementar o que for necessário
9. Consigo instalar minha ferramenta rapidamente
10. Acho justo o que pago (ou deixo de pagar) em minha ferramenta
11. A linguagem em que desenvolvo contribui com minha experiência de desenvolvimento
12. Consigo mudar meu código de forma eficiente
13. Meu aplicativo se parece nativo mesmo sem esforço
14. Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir
15. Minha ferramenta produz para todos os sistemas operacionais que preciso
16. Meus aplicativos são fluídos e reagem rapidamente
17. Utilizo de tecnologias que muitas outras pessoas estão utilizando
18. Consigo implementar rapidamente novas funções, ou criar aplicativos do zero sem esforço exagerado
19. Estou seguro de que a ferramenta continuará existindo e evoluindo por muito tempo
20. Meu aplicativo é tão rápido quanto um aplicativo nativo

Esta seção também conta com as seguintes questões qualitativas: “Já encontrei a(s) seguinte(s) dificuldade(s) com minha ferramenta.”; “Minha ferramenta faz muito bem.”; “O que quiser adicionar sobre sua ferramenta.”; e “Qual a ferramenta que mais utiliza ou que mais possui experiência? (Essa será a ferramenta que estará respondendo as questões a seguir)”. Estas questões podem ser respondidas optativamente de forma aberta.

## 3.6 Literatura Cinza

Literatura cinza é qualquer tipo de dado não científico publicado sem uma seleção apurada. Este tipo de literatura é composto de artigos de blogs, artigos de revistas, vídeos, *podcasts*, documentários, e qualquer outro tipo de forma de compartilhar conhecimento que não tenha sido necessariamente avaliada por acadêmicos.

Esta pesquisa utilizou de literatura cinza, mais especificamente de artigos de blogs e vídeos do *youtube* como parte dos dados a serem codificados. Para obter os artigos de blogs foram usadas as palavras chave “*cross platform development*” e “*react native xamarin flutter ionic*” (vários artigos utilizam nomes de ferramentas em seus títulos) na plataforma de busca online *Duck Duck Go*. Esta plataforma foi escolhida por apresentar resultados sem viés de empresas que comprem anúncios para promover suas páginas. Já na plataforma de vídeos *YouTube* foram usadas as palavras chave “*cross platform mobile development*”, “*mobile development frameworks*”, “*hybrid mobile development*” e “*react native xamarin flutter ionic*”.

Quanto a seleção dos itens encontrados, foram escolhidos apenas artigos e vídeos que tratassem mais do que uma explicação de como cada ferramenta funciona e que não demonstrem um viés para uma ferramenta em específico por motivos de anúncio comercial. Os vídeos também foram filtrados por uma breve avaliação no canal que o publicou: se o canal possui frequência de publicações nos últimos 6 meses, se tem pelo menos 500 inscritos. No final do processo de procura foram encontrados 10 artigos provenientes de literatura cinza e 10 vídeos.

## 3.7 Pesquisa de Campo

A pesquisa de campo compreende um diário de bordo escrito pelo autor durante um período de estágio em uma empresa que utiliza ferramentas de desenvolvimento multiplataforma. Mais especificamente com a abordagem PWA para *web* e *React Native* para produzir aplicativos Android (e futuramente iOS). O diário foi escrito diariamente quando o autor havia contato com estas ferramentas. Nele é escrito as experiências, opiniões, vantagens e desvantagens e qualquer dado que parecer relevante.

Este estudo de campo durou de junho de 2019 a março de 2020, com período

dividido entre desenvolvimento (utilizando ou não das ferramentas), escrita de diários e escrita da dissertação. A empresa utiliza uma plataforma *web* desenvolvida em *React.js* pelo autor, que é responsável principalmente pela exibição de mapas temáticos de agricultura sobrepondo um mapa com imagens de satélite da propriedade do agricultor. A plataforma também possui outras funcionalidades diversas que refletem o caso típico de um *web app*. Já o aplicativo *Android* tem como objetivo cadastrar monitoramento de pragas sem conexão com a internet na propriedade para depois sincronizar com a plataforma.

A plataforma possui 67 usuários ativos, dentre eles agricultores, técnicos agrícolas e agrônomos os quais usam a plataforma com frequência semanal ou diária. Já o aplicativo no momento da escrita desta dissertação ainda está em testes, mas terá ainda mais usuários, já que ele tem o foco apenas de coletar dados que são utilizados na plataforma. Isso implica que os aplicativos resultantes tem de estar num padrão de qualidade mínimo para suportar seu uso.

### 3.8 Análise Final

Para agrupar os dados coletados através de todas estas fontes de dados de diferentes naturezas foi utilizado de Métodos Mistos para obtenção, análise e combinação de dados. Os Métodos Mistos são justamente procedimentos referentes ao uso de dados quantitativos e qualitativos de diferentes fontes e formatos (PARANHOS et al., 2016).

Cada origem de dados foi analisada separadamente. As entrevistas, literatura cinza e revisão bibliográfica foram codificados. O questionário foi analisado quantitativamente com a ajuda de operações estatísticas simples. E as *proofs of concept* foram analisadas narrativamente, apenas como forma de validação entre o que foi observado com as outras análises e o que foi experienciado pelo pesquisador durante o desenvolver do estudo.

Finalmente para combinar todos estes resultados foi utilizado do seguinte desenho de pesquisa: Os dados quantitativos produzem uma fundação sólida sobre o que deve ser considerado relevante como critério; A codificação propõe a estrutura de como estes critérios serão organizados; As *proofs of concept* comprovam a organização através da observação do pesquisador. Já os resultados referentes às ferramentas de desenvolvimento multiplataforma seguiram um desenho parecido com o dos critérios, mas com a codificação apenas agregando aos resultados dos questionários com o mesmo peso.

## 4 Trabalhos Relacionados

Este capítulo revisa e discute trabalhos encontrados durante a revisão bibliográfica que se assemelham de alguma forma a esta pesquisa. Cada trabalho é brevemente apresentado, e depois discutido. Todos os trabalhos foram utilizados na etapa de codificação. Para maior separação de tópicos este capítulo se divide em tópicos, sendo cada título de seção uma alusão ao fator de agrupamento dos artigos discutidos. E na última seção é realizada uma discussão generalizada do que foi discutido.

### 4.1 Metodologia

Os artigos foram encontrados nas bases *IEEE*, *ACM*, *Springer*, e *Scopus* com as seguintes strings de pesquisa: “cross-platform development”, “multiplatform development” e “multiple platforms systems” publicados a partir de 2012. Para separar os artigos a serem avaliados foram utilizados dos seguintes critérios de inclusão:

- Artigos que em seu título deem uma indicação de que utiliza de ferramentas multi-plataforma, seja por conter alguma palavra chave ou por conter algum sinônimo;
- Artigos que em seu resumo utilizem de alguma palavra chave; e
- Esteja nos primeiros 100 resultados por ordem de relevância ou de citações dos sistemas de buscas da base de dados.

Foram coletados 32 artigos do *IEEE*, 25 na *ACM*, 12 no *Springer* e 15 no *Scopus*. Após a remoção de artigos repetidos sobraram 45 artigos. E destes sobraram 37 após aplicar os seguintes critérios de exclusão:

- Falar diretamente sobre desenvolvimento multiplataforma, seja; e
- Não ser apenas uma discussão de uma ferramenta isolada, ou seja, sem compará-las ao estado da arte e/ou outras ferramentas.

### 4.2 Comparação generalizada de ferramentas

A intenção destes artigos segue um modelo recorrente que consiste em selecionar alguns critérios de avaliação e avaliar algumas ferramentas com base nos critérios. De certa forma e em um grande escopo, este modelo é o mesmo que o utilizado nesta pesquisa. No entanto o principal diferencial se encontra na forma que os critérios são levantados,

na forma que são avaliadas as ferramentas e na forma de seleção das ferramentas. Todos os trabalhos nesta categoria ou utilizam de artigos anteriores para criar seus critérios, ou não citam fonte alguma (inferindo que os próprios pesquisadores os levantaram). E quase todos os trabalhos utilizam da própria experiência dos pesquisadores para atribuir um valor para cada critério e cada ferramenta. Seja esta experiência adquirida em um estudo de caso ou por prática profissional.

Os artigos Ahmad et al. (2018), Bernardes e Miyake (2016), Botella, Escribano e Peñalver (2016), Majchrzak, Biorn-hansen e Gronli (2017), Meirelles et al. (2019), Pinto e Coutinho (2018), Vilcek e Jakopec (2017), Ferreira et al. (2018), Vishal e Kushwaha (2019), Caballero, Bodden e Athanasopoulos (2016), Tufail et al. (2018), Dalmaso et al. (2013), Heitkotter, Hanschke e Majchrzak (2012), Latif et al. (2016), Palmieri, Singh e Cicchetti (2012), Pawar, Jagtap e Bhamare (2014), Ribeiro e Da Silva (2012), Xanthopoulos e Xinogalos (2013) se encaixam nesta categoria, mesmo que cada um possua sua peculiaridade. Há algumas exceções que merecem ser discutidas mais profundamente. Os artigos que não se incluem nestas exceções no geral realizaram seu levantamento de dados e comparação baseado apenas em opiniões dos próprios autores.

Bernardes e Miyake (2016), através de uma revisão sistemática, compararam as vantagens e desvantagens de diversas abordagens de desenvolvimento multiplataforma para dispositivos móveis que estão disponíveis no mercado. Também são listadas as vantagens e desvantagens do desenvolvimento multiplataforma de forma geral, sem considerar abordagens específicas. O estudo conta com resultados detalhados e discussões do que foi observado nesta revisão sistemática, uma síntese destes detalhes podem ser encontrados no Capítulo 2.

A partir do *snowballing* dos alvos de sua revisão sistemática, foram encontrados os seguintes autores: Xanthopoulos e Xinogalos (2013), Dalmaso et al. (2013), Heitkotter, Hanschke e Majchrzak (2012), Palmieri, Singh e Cicchetti (2012), Ribeiro e Da Silva (2012) e Raj e Tolety (2012). Melhores observações sobre esses estudos são encontrados ainda nesta subseção.

A descoberta de vantagens e desvantagens por meio dessa revisão sistemática implicou em alguns tópicos que estão inconsistentes com o cenário atual. Tais inconsistências são: “As aplicações dependem de conexão com a internet” e “Não permitem o acesso a APIs nativas” sobre as desvantagens da abordagem *web*, como já discutido na Seção 2.1, estas desvantagens já podem ser resolvidas; e “A UI não pode ser reutilizada, ou seja, deve-se desenvolver código específico de interface para cada plataforma. D.” sobre as desvantagens da abordagem generativa, que também foi discutida na Seção 2.4, mostrando que ferramentas modernas não apresentam mais esta característica.

Dalmaso et al. (2013) introduzem seu trabalho ilustrando fatores de decisão para desenvolver aplicativos multiplataformas, neste são comparadas as abordagens nativa,

*web* e multiplataforma (sem apontar para uma abordagem específica). Por ser um guia atemporal, diferente dos outros que foram encontrados, ele é de suma importância para este trabalho, portanto, foi traduzido para a Tabela 1. Contudo, há um problema com este guia em relação à falta de embasamento e metodologia, se tornando apenas uma opinião dos pesquisadores que realizaram este trabalho.

<b>Crítérios de Decisão</b>	<b>Abordagem Nativa</b>	<b>Abordagem Web</b>	<b>Abordagem Multiplataforma</b>
Naturalidade	Excelente	Muito Bom	Não tão bom quanto aplicativos nativos
Qualidade dos Aplicativos	Alta	Média	Média para baixa
Usuários em potencial	Limitados para plataforma específica	Máxima	Grande
Custo de Desenvolvimento	Alto	Baixo	Médio para baixo
Segurança dos Aplicativos	Excelente	Depende da segurança do navegador	Não é boa
Suportabilidade	Complexa	Simple	Médio para complexo
Facilidade de Atualização	Complexa	Simple	Médio para complexo
Tempo de Produção	Longo	Médio	Curto
Extensão do Aplicativo	Sim	Sim	Sim

Tabela 1 – Fatores de Decisão de Dalmaso et al. (2013).

Para prosseguir o trabalho, realizou-se um levantamento dos requisitos que um *framework* multiplataforma deveria possuir, seguido por um conceito genérico de arquitetura do desenvolvimento multiplataforma. Então, os autores realizam sua avaliação sobre as ferramentas *PhoneGap*, *Sencha Touch 2.0*, *PhoneGap + Sencha Touch 2.0*, *PhoneGap + JQuery Mobile*, *Application Craft* e *Appcelerator Titanium*, listando sua opinião pessoal sobre alguns aspectos que considerou relevante.

Finalmente é realizada uma comparação das ferramentas neste capítulo, e é encontrado um guia apontando a melhor abordagem para os seguintes requisitos: naturalidade, uso online/offline, compatibilidade, acesso limitado de recursos nativos e segurança. O artigo se encerra com comparações de desempenho entre as ferramentas escolhidas.

Heitkotter, Hanschke e Majchrzak (2012) foram um dos pioneiros a realizar estudos

comparativos sobre o desenvolvimento multiplataforma. Eles tiveram como objetivo classificar abordagens, analisar e comparar as soluções baseadas em tecnologia *web*, ou seja, aplicativos *web*, *PhoneGap* e *Titanium Mobile*. É mencionado, no capítulo de trabalhos relacionados, que até aquele momento, a academia só discutia plataformas de dispositivos móveis, mas não havia um estudo que tratasse de formas de desenvolver para mais de uma plataforma.

Os autores listam como abordagens multiplataforma os aplicativos *web*, híbridos e *runtime*, e então é descrita uma visão geral sobre eles. É válido ressaltar sua menção à abordagem dirigida por modelos, abordagem que nem todos os autores consideram. Para dar início às comparações, os autores definem os critérios em duas perspectivas: *Infraestrutura*, com os critérios de licença e custo, plataformas suportadas, acesso a recursos nativos, viabilidade a longo termo, *Look and Feel*, desempenho e distribuição; e *Desenvolvimento*, com os critérios de *Integrated Development Environment* (IDE), design de UI, facilidade de desenvolvimento, manutenibilidade, escalabilidade, oportunidades de aprofundamento do desenvolvimento e velocidade e custo de desenvolvimento.

A comparação é realizada pelos autores atribuindo uma nota, e explicando o motivo desta nota. Cada critério comparado é justificado extensamente para cada ferramenta analisada. Finalmente, é realizada uma discussão breve destas comparações.

Latif et al. (2016) realizam um estudo das abordagens existentes no mercado, com ênfase na abordagem dirigida por modelos. O artigo inicia com uma descrição bem detalhada das abordagens *web*, híbrida, interpretada, generativa (a qual chama de *transpilada*) e dirigida por modelos. Então é feito um levantamento dos requisitos desejáveis para essas ferramentas, discutindo sobre os tópicos de escalabilidade e manutenibilidade, acesso a recursos nativos, consumo de recursos, segurança e IDE. Os autores concluem com sua opinião de que a abordagem dirigida por modelos é a mais expansível, e de que possivelmente será uma tendência global em breve.

O artigo tem como objetivo comparar as abordagens, e identificar os requisitos desejáveis para o desenvolvimento multiplataforma, mas não há nenhuma justificativa ou metodologia que suporte a identificação feita. A conclusão dos autores é forte e impactante, mas também é apresentada sem embasamento além da própria opinião dos autores, portanto não é possível considerá-la como fato.

Palmieri, Singh e Cicchetti (2012) têm como objetivo providenciar uma visão geral de ferramentas multiplataformas, de forma a auxiliar o desenvolvedor a escolher a ferramenta ideal para seu projeto. O trabalho é iniciado tratando dos benefícios do desenvolvimento multiplataforma, seguido pelos critérios de seleção e comparação das ferramentas exploradas. Os autores prosseguem introduzindo didaticamente as ferramentas selecionadas: *Rhodes*, *PhoneGap*, *DragonRad* e *MoSync*.



As comparações contemplam os critérios: plataformas suportadas por cada ferramenta, linguagens de programação, forma de acesso às APIs nativas, IDE, extensibilidade, licença, código aberto, arquitetura e disponibilidade do padrão de design MVC e finalmente o acesso a recursos específicos da plataforma.

Ao se limitar a comparar as ferramentas, mesmo que contemplem formatos ligeiramente diferentes, os autores restringem seu estudo para as ferramentas e não para as abordagens. Também é evidente que as comparações são superficiais, limitando-se às características que podem ser encontradas nos *websites* das ferramentas, sujeitando os dados às tendências dos fabricantes das ferramentas.

Pawar, Jagtap e Bhamare (2014) tem como objetivo iniciar o leitor no desenvolvimento de aplicativos independentes de plataformas, descrevendo diferentes ferramentas e suas tecnologias. Esta descrição é baseada em: linguagem de programação utilizada, licença, acesso aos recursos nativos, IDE e plataformas suportadas. São comparadas as ferramentas: *Apache Cordova/Phonegap*, *Adobe Air*, *Sencha Touch*, *Qt*, *MoSync*, *Rhodes*, *Xamarin*, *Marmalade*, *Corona SDK*, *IBM Worklight*, *Titanium Appcelerator* e *Dragon-Rad*. O trabalho finaliza com uma comparação das ferramentas, com a finalidade de guiar o desenvolvedor com os seguintes critérios: funcionamento, plataformas suportadas, linguagem de desenvolvimento, tipo de saída gerada, IDE, código aberto, recursos nativos e propósitos especiais.

O artigo lida com aspectos superficiais das ferramentas, são informações que podem ser encontradas nos sites dos desenvolvedores. Também não há um estudo prático mostrando ao leitor os pontos de vista não técnicos. Há inconsistências com o cenário atual, pois ferramentas como *Qt*, *Xamarin* e *Sencha Touch* já possuem acesso a recursos nativos, ao contrário do sugerido pelos autores. Avanços na abordagem *web*, como os discutidos na Seção 2.1, também são desconsiderados.

Raj e Tolety (2012) tem como objetivo mostrar as falácias e armadilhas do desenvolvimento multiplataforma, provendo informações para fazer a escolha certa. Os autores iniciam explicando, de maneira muito didática, as diferentes abordagens de desenvolvimento multiplataforma, apontando as vantagens e desafios de cada uma. O trabalho é finalizado com um guia para seleção de abordagem, dependendo do tipo de aplicação que será construída. O guia recomenda qual tipo de abordagem deve ser utilizada ao construir aplicativos do tipo: dirigidos por dados em servidor, *standalone*, baseado em sensores ou entrada e saída de dados (Com processamento de dados no dispositivo), baseado em sensores ou entrada e saída de dados (Com processamento de dados no servidor) e cliente-servidor. Este formato de modelo de decisão funciona para aplicativos simples, mas ao adicionar complexidade ou aplicativos que realizem mais de uma categoria destas este modelo não se adéqua mais.

Há inconsistências com o cenários atual em vários pontos. Por exemplo, os autores

listam desafios da abordagem *web* como a falta de acesso a recursos nativos, que desde 2011 já pode ser parcialmente resolvida com *HTML5*, e a possível perda de desempenho por conta de conexão com a internet, que pode ser mitigada com técnicas de *caching* (WANG, 1999).

O artigo não deixa claro os métodos científicos utilizados para a elaboração do guia. Os autores buscaram algumas informações nos sites dos fabricantes, acrescentaram sua opinião e elaboraram uma tabela que diz qual abordagem é melhor para certos tipos de aplicações.

Ribeiro e Da Silva (2012) tem como objetivo providenciar uma visão geral do estado da área de ferramentas de desenvolvimento multiplataforma. O artigo selecionou as ferramentas *Rhodes*, *PhoneGap*, *DragonRAD*, *Appcelerator Titanium*, *mobl* e *Cannapi mdsl*, para suas comparações. As comparações se limitam a: linguagem de programação, aplicativo resultante, plataformas suportadas, IDE e acesso a recursos nativos. O artigo finaliza com uma discussão exibindo as características mais relevantes para o desenvolvedor selecionar a ferramenta mais apropriada e falando sobre os benefícios e desvantagens destas.

O artigo demonstra com sucesso o estado da arte de sua época, elaborando sobre as ferramentas mais populares, e discutindo sobre seus usos potenciais. Suas análises são baseadas em informações que podem ser encontrados nos sites do fornecedores destas ferramentas, ou seja, não há uma verificação de fatores não técnicos.

Um dos pontos que merecem atenção neste trabalho é a consideração de ferramentas dirigidas por modelos. Muitos dos trabalhos estudados não levam em consideração esta abordagem, que é tão válida quanto as outras. Os autores realizam uma pequena introdução aos conceitos de MDD e *Domain Specific Language* (DSL), e mostra como o *Cannapi* e o *mobl* os utilizam. Os autores concluem que o uso de MDD pode ser uma abordagem adequada para facilitar o desenvolvimento de aplicações produzidas para grandes números de usuários.

Xanthopoulos e Xinogalos (2013) dividem seu trabalho em três partes, iniciando com explorar e definir as quatro abordagens mais importantes, que são *web*, híbrida, interpretada e generativa. Então, são definidos os problemas chaves de cada abordagem, e é feita uma análise comparativa para demonstrar as vantagens e desvantagens. A partir destas informações, é identificada uma abordagem promissora que será investigada na prática. Finalmente, são extraídas as conclusões sobre as abordagens de desenvolvimento multiplataforma.

A análise comparativa toma como critérios: distribuição pela loja de aplicativos, tecnologias bem difundidas, acesso a recursos nativos, naturalidade e desempenho percebido pelo usuário. São comparadas as ferramentas: *Rhodes*, *PhoneGap*, *DragonRad* e

*MoSync*. Concluindo que não há uma abordagem específica para ser escolhida para um aplicativo genérico, os autores acabam por escolher uma abordagem interpretada para criar seu estudo de caso. Em sua conclusão, são acrescentadas algumas opiniões dos autores sobre alguns quesitos que julgaram importante. Estas opiniões são sobre as peculiaridades de cada ferramenta que compararam.

O estudo de caso trata-se de desenvolver um aplicativo de *RSS Feed*, exibindo o código *JavaScript* necessário para sua conclusão. Em contraste com Latif et al. (2016) e Ribeiro e Da Silva (2012), os autores observam que a abordagem mais promissora para o futuro do desenvolvimento multiplataforma é a generativa.

O artigo de Meirelles et al. (2019) faz uma comparação mais generalista entre desenvolvimento nativo e desenvolvimento multiplataforma. No entanto, seu principal diferencial é o fato de ter dados baseados em questionários, mesmo que maior parte dos colaboradores sejam estudantes. Os questionários foram direcionados ao uso de meios nativos de desenvolvimento e de desenvolvimento multiplataforma de uma forma muito ampla, sem considerar os diferentes tipos de ferramentas. O artigo também não demonstra como os critérios de comparação foram levantados. Esse conjunto de fatores preocupantes se juntam para demonstrar um resultado engessado em relação às ferramentas de desenvolvimento multiplataforma, já que a diferença entre as ferramentas não foi considerada.

O artigo de Ferreira et al. (2018) utiliza de estudos de caso para realizar a comparação. Mas possui o diferencial de utilizar do *GitHub* para decidir quais ferramentas serão utilizadas nos estudos de caso. O estudo optou por utilizar Sencha Touch, PhoneGap e Titanium para realizar as comparações. E conclui que todas as ferramentas tiveram capacidade de realizar a maioria das funcionalidades necessárias. A maior lição aprendida é sobre o uso de estatísticas obtidas no *GitHub* para obter a popularidade das ferramentas, que é uma técnica atemporal e de fácil execução para realizar este levantamento.

O artigo de Ahmad et al. (2018) é o trabalho que mais se aproxima com esta pesquisa. Os autores realizam uma revisão sistemática combinada a entrevistas por questionários para obter a opinião de praticantes sobre a importância de critérios e sobre os métodos de desenvolvimento nativo, web e híbrido. Através da combinação de ambos os resultados (que são quantitativos) obtiveram indicadores de que “Fragmentação”, “Testes”, “Naturalidade” e “Compatibilidade” são os critérios mais relevante.

As únicas críticas ao trabalho de Ahmad et al. (2018) são: a forma a qual escolheu os critérios a serem avaliados, os próprios autores classificaram uma coleção de “desafios” e elaboraram o trabalho baseado nestes, o que torna os resultados pouco flexíveis e difíceis de serem comparados; e também a forma que classifica as ferramentas em apenas três abordagens (nativa, web e híbrida), esta categorização elimina qualquer nuance que possa haver entre as ferramentas de desenvolvimento multiplataforma.

No final não há uma conclusão homogênea para todos estes artigos. Todos possuem peculiaridades, pontos fortes e fracos. Uma forma de tentar adquirir um consenso sobre estes dados, seria através de uma revisão sistemática assim como a feita por Ahmad et al. (2018).

### 4.3 Comparação específica de ferramentas

Esta categoria de artigo difere da categoria anterior (Comparação generalizada de ferramentas) em um ponto: ao invés de comparar todos os aspectos possíveis das ferramentas, é comparado apenas um critério. Normalmente este critério é dissecado em subcritérios que o pesquisador considerar relevante. Esta abordagem permite maior aprofundamento no critério explorado, o que garante uma maior qualidade dos dados resultantes.

Esta categoria possui menos exemplos do que a anterior. Os artigos encontrados são: Corbalan et al. (2018), Jia, Ebone e Tan (2018), Koziokas, Tselikas e Tselikis (2017), sendo o mais influente destes o artigo Corbalan et al. (2018) que aprofunda-se até as partes de engenharia elétrica, de computação e a matemática envolvida no critério de consumo de energia. O artigo explica detalhadamente os processos e métricas envolvidas e realiza testes em diferentes ferramentas de todas as abordagens de desenvolvimento multiplataforma exceto a dirigida por modelos. O artigo conclui principalmente que Titanium é a ferramenta que tem menor consumo de energia em todos os tipos de aplicativos testados.

A conclusão que se tira dos artigos estudados neste tópico é de que é uma forma bastante efetiva para se obter resultados válidos sobre ferramentas. No entanto este tipo de análise é mais comum de ser realizado em critérios que possam ser avaliados de forma mais precisa (normalmente através de *debuggers* e funções de *logging*), como no caso de Corbalan et al. (2018) com o consumo de energia e no de Jia, Ebone e Tan (2018) o desempenho.

### 4.4 Estudo sobre ferramenta e estudos de caso

Estudos sobre ferramentas são os artigos que decidem aprofundar apenas em uma ferramenta, independente ou não de critérios de decisão. Estudos de caso são artigos que tratam sobre descrever o processo de desenvolvimento de um aplicativo utilizando de desenvolvimento multiplataforma. Há também artigos que são uma combinação dessas duas práticas. E os artigos que possuem um estudo de caso, mas apresentam características de outra categoria não entram nesta.

Os artigos encontrados que pertencem a esta categoria são Abid e Karim (2017), Fan e Yang (2017), Biorn-Hansen, Majchrzak e Gronli (2017), Malavolta (2016). Os artigos de estudo de caso encontrados foram testes isolados, um aplicativo desenvolvido

artificialmente para o propósito, sem a preocupação de ser lido por usuários constantemente.

Biorn-Hansen, Majchrzak e Gronli (2017) realizam uma contextualização sobre o funcionamento, implementação, limites, potenciais e conceitos dos PWAs. Então os autores comparam-no com as outras abordagens, primeiro realizando uma comparação sob os fatores: *Instalável*, *Capaz de funcionar offline*, *Disponível em App stores*, *Notificações*, *Disponível em múltiplas plataformas*, *Acesso às APIs nativas* e *Sincronização em plano de fundo*. Também é realizada uma comparação de desempenho sob os fatores de tamanho de instalação e tempos de resposta à lançar uma *activity* e do clique de inicialização ao fim da renderização do App.

Como observado em grande parte dos trabalhos, os autores limitam sua comparação entre abordagens utilizando apenas uma ferramenta de cada abordagem tornando os resultados engessados com a ferramenta específica. Por exemplo, os autores afirmam que a abordagem generativa não pode ser testada antes de ser instalada, porém as ferramentas *Weex* e *Haxe* são exemplos de abordagens generativas que possuem tal suporte.

## 4.5 Discussão final

Antes de começar a discussão, é preciso comentar sobre os meta artigos Biorn-Hansen, Gronli e Ghinea (2018), El-Kassas et al. (2017), Nunkesser (2018). Estes artigos tratam sobre o estado da arte, levantando discussões principalmente sobre a taxonomia utilizada atualmente para os diferentes tipos de abordagens de desenvolvimento multiplataforma. Mas como este trabalho não possui a intenção de propor uma nova taxonomia não será necessária uma discussão destes trabalhos.

Mesmo com a grande quantidade de artigos que de alguma forma se assemelham ao escopo deste trabalho, de maneira geral, podemos afirmar que nenhum deles aborda o tema sob mais de uma ótica. Em particular, apenas o caso do Ahmad et al. (2018) parece que a visão dos profissionais foi levantada. Isto justifica que o uso de questionários e entrevistas com desenvolvedores não é apenas necessário, mas também algo pouco explorado na área. Até mesmo os estudos de caso, não há exemplos aonde um pesquisador realizou o desenvolvimento em ambiente industrial.



## 5 Resultados

Neste capítulo será apresentado em ordem cronológica os resultados de todas as formas de coleta de dados que ocorreram durante a pesquisa. Junto a estes dados também é fornecida uma análise do que os resultados podem significar ou das implicações do resultado.

### 5.1 *Proof of Concept*

Esta *proof of concept* foi uma abordagem inicial, utilizando das ferramentas generativas mais populares de acordo com o *Google Trends* no momento da realização da *proof of concept*. Estas ferramentas são: *React Native*, *Xamarin*, *Weex* e *Native Script*. A abordagem generativa foi a escolhida devido a análise de diversos artigos (DALMASSO et al., 2013; LATIF et al., 2016; DALMASSO et al., 2013; PALMIERI; SINGH; CICHETTI, 2012), e principalmente pela conclusão de Xanthopoulos e Xinogalos (2013) sobre o potencial da abordagem generativa.

A *proof of concept* inicial consistiu em elaborar um aplicativo capaz de acessar recursos nativos, e avaliar como seria a naturalidade resultante. Para isto, foi desenvolvido um aplicativo que tira uma foto e fornece a localização atual para o usuário. Estas duas funções foram escolhidas sem uma metodologia específica, e sim por intuição dos pesquisadores.

Com a finalização do desenvolvimento se iniciaram as análises. Começando pela avaliação da interface de usuário e um pouco do aplicativo resultante já é possível detectar diferenças apenas ao visualizar as telas produzidas. Na Figura 11 se encontra a interface do usuário antes e depois de utilizar a função proposta.

Analisando a Figura 11 observa-se que dentro da mesma abordagem, há uma grande diferença de resultado produzido em termos de interface de usuário. As ferramentas *Xamarin*, *NativeScript* e *Weex* ao acessar a câmera, utilizam do aplicativo nativo da câmera, o que faz que quando o usuário clica no botão para tirar a foto, é aberto o aplicativo nativo dentro deste aplicativo, ocasionando mais mudanças de tela e cliques, mas também permitindo configurações mais específicas da câmera. Este tipo de detalhe é relevante ao desenvolvedor por demandar que planeje seu aplicativo com isto em mente, acarretando em mudanças na interface do usuário.

A partir desta *proof of concept* também é possível avaliar a facilidade de desenvolvimento de cada plataforma. A métrica utilizada para comparar este ponto é analisar o código necessário para produzir o aplicativo em cada plataforma. Nas Listagens 5.1 a 5.4

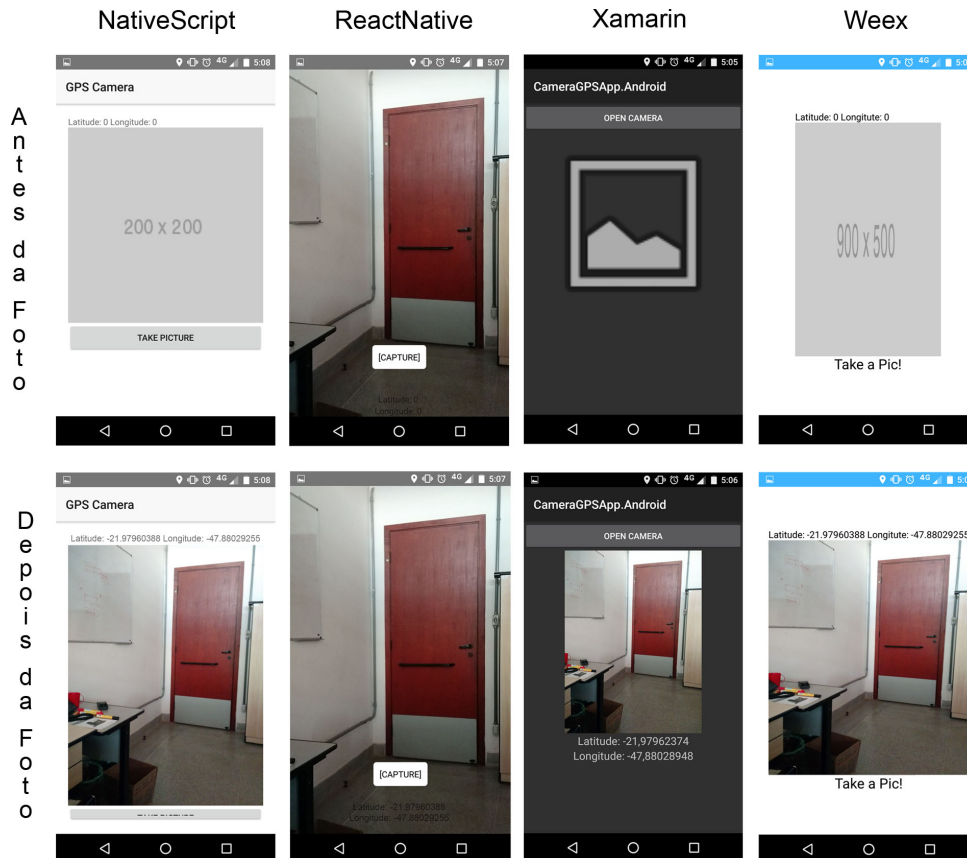


Figura 11 – Comparação de interfaces de usuário dos aplicativos produzidos.

é apresentado o código essencial de cada aplicativo. O código nestas listagens foi simplificado por motivos de leitura, estas linhas omitidas tratavam apenas de configurações de rotina e/ou tratamento de exceções.

```

1 //Funcao chamada ao iniciar o aplicativo
2 function onNavigatingTo(args) {
3 //Solicitar permissao para utilizar a Geolocalizacao
4 geolocation.enableLocationRequest();
5 //Solicitar permissao para utilizar a Camera
6 cameraModule.requestPermissions(); }
7 //Funcao chamada ao clicar no "Take Picture"
8 exports.onTap = function() {
9 //Tirar a foto e inseri-la na view
10 cameraModule.takePicture().then(function(picture) {
11 myImage.src = picture; });
12 //Requisitar a localizacao e exibi-la na view
13 var location = geolocation.getCurrentLocation();
14 coord.text = "Latitude: " + location.latitude
15 + " Longitude: " + location.longitude; }

```

Listagem 5.1 – Código resultante do *NativeScript*, excluindo tratamento de exceções



```

1 //Funcao chamada ao clicar no "[Capture]"
2   takePicture() {
3
4     //Requisitar a localizacao e exibi-la na view
5     navigator.geolocation.getCurrentPosition(
6       (position) => {
7         options.location = position;
8         this.setState({
9           latitude: position.coords.latitude,
10          longitude: position.coords.longitude, });});
11 //Tirar a foto e inseri-la na view
12 this.camera.capture({metadata: options}); }

```

Listagem 5.2 – Código resultante do *React Native*, excluindo tratamento de exceções

```

1 //Funcao chamada ao clicar no "Take a Pic!"
2   take_picture: function (e) {
3     //Tirar a foto e inseri-la na view
4     Nat.camera.captureImage((err, r) => {
5       domModule.addRule('src', {'src': r.path}) })
6     //Requisitar a localizacao e exibi-la na view
7     Nat.geolocation.get((err, r) => {
8       this.lat = r.latitude
9       this.long = r.longitude }) }) }

```

Listagem 5.3 – Código resultante do *Weex*, excluindo tratamento de exceções

```

1 private async void Btn_Clicked(object sender, EventArgs e) {
2   //Confere se ha permissao para utilizar a camera
3   if (CrossMedia.Current.IsCameraAvailable && CrossMedia.
4     Current.IsTakePhotoSupported) {
5     //Definir opcoes de armazenamento
6     var storage = new StoreCameraMediaOptions() {
7       SaveToAlbum = true,
8       Name = "MyPhoto.jpg" };
9     //Tirar a foto e inseri-la na view
10    var pic = await CrossMedia.Current.TakePhotoAsync(
11      storage);
12    Img.Source = ImageSource.FromStream(() => {
13      var stream = pic.GetStream();
14      pic.Dispose();
15      return stream; }); }

```

```

14     var locator = CrossGeocator.Current;
15     var position = await locator.GetPositionAsync(TimeSpan.
FromSeconds(10));
16     //Requisitar a localizacao e exibi-la na view
17     Lbl.Text = "Latitude: " + position.Latitude + " Longitude: "
+ position.Longitude; }

```

Listagem 5.4 – Código resultante do *Xamarin*, excluindo tratamento de exceções

Como pode ser observado nas Listagens 5.1 a 5.4, há muitas divergências entre os códigos, mesmo que possuam o mesmo objetivo. A primeira notável diferença entre os *frameworks* é a possibilidade de manipulação da imagem. Neste quesito cada ferramenta possibilita opções de customização diferentes, como tamanho final da imagem, nome do arquivo e câmera frontal ou traseira. A recomendação é: se ao utilizar de uma dessas ferramentas e uma dessas funções precisarem ser alteradas, é preciso verificar se a ferramenta suporta. Neste caso, apenas o *Weex* não possui opções de customização.

Outro critério notável é a quantidade de linhas de código produzidas, claro que para programadores experientes com a linguagem e *framework*, isto é irrelevante. No entanto, novos programadores contratados para dar manutenção neste aplicativo podem se sentir sobrecarregados com a quantidade de informação presente no código. Neste quesito, o *React Native* e o *Weex* possuem o código mais limpo, enquanto o *Xamarin* necessita escrever mais código para alcançar os mesmos objetivos.

No geral, a *proof of concept* evidenciou as seguintes observações: primeiro e mais importante é que existem diferenças relevantes entre ferramentas da mesma abordagem; e que essas diferenças podem dificultar significativamente um projeto. A *proof of concept* também mostra que o *React Native* demonstra vantagens em relação às outras ferramentas estudadas em ambos os critérios.

## 5.2 Questionário

Conforme discutido no Capítulo 3, foi aplicado um questionário que visa explorar a opinião dos profissionais sobre a importância dos critérios e sobre as vantagens e desvantagens das ferramentas. O questionário utiliza de escala Likert para organizar estas opiniões de forma métrica. Já a divulgação ocorreu principalmente por listas de email e fóruns na internet.

Dos 95 contribuidores: 45 são brasileiros, 31 europeus e 11 norte-americanos; 89,43% da indústria contra 11,57% da área acadêmica; 27,38% trabalham em microempresas (até 9 empregados), 26,19% em pequenas empresas (de 10 até 49 empregados), 8,33% em médias empresas (de 50 até 99 empregados) e 36,9% em grandes empresas (mais

de 100 empregados); 28,42% trabalham ativamente com desenvolvimento mobile, 25,26% são engenheiros de software e 13,68% desenvolvedores “full-stack”.

Identificador	Crítérios
P1	Acesso aos recursos nativos
P2	Aplicativo Resultante (Detalhes técnicos do aplicativo gerado)
P3	Desenvolvimento de UI (Facilidade de implementação)
P4	Distribuição (Estar disponível em lojas de aplicativos)
P5	Escalabilidade (O aplicativo poder crescer sem retrabalho)
P6	Facilidade de Desenvolvimento (A linguagem e ferramentas devem ser intuitivas e eficientes)
P7	IDE (Possuir um ambiente integrado)
P8	Licença e Custos (O preço para usar a ferramenta)
P9	Linguagem de Programação
P10	Manutenibilidade (Facilidade de compreender e modificar o funcionamento)
P11	Naturalidade (O aplicativo parecer natural à plataforma sem trabalho extra)
P12	Oportunidade para desenvolvimento futuro (Ser possível transferir o projeto para outro <i>framework</i> ou plataforma de forma facilitada)
P13	Plataformas Suportadas (Portabilidade para além de Android e iOS)
P14	Responsividade (O aplicativo possuir uma navegação fluída)
P15	Tecnologia Difundida (Utilizar de ferramentas ou linguagens populares para desenvolvimento)
P16	Velocidade e Custo de Desenvolvimento (Haver atalhos ou complicações técnicas para o desenvolvimento)
P17	Viabilidade a Longo Prazo (A ferramenta continuar evoluindo)
P18	Comunidade (A possibilidade de tirar dúvidas complexas quanto precisar)
P19	Documentação (Facilidade de iniciar o aprendizado)
P20	Desempenho (O aplicativo precisa processar dados rapidamente)

Tabela 2 – Identificadores atribuídos aos critérios

Os gráficos a seguir representam a escala de Likert (“nada importante”, “pouco importante”, “neutro”, “quase fundamental” e “fundamental”) por meio de cores. A cor azul equivale ao “fundamental” e vermelho “nada importante”, enquanto as cores intermediárias representam os valores entre estes. A altura das barras de cada cor representa sua correspondente porcentagem, enquanto o eixo horizontal representa os critérios de acordo com a Tabela 2. O objetivo destas questões é definir o quão relevante é o critério para os desenvolvedores quando estão selecionando uma ferramenta.

A Figura 12 resume as respostas do questionário. Nela, é possível observar que, em doze critérios (P1, P3, P4, P5, P6, P9, P10, P14, P17, P18, P19 e P20) a quantidade de respostas que consideram aquele critério como “fundamental” é próxima ou superior a 50%. Quando se considera também a opção “quase fundamental”, esse número sobe para dezoito (apenas P2 e P12 parecem ser menos importantes para os participantes). Isso significa que quase todos os critérios tem alguma importância no momento da escolha de

uma determinada solução multiplataforma. Resta portanto analisar as particularidades e as importâncias desses critérios em cada contexto.

Começando pelo primeiro critério: **acesso aos recursos nativos**. 82% dos participantes concordam com a importância deste critério, portanto é com certeza um aspecto vital para o desenvolvimento multiplataforma. O acesso aos recursos nativos é a base e motivação que iniciou a demanda por estas ferramentas. Afinal, uma ferramenta de desenvolvimento multiplataforma só será válida se possuir acesso aos recursos nativos de todas as plataformas para qual produzirá o aplicativo. No entanto esta informação traz mais do que apenas uma confirmação superficial. Todas as ferramentas possuem acesso a um ou mais recursos nativos, mas nem todas possuem acesso a todos os recursos nativos. Portanto é vital que antes de iniciar um projeto verifique-se de que a ferramenta possui acesso facilitado a todos os recursos planejados.

O segundo critério trata sobre o **aplicativo resultante**. Isto é o sistema desenvolvido poder ser editado nativamente depois de gerado, permitindo refinar detalhes específicos que só seriam possíveis nativamente. Este critério é bastante controverso, já que 30% declararam que não é importante, 28% que é pouco importante e 42% de que é importante. Esta divisão pode ser um reflexo das experiências dos participantes, já que desenvolvedores que usam ferramentas com esta funcionalidade podem achá-la indispensável, enquanto os outros possuem outras formas de resolver estes problemas. De qualquer forma, esta divergência de opiniões significa que é um critério não crucial para escolher uma ferramenta, mas ainda deve ser considerado ao iniciar um projeto mais complexo.

O critério de **desenvolvimento de UI** é bastante subjetivo em sua definição, no entanto pode ser entendido como ferramentas que diminuirão o tempo de desenvolvimento de uma UI. 87% dos participantes concordam com a importância deste critério, portanto o critério pode ser considerado muito desejado ao escolher a ferramenta mais adequada. O que acontece com frequência entre as ferramentas é que todas são capazes de providenciar uma UI mínima para o aplicativo, no entanto, com o aumento da importância da *User Experience* (UX) as interfaces tem se tornado cada vez mais vitais para os aplicativos. E com um maior peso (e mais tempo de dedicação no desenvolvimento) mais importante se torna possuir uma ferramenta que vai suavizar esta tarefa.

A **distribuição** dos aplicativos é a capacidade de publicar o projeto em lojas de aplicativos, ou de facilitar a instalação e/ou localização do mesmo de outra forma. 81% dos participantes concordam com a importância deste critério. Ainda é uma preocupação dos desenvolvedores, mesmo que atualmente todas as ferramentas (que não sejam da abordagem web) conhecidas geram um produto que pode ser publicado em lojas de aplicativo.

O potencial de **escalabilidade** de uma ferramenta é sua aptidão para expandir o escopo de um projeto com a menor quantidade de retrabalho possível. Para este critério

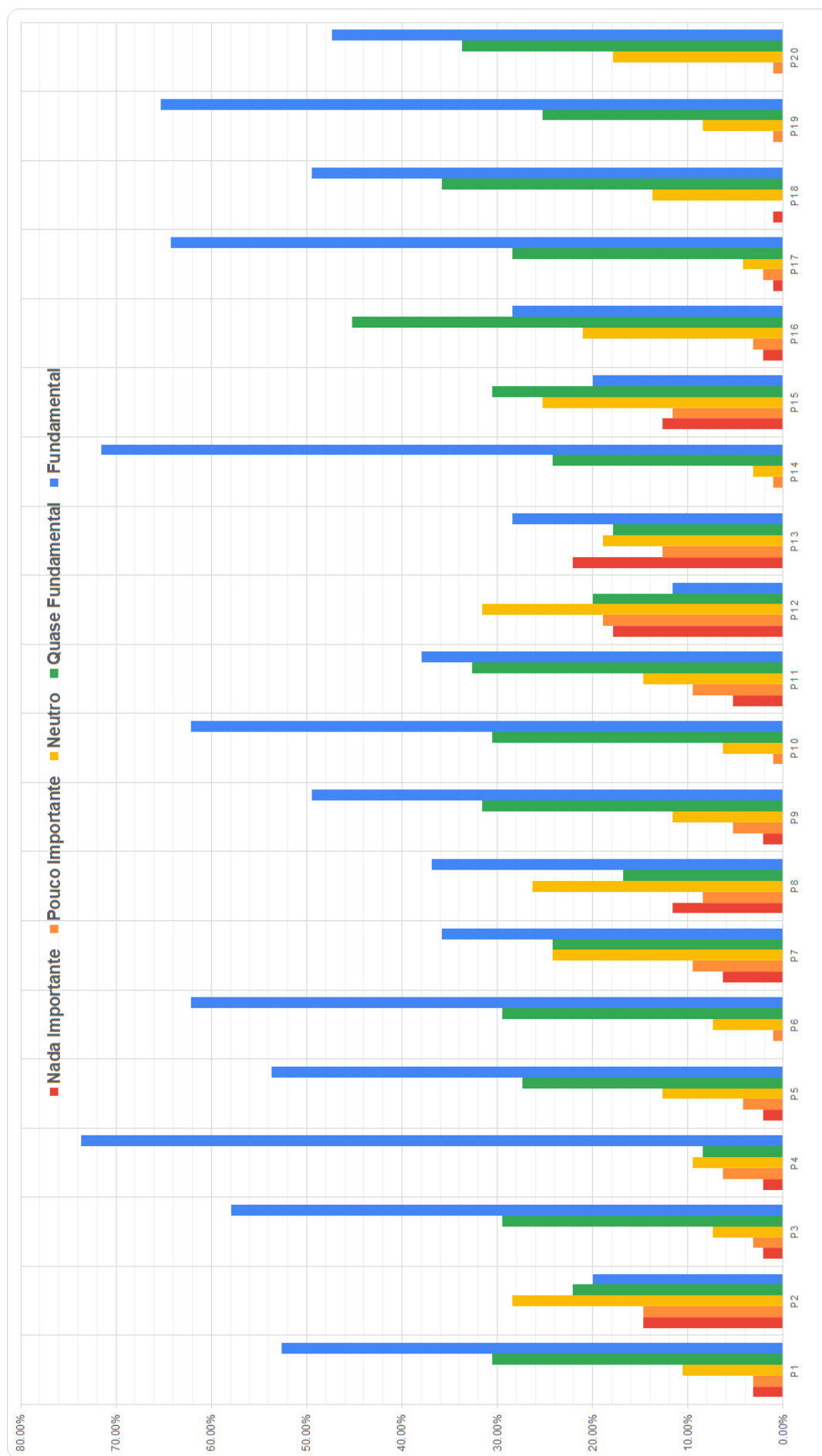


Figura 12 – Distribuição de respostas sobre quais os critérios mais relevantes para o desenvolvimento multiplataforma. (n=95)

81% dos participantes declararam ser fundamental ou “quase fundamental”. Esta métrica denota que escalabilidade continua sendo uma preocupação para os desenvolvedores.

Não surpreendentemente a **facilidade de desenvolvimento** teve 92% dos participantes concordando com a importância deste critério. Este critério é sempre uma prioridade para os desenvolvedores, já que não apenas lhe garantirá uma melhor qualidade de desenvolvimento, também irá acelerar o processo. Porém, a forma de definir a facilidade de desenvolvimento também é um desafio importante. Portanto, essa questão foi avaliada de maneiras complementares, como discute-se na Seção 5.3.

As **IDEs** são a principal forma de contato dos desenvolvedores com o projeto que estão desenvolvendo. Além de prover *syntax highlighting* (Colorização de sintaxe) também inclui ferramentas de teste e validação, integração do aplicativo gerado com *smartphones* conectados e outras funcionalidades que garantem uma experiência melhor ao desenvolvedor. Claro que nem todas ferramentas possuem uma IDE dedicada, no entanto elas fornecem muitas ferramentas que uma IDE tradicional forneceria através de bibliotecas Javascript e extensões de editores de texto. Quanto a opinião dos participantes 60% concordam sobre a importância deste critério, enquanto 24% consideram pouco importante e 16% classificam como não importante. Um fato importante é que desses 16% nenhum participante apresentava mais que 6 meses de experiência com Xamarin, a única ferramenta questionada que possui uma IDE. Isto pode indicar que talvez o interesse nas capacidades das IDEs esteja diminuindo, graças à popularização de ferramentas que apenas recomendam editores de texto.

Quanto aos **custos ou licenças** 53% dos participantes afirmaram que possuir um custo por utilização ou publicação pode inibir o uso de uma ferramenta. No entanto, dos 20% que afirmam não inibir, 63% deles trabalham em micro ou pequena empresas (até 49 funcionários). Isto pode ser traduzido como “Pequenas empresas são a que mais possuem potencial para optar por ferramentas pagas”. O que também pode implicar que ferramentas pagas são as que trazem mais benefícios para pequenas equipes.

O critério de **linguagem de programação** sempre foi um tópico muito discutido dentro da área acadêmica. Afinal a linguagem não ser um fardo para a equipe é sempre uma ótima notícia. 81% dos participantes concordam com a importância deste critério comprovando isso, a linguagem é uma das preocupações dos desenvolvedores, seja por possuir conhecimento prévio ou por apresentar mais funcionalidades. Curiosamente 75% dos desenvolvedores que aprovam o critério utilizam ferramentas que tem Javascript como principal linguagem. Esta informação pode tanto significar que esta linguagem por ser popular tem maior procura por transferência de conhecimento; que os que utilizam ela por tanto tempo dão valor a este critério pela sua linguagem não fornecer o necessário; ou apenas ser uma confirmação da popularidade da linguagem refletida estatisticamente.

**Manutenibilidade** sempre foi uma preocupação da comunidade, e os dados refle-

tem isso diretamente já que 92,7% dos participantes concordam com a importância deste critério e apenas 1% consideram como “pouco importante”. A capacidade de modularização é um ponto extremamente positivo para a saúde de uma ferramenta, não apenas fornecendo mais legibilidade para o código, mas também facilitando o trabalho em equipe com melhor capacidade de programar paralelamente com outros desenvolvedores.

A **naturalidade** de um aplicativo foi descrito pela primeira vez por Heitkotter, Hanschke e Majchrzak (2012). É basicamente a capacidade do aplicativo resultante se parecer e comportar com um aplicativo nativo ao invés de se assemelhar mais a um *website*. 70% dos participantes concordam com a importância deste critério, comprovando que ainda é um critério relevante atualmente. No entanto em uma época em que a UX tem sido cada vez mais valorizada é curioso este valor não ser maior. Isto pode ser explicado com o fato de que cada vez mais *websites* serem muito próximos visualmente de aplicativos, o que potencialmente pode tornar este critério menos relevante no futuro.

O critério **oportunidade de desenvolvimento futuro** é um tema polêmico e que causa certa divisão entre os desenvolvedores. Este critério trata sobre a possibilidade de migração de um projeto para outra ferramenta. Normalmente isto é possível quando ambas as ferramentas compartilham da mesma linguagem. É claro que não é um processo simples e rápido, com uma taxa de reuso baixa, mas ainda assim pode ser uma opção quando a ferramenta que escolheu inicialmente não cumpre os objetivos esperados. É um assunto difícil de lidar e que cada caso deve ser analisado individualmente, por isso a polêmica. E esta divisão é evidenciada nos dados, com um índice médio de 2,88 e uma apenas variação entre 12% e 30% nos 5 possíveis valores, totalizando uma rejeição (respostas “nada importante” e “pouco importante” somadas) de 37%. Isto significa claramente que é um critério bastante controverso e que não parece ser um fator claramente decisivo para os participantes.

Outro critério polêmico é sobre as **plataformas suportadas**, um critério observado em vários artigos científicos e provenientes de literatura cinza da área. E pode ser considerado polêmico pois mesmo que seja muito popular, o desenvolvedor raramente se preocupa com a necessidade de o aplicativo funcionar além do *Android* e *iOS*. Isto é suportado pelos 33% de rejeição (respostas “nada importante” e “pouco importante” somadas), que só não é maior do que o critério anterior. Claro que ainda possui uma taxa de aceitação maior, com 45%, mas ainda assim em comparação aos outros critérios este está muito atrás em relevância.

Já a **responsividade** é a capacidade do aplicativo responder as ações do usuário. Diferente do desempenho, este critério trata mais especificamente sobre a fluidez de UIs. Este critério sempre foi bastante debatido principalmente por conta da falta de responsividade que ferramentas antigas de desenvolvimento multiplataforma apresentavam (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2012; PALMIERI; SINGH; CICCETTI, 2012;

DALMASSO et al., 2013). Este foi o critério mais votado como essencial, já que 95,8% dos participantes concordam com a importância deste critério. E provavelmente é um reflexo desta preocupação, já que muitas destas ferramentas continuam sendo utilizadas até hoje. Além do fato de em conjunto com a naturalidade ser a primeira impressão que o usuário experiencia, dessa forma, uma má impressão será mais marcante do que para outros critérios.

O próximo critério é sobre a ferramenta utilizar de **tecnologias difundidas** para alguma coisa no ciclo de desenvolvimento. Normalmente pode ser uma linguagem, arquitetura, conector de banco de dados ou API e até uma IDE. Possuir uma tecnologia difundida acelera o processo de aprendizado aumentando o interesse de migração e também aproveita uma comunidade em comum, fortificando sua própria comunidade. Mesmo que apenas 50% dos participantes concordem com a importância deste critério, ainda é um tópico um pouco controverso, provavelmente por que dependendo do benefício proporcionado, o desenvolvedor pode ignorar o uso de uma tecnologia difundida.

O critério de **velocidade e custo de desenvolvimento** é um dos mais subjetivos dos propostos. Isto vem do fato deste critério poder ser subdividido em diversos critérios que o influenciam. No entanto com 73% dos participantes concordam com a importância deste critério, ainda um número alto, pode ser que este seja um tema abrangente de mais para ser considerado sem subdivisões. E ainda resta saber como seria feita esta divisão, afinal, seguir apenas a intuição do pesquisador levanta a questão de enviesamento.

A **viabilidade a longo prazo** pode ser entendida como garantias e meios de que uma ferramenta permanecerá relevante com os avanços da tecnologia. Isto ocorre através de fatores como envolvimento da comunidade, ser *open-source*, suporte financeiro da ferramenta, grande número de usuários ou qualquer outro indicador que seja relevante ao caso. Um dos fatores que podem contribuir para sua viabilidade é o suporte financeiro recebido, exemplos deste caso são: React (Recebe apoio do Facebook), Flutter (Recebe apoio do Google) e Xamarin (Recebe verba da Microsoft). Como esperado é um critério bastante importante, 82% dos participantes concordam com a importância deste critério. Afinal quando um projeto começa a existir sem o suporte da ferramenta surge a necessidade da migração, que é um processo muito custoso, especialmente em projetos de grande porte.

A **comunidade** da ferramenta é um aspecto importante que influencia outros critérios já discutidos, mas que também é relevante independente de sua influência. A principal vantagem de uma comunidade vem quando a documentação começa a ser insuficiente, quando um problema não documentado é encontrado pelo desenvolvedor, a comunidade da ferramenta é aonde o desenvolvedor encontrará soluções. 85% dos participantes concordam com a importância deste critério, os desenvolvedores concordam com a importância deste critério. Uma ferramenta sem comunidade é uma ferramenta que está



defasando.

Aproveitando o tema, a **documentação** é sempre o primeiro contato prático que os desenvolvedores tem com a ferramenta. Apenas este fato já seria suficiente para ser levada em consideração, mas a documentação também introduz os desenvolvedores às boas práticas, customização e comunidade. E 90,5% dos participantes concordam com a importância deste critério.

Finalizando com o critério de **desempenho**, um fator sempre discutido na literatura. 80% dos participantes concordam com a importância deste critério. O desempenho pode ser compreendida de diversas formas, seja com a velocidade que o aplicativo se inicializa, ou na agilidade de processamento de dados e até em fluidez de navegação. Mas é importante considerar todos estes aspectos quando tratar sobre o desempenho de um aplicativo.

Os dados de cada critério foram ordenados de forma decrescente na Tabela 3 usando seus índices de concordância e discordância. Estes índices são a soma dos resultados não neutros de cada lado da escala Likert, portanto a soma de “fundamental” e “quase fundamental” constituem a concordância e “nada importante” e “pouco importante” a discordância. A tabela está ordenada pela concordância em ordem decrescente. A partir da análise desta tabela é possível observar que os critérios P14, P10, P17, P6 e P19 são os cinco que possuem maior interesse dos participantes, enquanto P12, P2, P13, P15 e P8 são os cinco mais controversos.

<b>Identificador</b>	<b>Critério</b>	<b>Concordância (%)</b>	<b>Discordância (%)</b>
<b>P14</b>	Responsividade (O aplicativo possuir uma navegação fluída)	95.79	1.05
<b>P10</b>	Manutenibilidade (Facilidade de compreender e modificar o funcionamento)	92.63	1.05
<b>P17</b>	Viabilidade a Longo Prazo (A ferramenta continuar evoluindo)	92.63	3.16
<b>P6</b>	Facilidade de Desenvolvimento (A linguagem e ferramentas devem ser intuitivas e eficientes)	91.58	1.05
<b>P19</b>	Documentação (Facilidade de iniciar o aprendizado)	90.53	1.05
<b>P3</b>	Desenvolvimento de UI (Facilidade de implementação)	87.37	5.26

<b>P18</b>	Comunidade (A possibilidade de tirar dúvidas complexas quanto precisar)	85.26	1.05
<b>P1</b>	Acesso aos recursos nativos	83.16	6.32
<b>P4</b>	Distribuição (Estar disponível em lojas de aplicativos)	82.11	8.42
<b>P5</b>	Escalabilidade (O aplicativo poder crescer sem retrabalho)	81.05	6.32
<b>P9</b>	Linguagem de Programação	81.05	7.37
<b>P20</b>	Desempenho (O aplicativo precisa processar dados rapidamente)	81.05	1.05
<b>P16</b>	Velocidade e Custo de Desenvolvimento (Haver atalhos ou complicações técnicas para o desenvolvimento)	73.68	5.26
<b>P11</b>	Naturalidade (O aplicativo parecer natural à plataforma sem trabalho extra)	70.53	14.74
<b>P7</b>	IDE (Possuir um ambiente integrado)	60.00	15.79
<b>P8</b>	Licença e Custos (O preço para usar a ferramenta)	53.68	20.00
<b>P15</b>	Tecnologia Difundida (Utilizar de ferramentas ou linguagens populares para desenvolvimento)	50.53	24.21
<b>P13</b>	Plataformas Suportadas (Portabilidade para além de Android e iOS)	46.32	34.74
<b>P2</b>	Aplicativo Resultante (Detalhes técnicos do aplicativo gerado)	42.11	29.47
<b>P12</b>	Oportunidade para desenvolvimento futuro (Ser possível transferir o projeto para outro <i>framework</i> ou plataforma de forma facilitada)	31.58	36.84

Tabela 3 – Critérios ordenados por índice de concordância

Após as questões de decisão sobre os critérios no geral, a segunda parte do questionário trata sobre a opinião dos desenvolvedores sobre as ferramentas que utilizam. As

questões foram elaboradas considerando os critérios propostos na primeira parte, com algumas combinações entre eles por motivos de otimizar o tempo de realização. As questões são respondidas de forma parecida, utilizando ainda da escala Likert, mas com os parâmetros variando de “concordo totalmente” a “discordo totalmente”. Também foram feitas três perguntas qualitativas, que podem ser respondidas livremente pelos participantes. Na Tabela 4 são enumeradas as questões, e a seguir se iniciam as subseções para expor os resultados relativos a cada ferramenta.

Uma métrica usada algumas vezes durante a discussão dos resultados das ferramentas é a média das avaliações através da escala Likert. Ela é obtida atribuindo a cada possível resposta um número, no caso 1 para “nada importante” e 5 para “fundamental” e um valor crescente para as respostas intermediárias. A partir disto é possível somar o valor total de todas as respostas e dividi-los pela quantidade de participantes, obtendo a média. Este tipo de métrica não é ideal para muitos cenários, já que não é sensível a resultados muito divididos (poucas respostas neutras, e divisão de respostas positivas e negativas, resultando numa média neutra). Nesta seção a média utilizada é obtida a partir das médias de todas as perguntas, produzindo um valor mais genérico para a avaliação da ferramenta.

### 5.2.1 Flutter

Flutter é a nova ferramenta do *Google* para desenvolvimento multiplataforma para *web*, *desktop* e *mobile*. Apesar de relativamente nova no cenário, já possui uma grande comunidade e ocupa um espaço importante, sendo uma das poucas ferramentas a produzir aplicativo para plataformas além da *mobile*. Das ferramentas que obtiveram mais do que 5 avaliações, o Flutter foi a que possuiu a maior média de todas, com 4,4 (de 1 a 5). Na Figura 13 encontram-se as avaliações realizadas pelos 12 participantes que optaram por responder sobre a ferramenta.

Por obter uma média alta, é de se esperar que seus critérios estejam todos bem avaliados. E é exatamente este o observado. As únicas exceções que se encontram com menos de 75% de concordância (respostas “fundamental” e “quase fundamental” somadas) são o P2, P5, P11, P14 e P17. P2 (Minha ferramenta é a melhor escolha disponível atualmente) estar em déficit pode apenas significar que o Flutter não está completamente refinado e pronto para lidar com todos os cenários reais, por ser uma ferramenta nova. O déficit do P5 (Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta) é algo que foi observado em quase todas as ferramentas, por mais bem construídas, ainda podem surgir algumas falhas que serão detectadas por desenvolvedores, de qualquer forma a ferramenta está bem no limiar do 75%. Assim como o déficit do P14 (Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir) que também pode ser observado em outras ferramentas.

Identificador	Critérios
P1	Quanto você recomenda esta ferramenta para outras pessoas interessadas em começarem com desenvolvimento multiplataforma?
P2	Minha ferramenta é a melhor escolha disponível atualmente
P3	Tenho facilidade em encontrar tudo o que preciso sobre minha ferramenta
P4	O resultado final da minha ferramenta cumpre as expectativas do meu cliente
P5	Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta
P6	Consigo desenvolver interfaces de usuário rapidamente ou do jeito que eu desejar
P7	Posso publicar meus aplicativos aonde quiser
P8	Tenho facilidade em implementar o que for necessário
P9	Consigo instalar minha ferramenta rapidamente
P10	Acho justo o que pago (ou deixo de pagar) em minha ferramenta
P11	A linguagem em que desenvolvo contribui com minha experiência de desenvolvimento
P12	Consigo mudar meu código de forma eficiente
P13	Meu aplicativo se parece nativo mesmo sem esforço
P14	Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir
P15	Minha ferramenta produz para todos os sistemas operacionais que preciso
P16	Meus aplicativos são fluídos e reagem rapidamente
P17	Utilizo de tecnologias que muitas outras pessoas estão utilizando
P18	Consigo implementar rapidamente novas funções, ou criar aplicativos do zero sem esforço exagerado
P19	Estou seguro de que a ferramenta continuará existindo e evoluindo por muito tempo
P20	Meu aplicativo é tão rápido quanto um aplicativo nativo
PQ1	Já encontrei a(s) seguinte(s) dificuldade(s) com minha ferramenta:
PQ2	Minha ferramenta faz muito bem:
PQ3	O que quiser adicionar sobre sua ferramenta:

Tabela 4 – Identificadores atribuídos às questões pertinentes as ferramentas

Este déficit provavelmente ocorre devido as ferramentas não serem projetadas com esta possibilidade em mente.

No quesito do P11 (A linguagem em que desenvolvo contribui com minha experiência de desenvolvimento) encontra-se uma informação muito interessante para esta ferramenta. Flutter é uma das poucas ferramentas que não utilizam diretamente de JavaScript, e ao contrário do esperado sua linguagem (Dart) não contribuiu para uma concordância (respostas “fundamental” e “quase fundamental” somadas) unânime. Isto pode vir do fato de não ser uma linguagem popular, dificultando sua curva de aprendizado e dificultando a experiência dos desenvolvedores novatos. Isto se reflete no P17 (Utilizo de tecnologias que muitas outras pessoas estão utilizando), já que Flutter não utiliza de muitas tecnologias

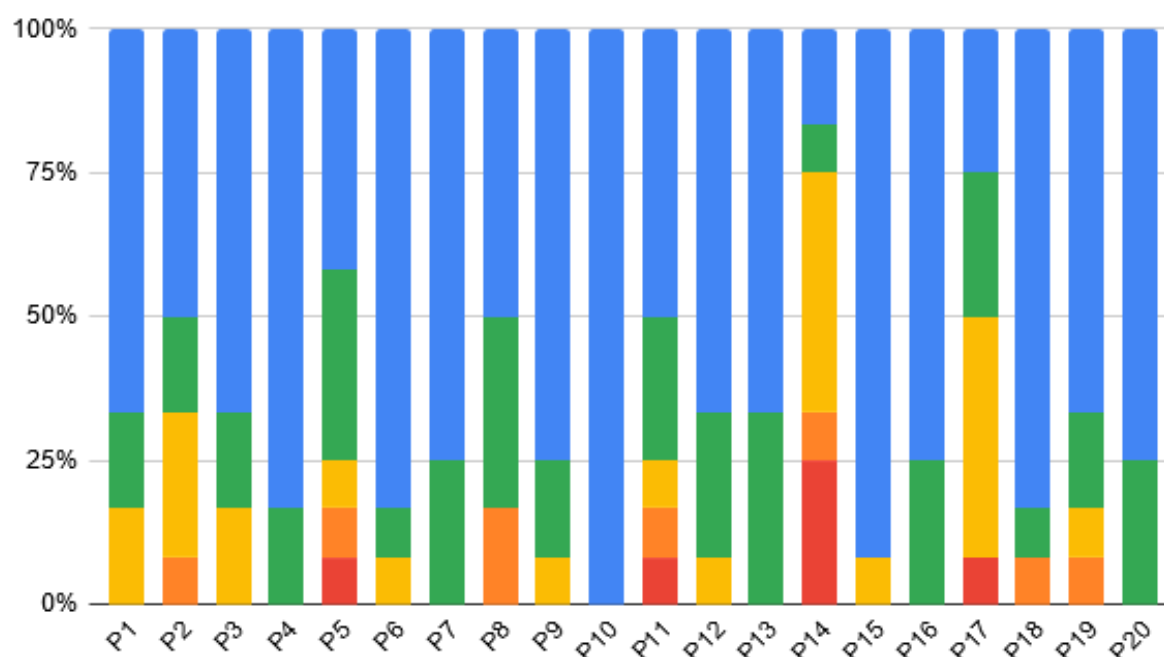


Figura 13 – Distribuição de respostas de avaliação do Flutter. (n=12)

populares. Novamente perdendo este início de aprendizado facilitado.

Nas respostas qualitativas, houve menções que são reflexos dos dados quantitativos, como: “Por ser uma ferramenta nova da Google, ainda é pequena a comunidade quando comparada a outras tecnologias no mercado. Mas este fator está mudando rapidamente” e “Aprender uma linguagem muito específica para utilizar a ferramenta”. E no outro lado do espectro, confirmando suas vantagens: “Animação fluida, modularização de widget e fluidez no aplicativo” e “O Flutter vem crescendo bastante e cada vez mais ganhando espaço no mercado”. No entanto não teve respostas que divergem do que foi confirmado através da análise quantitativa.

### 5.2.2 Ionic

Ionic é uma evolução tecnológica do *Phonegap*, tecnologia que permitia transformar um *website* em um aplicativo. Ao contrário do *framework* que o inspirou, o Ionic não produz um *website* que é executado em um navegador embutido no aplicativo resultante. O que o Ionic faz é interpretar o aplicativo *web* gerado durante tempo de execução do aplicativo resultante. A principal vantagem de utilizar desta abordagem é a possibilidade de integrar o Ionic a outros *frameworks* de desenvolvimento *web*, como *ReactJS*, *Angular* ou *VueJS*. Outro ponto importante é que apesar de gratuito, a distribuidora do Ionic oferece o Ionic Pro, que integra uma IDE com suporte a *Cloud* e possibilidade de suporte. Das ferramentas que obtiveram mais do que 5 avaliações, o Ionic possui uma média boa, com 4,28 (de 1 a 5). Na Figura 14 encontram-se as avaliações realizadas pelos 7 participantes que optaram responder sobre a ferramenta.

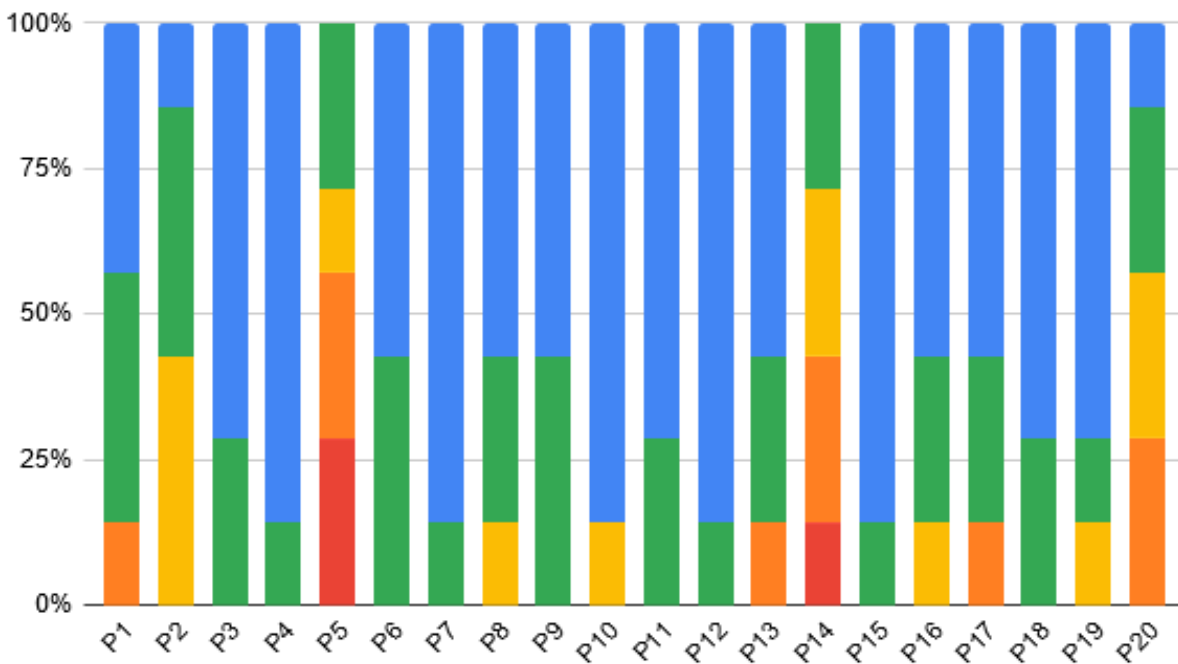


Figura 14 – Distribuição de respostas de avaliação do Ionic. (n=7)

Mesmo com a média ligeiramente inferior ao Flutter, o Ionic possui menos déficits em critérios, com apenas o P2, P5, P14 e P20. Começando pela P2 (Minha ferramenta é a melhor escolha disponível atualmente), não há nenhuma avaliação discordando dela, mas com 42% de “indiferentes” pode implicar que os desenvolvedores sabem que novas ferramentas estão conquistando o mercado. Já o P5 (Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta) pode ser um fator muito danoso à ferramenta, ter um acesso prejudicado às funções nativas pode ser um forte inibidor para vários desenvolvedores. P14 (Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir) continua um empecilho, mesmo em uma ferramenta que utiliza uma sintaxe muito próxima à *web*. O déficit do P20 (Meu aplicativo é tão rápido quanto um aplicativo nativo) pode ser outro problema crítico para a ferramenta, já que falta de fluidez e desempenho causa perda de usuários.

Nas respostas qualitativas houve comentários mistos sobre a extensibilidade da ferramenta, mas um participante com 2 anos de experiência com Apache/Cordova e 2 anos de experiência com Ionic disse: “Curva de aprendizado é muito grande, é fácil criar experiências horríveis, acessar recursos específicos do dispositivo as vezes é muito difícil e você acaba criando um plugin apenas para isso” e “Fornece componentes ótimos, se usar a base em Angular você possui muitas bibliotecas disponíveis, e pelo fato de ser basicamente JavaScript e HTML é possível ter acesos a muitas libs e códigos que economizam muito tempo”.

### 5.2.3 Kotlin

*Kotlin Cross-platform* é uma tentativa da JetBrains integrar a linguagem em uma ferramenta que produz aplicativos para *iOS* e *Android* de forma generativa. No momento da escrita desta dissertação ainda é um projeto com pouca presença de mercado de acordo com o *Google Trends*, mas que traz uma comunidade de admiradores do Kotlin nativo consigo. Esta baixa presença é refletida no questionário, que mesmo divulgado em fóruns sobre a linguagem angariou apenas dois participantes, ambos com experiência de aproximadamente um ano, e suas opiniões se encontram na Figura 15.

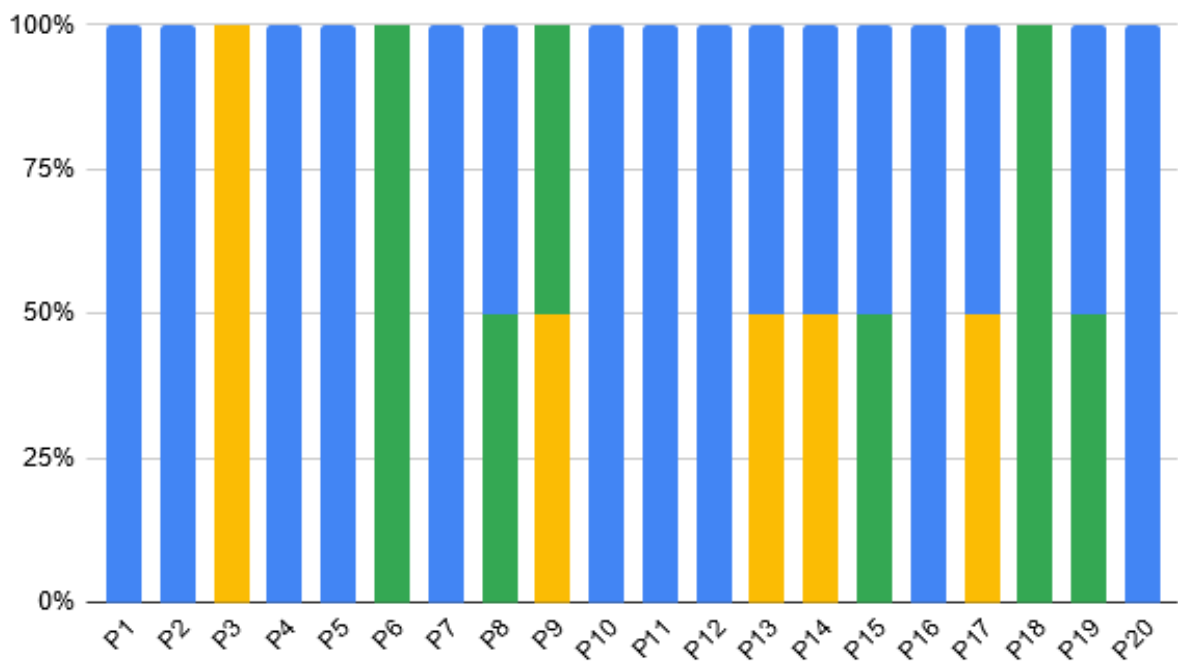


Figura 15 – Distribuição de respostas de avaliação do Kotlin. (n=2)

Mesmo com a baixa quantidade de participantes, a experiência de ambos é um fator importante para a relevância destes dados. O principal consenso deficitário é o P3 (Meu aplicativo é tão rápido quanto um aplicativo nativo), que seria de fato uma das principais preocupações para uma ferramenta que não possui tanta presença de mercado. Essa seria a dificuldade de encontrar recursos que auxiliariam o desenvolvedor a construir seu projeto. No entanto ambos concordam que isto é indiferente para a ferramenta, portanto não será necessariamente uma preocupação tão grande. Outro ponto observado é o P9, que trata sobre a instalação da ferramenta. Dificuldades de instalação podem gerar desistências quando um desenvolvedor tentar se iniciar na ferramenta.

Estas conclusões se confirmam através das respostas qualitativas, como nesta resposta às dificuldades encontradas: “Paralelismo (melhorando), documentação (melhorando) e instalação inicial (melhorando)”. O participante admite que estes problemas são reais, mas também percebe uma evolução. Isto é esperado de uma ferramenta que está co-

meçando neste mercado, e não seria irreal esperar uma utilização maior desta ferramenta no futuro.

#### 5.2.4 Qt

Qt é a ferramenta que especializa no nicho de ocupar diversas plataformas enquanto mantém um desempenho de ponta em todas elas. Pode ser utilizado com diversas linguagens, mas mais comumente com C++, justamente por esta questão de desempenho. Quanto ao suporte de plataformas, Qt é compatível Android, iOS e UWP para dispositivos móveis, Linux, macOS e Windows para desktop, possibilidade web, e também plataformas diversas de televisões smart e automóveis, e também inclui várias bibliotecas para desenvolvimento de jogos. Das ferramentas que obtiveram mais do que 5 avaliações, o Qt possui uma média com 4,13 (de 1 a 5). Na Figura 16 encontram-se as avaliações realizadas pelos 6 participantes que optaram por responder sobre a ferramenta.

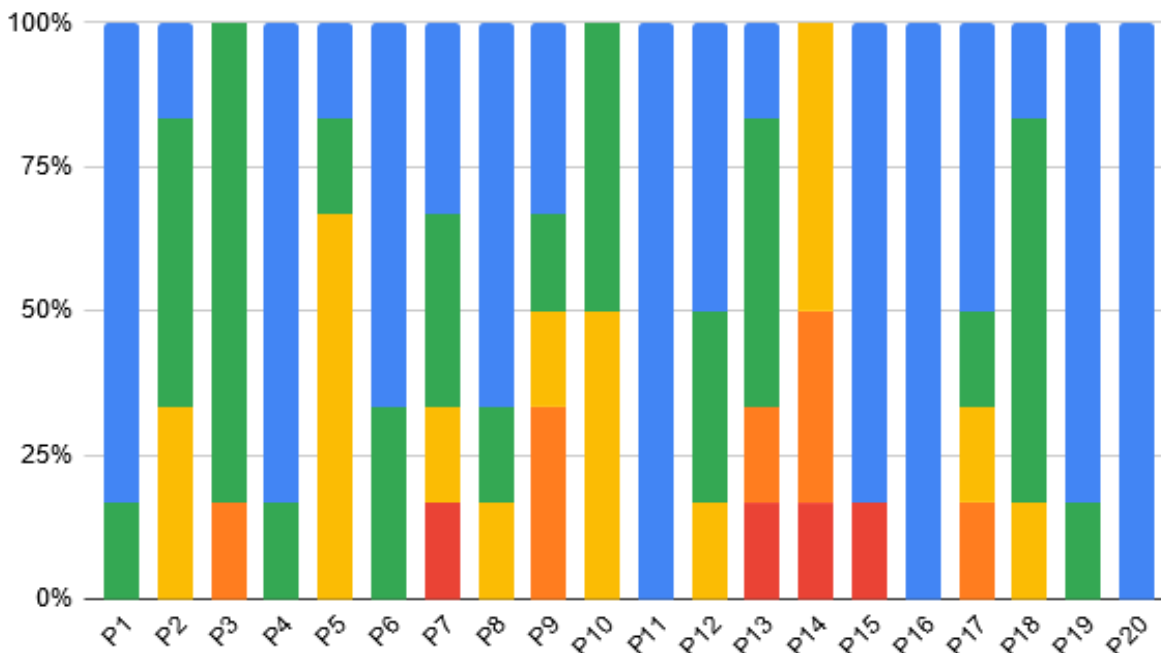


Figura 16 – Distribuição de respostas de avaliação do Qt. (n=6)

Mesmo com as supostas qualidades destacadas pelo fabricante o Qt apresentou déficit no P5, P7, P9, P10, P13, P14 e P17. Começando pelo P5 (Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta), o problema com o acesso às APIs nativas, este problema poderia estar vinculado ao fato de abranger tantas plataformas diferentes. Esta questão retorna no déficit do P7 (Posso publicar meus aplicativos aonde quiser), onde fica indicado problemas de publicação, talvez algumas das plataformas suportadas não estejam tão bem integradas. P9 (Consigo instalar minha ferramenta rapidamente) também ainda pode ser enquadrado neste padrão, a instalação da ferramenta varia de plataforma para plataforma. O P13 (Meu aplicativo se parece nativo mesmo sem



esforço) também se encaixa nesse padrão, já que operar em um nível baixo de abstração inviabiliza o uso de uma aparência nativa sem esforço extra. P10 (Acho justo o que pago ou deixo de pagar em minha ferramenta) mesmo não sendo uma das principais queixas, é um problema bastante alarmante, o preço de uma licença apenas para *desktop e mobile* para um desenvolvedor por um ano custa U\$5.500,00, dificultando o uso para pequenas empresas. P14 (Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir) e P17 (Utilizo de tecnologias que muitas outras pessoas estão utilizando) estão relacionados, mesmo que utilizem linguagens populares (que nem sempre é o caso), a arquitetura do Qt é muito diferente do habitual, dificultando migrações.

Nas respostas qualitativas há algumas queixas de bugs infrequentes e dificuldade de conexão com grandes bases de dados. E todos os participantes concordam em um fator: “desempenho”. Uma das adições é uma confirmação de que Qt é indicado para grandes empresas, que precisam desta abrangência e desempenho, sem se importar com valores ou necessidade de esforço extra para tarefas menores.

### 5.2.5 React Native

React Native é a versão multiplataforma do ReactJS, ambas ferramentas de desenvolvimento suportada e utilizada pelo Facebook. A ferramenta tem a premissa de transferir a experiência de sua versão web para a versão multiplataforma. Utiliza de alto índice de componentização e uma arquitetura diferenciada do habitual. A ferramenta possui uma comunidade imensa e no momento da escrita desta dissertação é a mais popular no mercado. Foi a ferramenta com a maior quantidade de avaliações, com 15 participantes, mas apenas 3,6 (de 1 a 5) de média, a pior classificada. Na Figura 17 encontram-se as avaliações realizadas pelos participantes que optaram responder sobre a ferramenta.

Contando apenas com 6 critérios com mais de 75% dos participantes concordam com sua importância, os déficits que mais chamam a atenção são P2, P5, P14, P19 e P20. P2 (Minha ferramenta é a melhor escolha disponível atualmente) é apenas uma opinião, sem uma direção precisa do que significa, mas é um indicador relevante para aprender sobre o que pensam seus usuários, e neste caso estão bastante divididos. Novamente o P5 (Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta) gerando preocupações com o acesso às APIs nativas, indicando que o React pode não ser uma boa opção para aplicativos que façam bastante uso delas. A preocupação do P14 (Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir) se repete de forma bastante intensificada na ferramenta, talvez a sua arquitetura seja um empecilho em casos de uma migração forçada. P19 (Estou seguro de que a ferramenta continuará existindo e evoluindo por muito tempo) é a métrica mais relevante para o estado atual da ferramenta, seria este um indicador de que o React

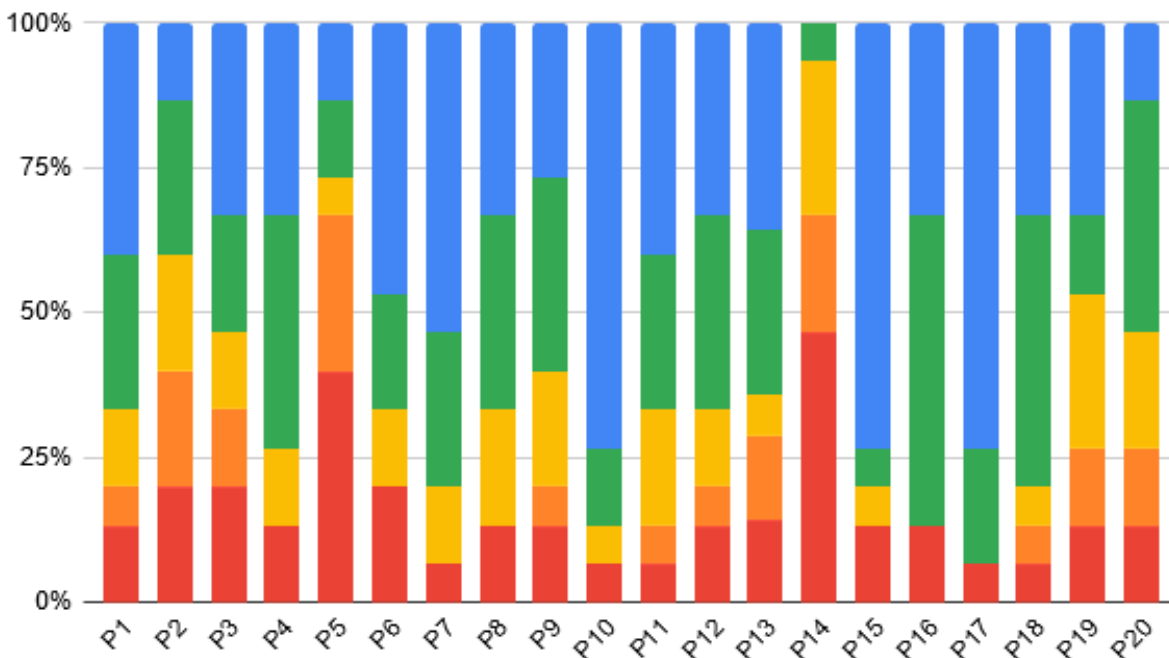


Figura 17 – Distribuição de respostas de avaliação do React Native. (n=15)

já atingiu seu ápice? Por fim P20 (Meu aplicativo é tão rápido quanto um aplicativo nativo) indicando um desconforto com o desempenho e fluidez do aplicativo resultante.

Nos resultados qualitativos há muitas críticas que refletem diretamente o observado no quantitativo, chamando a atenção apenas: “Se não for um aplicativo com desenvolvimento contínuo e precisar atualizar apenas depois de um ano será muito doloroso. Ele não param de realizar mudanças bruscas na forma de desenvolvimento”. No lado dos elogios há menções ao seu desenvolvimento rápido, com facilidade de desenvolver UIs e animações e a sua imensa comunidade.

## 5.2.6 Desenvolvimento Web

Desenvolvimento Web foi a primeira aparição do desenvolvimento multiplataforma já que os primeiros smartphones que possuíam navegador já eram capazes de acessar qualquer página. Claro que com o tempo a tecnologia foi se refinando ao ponto de aplicativos *web* serem em alguns cenários quase indistinguíveis de um nativo ao utilizar de um PWA. O método mais comum atualmente ainda é a utilização de *frameworks* que permitem mais expressividade e funcionalidade com o JavaScript. Das ferramentas que obtiveram mais do que 5 avaliações, o desenvolvimento web possui uma média de 3,85 (de 1 a 5). Na Figura 18 encontram-se as avaliações realizadas pelos 9 participantes que optaram responder sobre a ferramenta. Destes 9 participantes, 3 utilizam de ReactJS, 2 de Angular, 2 de VueJS e 2 de JavaScript + Bootstrap apenas.

A abordagem recebeu apenas dois critérios com 75% dos participantes concor-

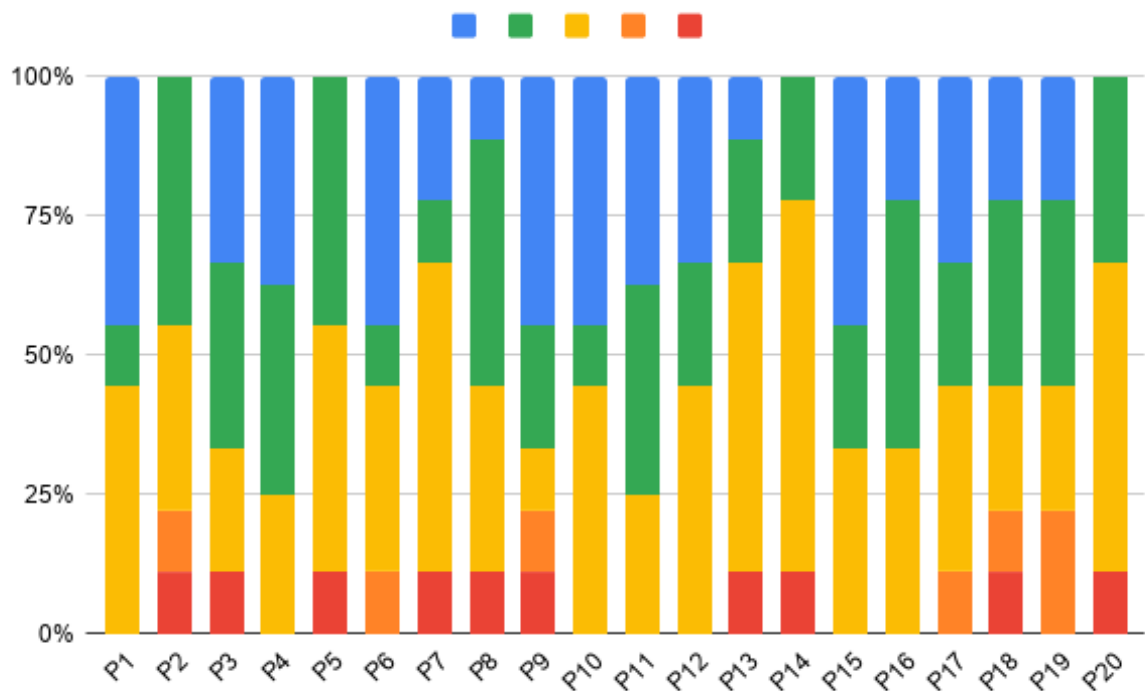


Figura 18 – Distribuição de respostas de avaliação do desenvolvimento web. (n=7)

dando com sua importância, mas os déficits mais chamativos são P7, P13, P14 e P20. P7 (Posso publicar meus aplicativos aonde quiser) é um dos maiores problemas da abordagem atualmente, não é possível publicar aplicativos web em nenhuma loja de aplicativos, o que acaba inibindo muitos desenvolvedores. P13 (Meu aplicativo se parece nativo mesmo sem esforço) também pode consumir muito do tempo de desenvolvimento se o framework não proporcionar formas de amenizar este problema. P14 (Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir) não era esperado de ser um problema, já que ferramentas da abordagem híbrida deveriam suprir esta demanda. Talvez este déficit ocorra por uma queda na popularidade desta abordagem. Já P20 (Meu aplicativo é tão rápido quanto um aplicativo nativo) também é outro problema recorrente, já que sem utilizar um PWA, a navegação além de pouco fluída se torna dependente da conexão do usuário.

### 5.2.7 Xamarin

Xamarin é uma das ferramentas mais antigas que se mantém relevante no mercado. Suportada pela Microsoft, é uma ferramenta que produz aplicativos nativos utilizando C# e .NET. Uma das vantagens de seu uso é sua flexibilidade, permitindo o desenvolvedor escolher a quantidade de código compartilhado que seu projeto precisa. Isso ocorre na configuração inicial do projeto ao escolher sua forma de integração. Das ferramentas que obtiveram mais do que 5 avaliações, o Xamarin possui uma média de 3,74 (de 1 a 5). Na Figura 19 encontram-se as avaliações realizadas pelos 11 participantes que optaram por

responder sobre a ferramenta.

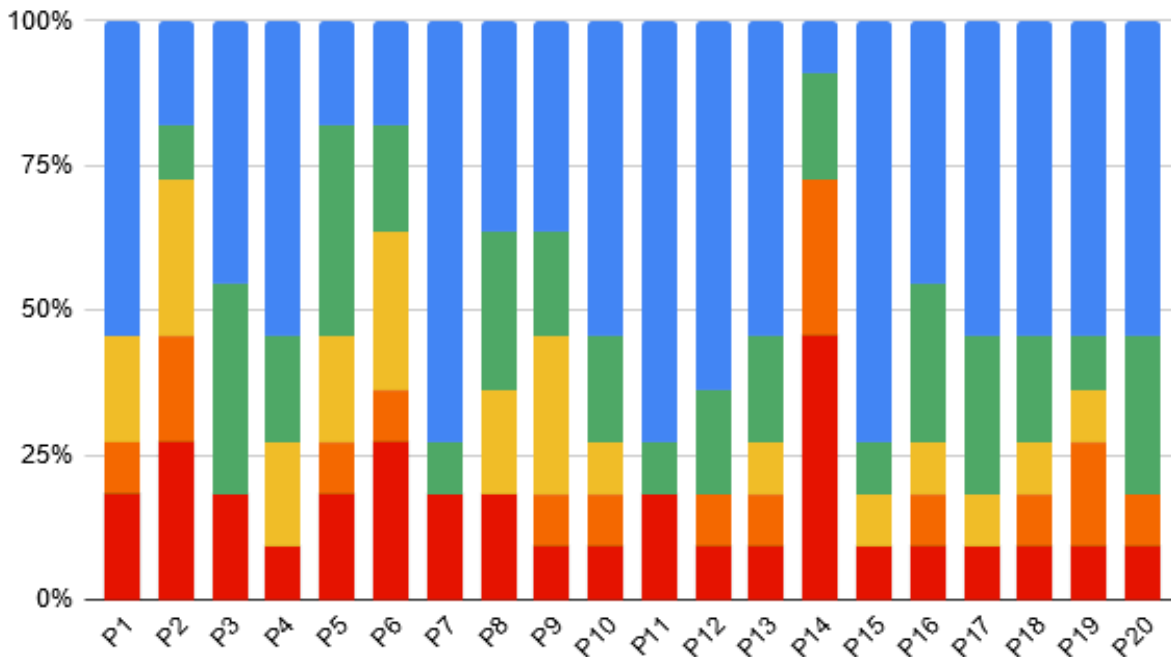


Figura 19 – Distribuição de respostas de avaliação do Xamarin. (n=11)

Dos déficits encontrados os mais chamativos são P1, P2, P5, P6 e P14. P1 (Quanto você recomenda esta ferramenta para outras pessoas interessadas em começarem com desenvolvimento multiplataforma?) e P2 (Minha ferramenta é a melhor escolha disponível atualmente) são indicativos da posição que Xamarin ocupa em relação a outras ferramentas do ponto de vista dos usuários do Xamarin. Esse valor baixo de concordância (respostas “fundamental” e “quase fundamental” somadas) indica que a comunidade não está completamente contente com o estado atual. O déficit do P5 (Nunca tive problema em acessar alguma função do dispositivo com minha ferramenta) indica que pode ser trabalhoso acessar APIs nativas com esta ferramenta. Enquanto P6 (Conseguo desenvolver interfaces de usuário rapidamente ou do jeito que eu desejar) expressa a frustração de muitos desenvolvedores com a forma de desenvolvimento de UI do Xamarin. Por fim P14 (Tenho a segurança de conseguir migrar meu aplicativo facilmente caso minha ferramenta esteja me impedindo de progredir) continua sendo uma preocupação global, poucas ferramentas são feitas com a preocupação em mente, e o diferente estilo de arquitetura do Xamarin não facilitaria sua migração.

As principais queixas foram: “Xamarin.Forms é cheia de bugs, grande e lenta; Xamarin Native é muito melhor, mas não é uma grande diminuição no esforço multiplataforma” e “Quando o framework atualiza as vezes causa problemas com dependências e isso leva muito tempo para resolver”. Enquanto os elogios são ao seu ambiente de desenvolvimento que possui um “ecossistema completo”.

Para melhor ilustrar os dados de cada ferramenta, eles foram condensados na

Tabela 5 exibindo as Qualidades e Déficits de cada ferramenta. Na tabela, Qualidades se referem àqueles critérios os quais os desenvolvedores acreditam serem bem atendidos pela ferramenta, e Déficits são aqueles os quais os desenvolvedores acreditam não serem bem atendidos pela ferramenta.

<b>Ferramenta</b>	<b>Participantes</b>	<b>Qualidades</b>	<b>Déficits</b>
Flutter	12	P10, P4, P15, P6, P7, P16 e P20	P14, P17, P5, P11 e P2
Ionic	7	P4, P7, P12, P15, P3, P10, P11 e P18	P5, P14, P20, P2 e P1
Kotlin	2	P1, P2, P4, P5, P7, P10, P11, P12, P16, P20	P3, P9, P6, P13, P14 e P17
Qt	6	P11, P16, P20, P1, P4 e P19	P14, P13, P5, P9 e P10
React Native	15	P17, P10, P15, P7, P16 e P18	P14, P5, P2, P20 e P3
Desenvolvimento Web	7	P15, P6, P11, P3, P4, P9, e P10	P2, P20, P14, P5 e P13
Xamarin	11	P15, P7, P11, P12 e P17	P14, P2, P6, P4 e P9

Tabela 5 – Qualidades e Déficits de cada ferramenta presente no questionário

### 5.3 Codificação

Aqui são discutidos os resultados da codificação. Como discutido no Capítulo 3, adotou-se uma técnica da Grounded Theory para identificar observações e tirar conclusões a partir de diferentes fontes e estudos. O objetivo foi identificar critérios para escolha das ferramentas, e pontos fortes/fracos de cada ferramenta.

Ao todo, para a codificação foram utilizados 37 artigos científicos (os mesmos apresentados no Capítulo 4), 10 artigos provenientes de literatura cinza, 10 vídeos e 7 entrevistas. Os artefatos que não são entrevistas estão listados na Tabela 6. Nesta mesma Tabela 6 também estão os identificadores de cada artefato que serão utilizados ainda nesta seção. Nesta mesma tabela também estão os entrevistados com uma breve descrição de sua experiência.

<b>Identificador</b>	<b>Entrevistados</b>
E1	Desenvolvedor Full Stack com 2 anos de experiência com Ionic;

E2	Acadêmico e um dos desenvolvedores de uma grande ferramenta de desenvolvimento multiplataforma de abordagem dirigida por modelos. Possui experiências prévias com Phonegap;
E3	Desenvolvedor Mobile com 1 ano e meio de experiência com React Native;
E4	Desenvolvedor Mobile com 2 anos de experiência com Xamarin, e alguns meses com desenvolvimento nativo e PhoneGap;
E5	Desenvolvedor Web com 1 ano de experiência com Native Script;
E6	Sócio-desenvolvedor com 2 anos de experiência com React Native e 1 ano com desenvolvimento nativo; e
E7	C.E.O. de Consultoria de Desenvolvimento de Software, com experiência liderando várias equipes que utilizaram de Ionic e React Native, mas possuem preferência por desenvolvimento nativo.
<b>Identificador</b>	<b>Artigos Científicos</b>
A1	<i>A Comparative Analysis of Cross-platform Development</i> - Xanthopoulos e Xinogalos (2013)
A2	<i>A Performance Evaluation of Cross-Platform Mobile Application</i> - Jia, Ebone e Tan (2018)
A3	<i>A Students Perspective of Native and Cross-Platform Approaches</i> - Meirelles et al. (2019)
A4	<i>A Study on Approaches to Build Cross-platform Mobile Applications and Criteria to Select Appropriate Approach</i> - Raj e Tolety (2012)
A5	<i>A Survey and Taxonomy of Core Concepts and Research Challenges</i> - Biorn-Hansen, Gronli e Ghinea (2018)
A6	<i>An Empirical Study of Investigating Mobile Applications Develop</i> - Ahmad et al. (2018)
A7	<i>An Evaluation Framework for Cross-Platform Mobile App Development Tools: A case Analysis of Adobe PhoneGap framework</i> - Ahti, Hyrynsalmi e Nevalainen (2016)
A8	<i>An evaluation framework for cross-platform mobile application development tools</i> - Dhillon e Mahmoud (2015)
A9	<i>An Evaluation Framework for Selection of Mobile App Development Platform</i> - Hudli, Hudli e Hudli (2015)
A10	<i>An Evaluation of Cross-Platform Frameworks for Multimedia Mobile Applications Development</i> - Ferreira et al. (2018)
A11	<i>Beyond Native Apps Web Technologies to the Rescue!</i> - Malavolta (2016)

A12	<i>Beyond Web/Native/Hybrid A New Taxonomy for Mobile App Development - Nunkesser (2018)</i>
A13	<i>Challenges of Mobile Applications Development Initial Results - Ahmad et al. (2017)</i>
A14	<i>Comparative analysis of tools for development of native and hybrid mobile applications - Vilcek e Jakopec (2017)</i>
A15	<i>Comparison of Cross-Platform Mobile Development Tools - Palmieri, Singh e Cicchetti (2012)</i>
A16	<i>Comprehensive Analysis of Innovative Cross-Platform App Development - Majchrzak, Biorn-hansen e Gronli (2017)</i>
A17	<i>Cross-platform approach for mobile application development A survey - Latif et al. (2016)</i>
A18	<i>Cross-Platform Development for an online Food Delivery Application - Abid e Karim (2017)</i>
A19	<i>Cross-platform Mobile Applications development - Redda (2012)</i>
A20	<i>Cross-platform Mobile Development Approaches A Systematic Review - Bernardes e Miyake (2016)</i>
A21	<i>Cross-platform Mobile Development Approaches - Charkaoui, Adraoui e Habib Benlahmar (2014)</i>
A22	<i>Design and Implementation of Cross-platform Friends-Positioning Mobile APP Based on Phonegap and HTML5 - Fan e Yang (2017)</i>
A23	<i>Development Frameworks for Mobile Devices A Comparative Study about Energy Consumptions - Corbalan et al. (2018)</i>
A24	<i>From Native to Cross-platform Hybrid Development - Pinto e Coutinho (2018)</i>
A25	<i>Mobile Application Development Research Based on Xamarin Platform - Vishal e Kushwaha (2019)</i>
A26	<i>Model Driven Approaches to Cross-Platform Mobile Development - Gaouar, Benamar e Bendimerad (2015)</i>
A27	<i>Model-Driven Development of Mobile Applications A Systematic Literature Review - Tufail et al. (2018)</i>
A28	<i>On the Static Analysis of Hybrid Mobile Apps A Report on the State of Apache Cordova Nation - Brucker e Herzberg (2016)</i>
A29	<i>Progressive Web Apps for the Unified Development of Mobile Applications - Biorn-Hansen, Majchrzak e Gronli (2017)</i>
A30	<i>Progressive Web Apps the Definite Approach to Cross-Platform Development - Majchrzak, Biorn-Hansen e Gronli (2018)</i>

A31	<i>Selecting the best mobile framework for developing web and hybrid mobile apps</i> - Botella, Escribano e Peñalver (2016)
A32	<i>Survey, Comparison and Evaluation of Cross-Platform Mobile Application Development Tools</i> - Dalmasso et al. (2013)
A33	<i>Survey on Cross-Platforms and Languages for Mobile Apps</i> - Ribeiro e Da Silva (2012)
A34	<i>Survey on Techniques for Cross-Platform Mobile Application Development</i> - Pawar, Jagtap e Bhamare (2014)
A35	<i>Taxonomy of Cross-Platform Mobile Applications Development Approaches</i> - El-Kassas et al. (2017)
A36	<i>Usability Testing of Mobile Applications Web vs. Hybrid Apps</i> - Koziokas, Tselikas e Tselikis (2017)
A37	<i>Weighted Evaluation Framework for Cross-Platform App Development Approaches</i> - Rieger e Majchrzak (2016)
<b>Identificador</b>	<b>Artigos provenientes de literatura cinza</b>
I1	<i>9 Best Cross-Platform Mobile App Development Tools</i> - Richard (2017)
I2	<i>Cross-Platform Mobile App Development Guide (2019)</i> - Dogtiev (2019)
I3	<i>Cross-Platform Mobile Development in 2018 A Beginner's Guide</i> - Thomas (2018)
I4	<i>Cross-Platform Mobile Development Tools: Ending the iOS vs. Android Debate</i> - Baldwin (2019)
I5	<i>Flutter, Ionic, React Native or Xamarin? What you use and why?</i> - Moumni (2018)
I6	<i>React Native vs Xamarin vs Ionic</i> - Dyvliash (2019)
I7	<i>React Native Vs Xamarin. What to choose for cross-platform app development?</i> - Aggarwal (2019)
I8	<i>Top 10 Cross-Platform Mobile Development Tools</i> - KS (2018)
I9	<i>Xamarin vs Ionic vs React Native: differences under the hood</i> - Cruxlab (2017)
I10	<i>Xamarin vs React Native vs Ionic vs NativeScript: Cross-platform Mobile Frameworks Comparison</i> - Altexsoft (2018)
<b>Identificador</b>	<b>Vídeos</b>
V1	<i>Choosing the best mobile app framework</i> - Startups (2019)
V2	<i>Hybrid App vs native app</i> - Choudhary (2017)
V3	<i>Ionic 4 v/s React Native - A detailed analysis of both</i> - codedamn (2018)



V4	<i>Mobile Apps - Web vs. Native vs. Hybrid - Media (2017)</i>
V5	<i>Mobile development in 2019: native versus cross-platform - Busch (2019)</i>
V6	<i>React Native vs Flutter vs WebView - Hybrid Mobile App Development for 2018 - Techlead (2018)</i>
V7	<i>The Future of Mobile Development - Mischook (2019)</i>
V8	<i>Top 10 Mobile Frameworks For App Development   Mobile App Development Frameworks - Solutions (2017)</i>
V9	<i>The Mobile Frameworks Battle   Flutter Vs React Native Vs Native App Development - Solutions (2018)</i>
V10	<i>What Mobile Development Framework Should I Learn in 2017? - Halliday (2017)</i>

Tabela 6 – Artefatos utilizados na codificação e identificador atribuído

A partir da codificação deste artefatos (Tabela 6) e das entrevistas foi criada a classificação representada na Figura 20. Classificações são agrupamentos de códigos criados pelos pesquisadores a partir da observação da origem e utilização dos segmentos codificados. Desta forma é possível agrupar os códigos dentro de categorias que compartilham hierarquicamente algum significado. Como resultado neste caso específico é possível extrair a divisão dos critérios de forma lógica e baseada em dados.

No caso da Figura 20 a classificação dos critérios foi descrita através de um mapa mental. Mapas mentais são diagramas hierárquicos que permitem mostrar relações entre objetos de um todo. Neste mapa mental foi utilizado de um aumento na grossura das linhas hierárquicas para mostrar relativamente a quantidade de códigos encontrados. Já a coloração dos nódulos são apenas para agrupar objetos que estão no mesmo nível hierárquico de forma mais clara. Também há o uso de setas que são usadas para indicar uma relação mais próxima entre objetos de mesmo nível hierárquico.

Na Tabela 7 se encontram todos os códigos descobertos. Eles estão ordenados de forma decrescente de acordo com seu índice de aparecimento. Isto é a quantidade de documentos que possuem pelo menos um código em relação a quantidade total de documentos. Também estão listados todos os artefatos em que cada código aparece.

Código Pai	Código	%	Artefatos
Critérios	Desempenho	51,85	A1, A2, A3, A4, A5, A7, A8, A10, A17, A18, A20, A23, A27, A28, A29, A32, A33, A35, A37, E1, E3, I1, I5, I6, I9, I10, V1 e V9

Custo de Desenvolvimento	Uso de APIs Nativas	40,74	A1, A5, A6, A8, A9, A15, A16, A17, A20, A21, A25, A27, A29, A31, A33, A34, A35, A37, E1, E4, E7 e V1
Custo de Desenvolvimento	Desenvolvimento de UI	35,18	A1, A3, A5, A6, A7, A16, A18, A20, A25, A27, A32, A37, E4, I5, I9, 10, V1, V6 e V9
Curva de Aprendizado	Linguagem	29,62	A3, A4, A10, A14, A15, A17, A21, A24, A25, A33, A34, A35, E4, I9, I10 e V1
Distribuição Facilitada	Suporte a Plataformas	27,77	A4, A8, A14, A15, A16, A21, A24, A29, A33, A34, A35, A37, E4, I5 e I10
Curva de Aprendizado	Tecnologia difundida	25,92	A1, A3, A9, A20, E1, E3, E4, E7, I2, I5, I7, I10, V2 e V4
Critérios	Custo de Desenvolvimento	25,92	A3, A6, A7, A9, A14, A16, A18, A21, A24, A27, A31, A32, A37 e V1
Custo de Desenvolvimento	Ambiente de Desenvolvimento	20,37	A8, A15, A16, A17, A24, A25, A32, A33, A34, A35 e A37
Popularidade	Comunidade Ativa	16,66	A3, A10, A14, A16, E1, E3, E4, I1 e I10
Maturidade	Segurança	12,96	A5, A6, A8, A9, A17, A32 e A37
Critérios	Distribuição Facilitada	11,11	A1, A20, A21, A29, A37 e E3
Custo de Desenvolvimento	Ferramentas de Teste	11,11	A6, A9, A16, A29, A37 e V9
Curva de Aprendizado	Documentação	11,11	A10, A14, E1, E2, E3 e E4
Custo de Desenvolvimento	Facilidade de Desenvolvimento	11,11	A32, A37, E3, E4, V1 e V9
Maturidade	Curva de Aprendizado	9,25	A3, A37, E1, E3 e E4
Custos	Licença	9,25	A8, A15, A34, A37 e I6
Maturidade	Viabilidade a longo prazo	7,4	A32, A37, E1 e E7

Critérios	Escalabilidade	7,4	A17, A27, A28, e A37
Critérios	Custos	7,4	A8, A9, A24 e I10
Desenvolvimento de UI	Naturalidade	7,4	A37, I5, V6 e V9
Curva de Aprendizado	Facilidade de Instalação	7,4	A3, A14, A16 e E3
Viabilidade a longo prazo	Open Source	7,4	A15, A27, I7 e V1
Facilidade de Desenvolvimento	Custo de Manutenção	5,55	A32, A37 e V1
Critérios	Maturidade	5,55	I5, V1 e V9
Comunidade Ativa	Extensibilidade	5,55	A9, A15 e A37
Viabilidade a longo prazo	Suporte (Apoio Financeiro)	3,7	A28 e I5
Facilidade de Desenvolvimento	Quantidade de Código Compartilhado	1,85	I6
Curva de Aprendizado	Passagem de Conhecimento	1,85	E1
Custos	Monetização	1,85	A37
Maturidade	Popularidade	1,85	I10
Facilidade de Desenvolvimento	Alterar Código Nativo	1,85	A37
Facilidade de Desenvolvimento	Arquitetura	1,85	A15
Viabilidade a longo prazo	Atualização Frequente	1,85	V5
Uso de APIs Nativas	Controle Offline/Online	1,85	E4
Arquitetura	Produção de Aplicativos Complexos	1,85	E3
Desempenho	Uso de Gráficos Intensivos	1,85	I1
Critérios	Desinteresse em migração	1,85	E1
Uso de APIs Nativas	Controle de Impressão	1,85	E1
Uso de APIs Nativas	Controle de Status de Rede	1,85	E1

---

Tabela 7 – Porcentagem de aparecimento de cada código de critério

---

Como observado na Figura 20 e Tabela 7 foram encontrados 6 critérios base, sendo eles: Custos; Escalabilidade; Custo de Desenvolvimento; Distribuição Facilitada; Maturidade e Desempenho. Isto pode implicar que a avaliação de uma ferramenta pode ocorrer apenas com estes critérios, utilizando os critérios filhos destes como forma de avaliação apenas.

O **Custo de Desenvolvimento** é um dos critérios mais relevantes de acordo com os códigos (Tabela 7). Ele se divide em: Ambiente de Desenvolvimento; Ferramentas de Teste; Uso de APIs Nativas; Desenvolvimento de UI; e Facilidade de Desenvolvimento. Ferramentas de Teste e Ambiente de Desenvolvimento estão diretamente ligados, já que não é incomum ambientes de desenvolvimento já virem acompanhados de ferramentas de teste. Desenvolvimento de UI possui um código filho referente à naturalidade, já que a ferramenta produzir aplicativos com alta naturalidade diminuirá a quantidade de desenvolvimento de UI e consequentemente de desenvolvimento ao todo. Já a Facilidade de Desenvolvimento pode ser avaliada por sua arquitetura, custo de manutenção, possibilidade de alterar código nativo e por quantidade de código compartilhado entre as plataformas suportadas.

Já o **desempenho** que é o critério mais codificado, não possui códigos filho. Isto não significa que não seja possível criar uma subdivisão para ele, apenas que no contingente de artefatos codificados não foram encontrados códigos relacionados. Porém já existe na literatura um artigo que supre esta necessidade. Jia, Ebone e Tan (2018) realiza uma avaliação geral do desempenho de aplicativos produzidos por diferentes ferramentas, e a forma que divide o desempenho em critérios é bastante eficiente.

Os **custos** são sempre um critério importante para os desenvolvedores, já que não são todos que podem arcar com eles. Este critério pode ser avaliado através do quanto é necessário pagar para utilizar a ferramenta. Mas esta não é a única forma, já que também podem existir ferramentas gratuitas mas que possuem uma licença comercial que exigem uma porcentagem caso o aplicativo comece a gerar lucro. E o contrário também deve ser considerado, algumas ferramentas possibilitam uma monetização facilitada do seu aplicativo.

A **escalabilidade** é outro critério que não houve códigos relacionado a ela. Mas no geral a escalabilidade pode ser avaliada através de facilitadores de compartilhamento de código, segurança e até mesmo como o tamanho total do aplicativo aumenta com a quantidade de telas e código produzido. Outro fator importante é a possibilidade de trabalhar com o código em um baixo nível, já que desempenho se torna muito importante quando é possível dedicar uma equipe especialmente para isto.

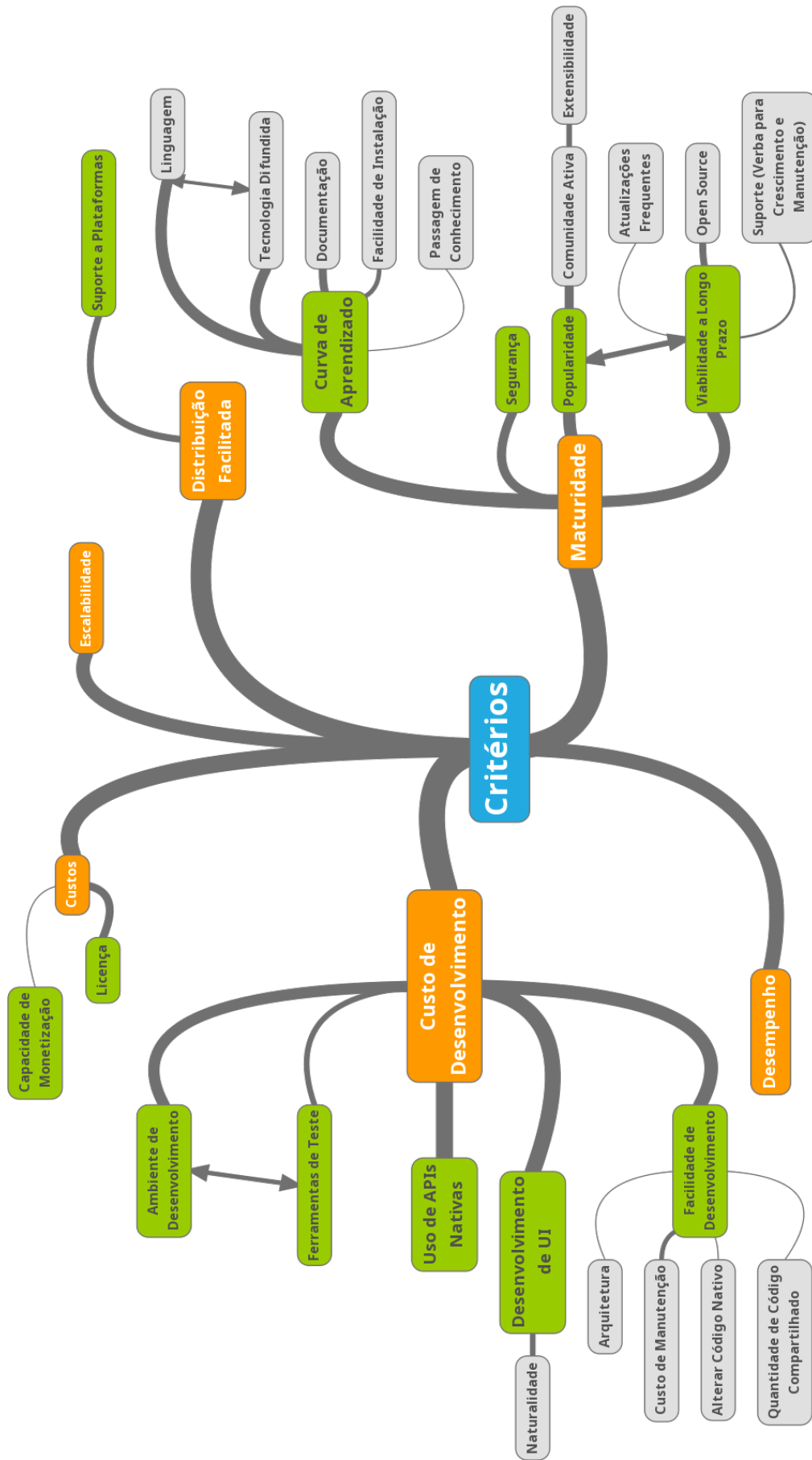


Figura 20 – Mapa mental com a classificação dos critérios descobertos durante a codificação

A **distribuição facilitada** é a capacidade do aplicativo ser distribuído, não apenas por lojas de aplicativos, mas também por links, compartilhamento direto de APKs, ou qualquer forma de compartilhar o seu aplicativo com outros usuários. O código de suporte a plataformas se relaciona com a distribuição no sentido de que as plataformas são apenas os meios em que a distribuição ocorre.

E finalmente a **maturidade** da ferramenta que é outro fator de extrema importância na escolha dos desenvolvedores. Começando pela curva de aprendizado, que é um código filho da maturidade pois foi observado que ferramentas mais maduras possuem curvas de aprendizado menos íngremes. A curva de aprendizado pode ainda ser avaliada através de linguagem, uso de tecnologias difundidas, documentação, facilidade de instalação e passagem de conhecimento. O quesito de segurança também é muito relevante, e também foi notado que segurança se torna uma preocupação crescente conforme a ferramenta evolui. A popularidade da ferramenta indica sua maturidade já que quanto mais uma ferramenta se estabelece, maior cresce sua comunidade. E quanto maior a comunidade mais haverá pessoas desenvolvendo bibliotecas, aumentando sua extensibilidade. Já a viabilidade a longo prazo se relaciona diretamente com a popularidade devido à observação de que as ferramentas que mais duraram, foram ferramentas bastante populares. E se uma ferramenta for open source, este vínculo se intensifica ainda mais.

As Tabelas 8 a 15 são os códigos obtidos para cada ferramenta encontrada. Junto a elas também haverá alguns comentários sobre suas implicações.

<b>Código</b>	<b>Qtd</b>	<b>Artefatos</b>
Facilidade de Teste	1	V9
Facilidade de Desenvolvimento	1	V1
Viável a longo prazo	1	I5
Naturalidade	2	V1 e V9
Imatura	2	I5 e V1
Desempenho	3	I5, V1 e V9

Tabela 8 – Quantidade de códigos encontrados para Flutter

A nova ferramenta do Google, **Flutter** (Tabela 8), foi muito bem aclamada por diversos fatores. No entanto sua principal crítica é a falta de maturidade. A ferramenta é de fato muito nova, lançada em maio de 2017, mas vem conquistando uma comunidade cada vez maior. E como visto na Seção 5.2 é possível que seja a próxima ferramenta mais utilizada no futuro.

Já o **Ionic** (Tabela 9), sucessor tecnológico do Phonegap, possui também vários códigos positivos e otimistas. No entanto a preocupação com o desempenho prejudicado continua existindo. Mas caso desempenho não seja tão importante para seu aplicativo, ainda é uma opção mais que sólida.

Código	Qtd	Artefatos
Alta Popularidade	1	E3
Open Source	1	I6
Não trabalha nativamente com a Apple	1	E1
Novas versões adicionam muitas opções de customização	1	E1
Atualização para nova versão relativamente rápida	1	E1
Falta de Controle de Câmera	1	E1
Controle de Status de Rede insuficiente	1	E1
Limitações toleráveis	1	E1
Controle de Impressão disponível	1	E1
Boa Escalabilidade	1	E1
Possibilidade de Otimização	1	E1
Arquitetura MVC	1	E3
Comunidade ativa	2	E1 e V9
Curva de Aprendizado boa	2	A24 e E1
Viável a longo prazo	2	E1 e I5
Desempenho Inferior ao Nativo	3	A24, I6, I10 e V10
Desenvolvimento de UI Eficaz	4	A16, A31, E1, I6
Tecnologia Difundida	4	A24, I10, V3 e V9

Tabela 9 – Quantidade de códigos encontrados para Ionic

Código	Qtd	Artefatos
Produção em Massa	1	E2
Maior nível de abstração	1	E2
Documentação escassa	1	E2
Curva de Aprendizado Acentuada	1	E2
Manutenibilidade e Reusabilidade	1	E2
Produtividade muito alta	1	E2
Dificuldade de Instalação	1	E2
Foco acadêmico	1	E2
Ajustes Finos	1	E2

Tabela 10 – Quantidade de códigos encontrados para MD<sup>2</sup>

**MD<sup>2</sup>** (Tabela 10) é a ferramenta mais mencionada e popular dentre as ferramentas da abordagem dirigidas por modelos. E como característica da abordagem, esta ferramenta possui positivos muito fortes, e negativos muito prejudiciais. “Produção em massa”, “Alto nível de abstração”, “Manutenibilidade e reusabilidade”, “Produtividade muito alta” e “Ajustes Finos” são todas importantíssimas, mas pode acabar inibidas pela “Documentação escassa”, “Curva de Aprendizado acentuada” e “Dificuldade de Instalação”, características que dificultam muito o engajamento inicial com a ferramenta.

**Desenvolvimento Nativo** (Tabela 11) é a principal forma de desenvolvimento móvel atualmente, e isto pode ser explicado pelos códigos encontrados. Começando pelo desempenho que é basicamente imbatível e crucial para aplicações que utilizam intensa-

Código	Qtd	Artefatos
Maturo	1	V9
Adequado para aplicativos complexos e de duração extensa	1	E7
Ajustes Finos	1	E7
Distribuição	1	V4
Facilidade de Teste	1	V9
Comunidade Ampla	2	A10 e A14
Naturalidade	2	V4 e V9
Desempenho	3	V2, V4 e V9
Desenvolvimento Custoso	4	A14, E3, V4 e V7

Tabela 11 – Quantidade de códigos encontrados para desenvolvimento Nativo

mente das capacidades do celular. Também pela naturalidade, que literalmente é o ponto de comparação com as ferramentas multiplataformas (“esta ferramenta consegue produzir telas tão naturais quanto um aplicativo desenvolvido nativamente?”). No entanto a preocupação com o Desenvolvimento Custoso ainda é muito relevante, e é justamente esta a motivação para a existência de ferramentas multiplataformas.

Código	Qtd	Artefatos
Componentização	1	E4
Problemas de compatibilidade	1	E4
Documentação efetiva	1	A10 e E2
Comunidade Ampla	1	A10 e E2
Não adequado para uso intensivo de gráficos	2	I1 e I8
Poucas Funcionalidades Nativas	2	A32 e I8
Tecnologia difundida	3	A32, E2 e I8

Tabela 12 – Quantidade de códigos encontrados para Phonegap

Partindo para o **PhoneGap** (Tabela 12) que foi a primeira ferramenta multiplataforma a se tornar popular. Ainda há alguns códigos positivos, mas a idade da ferramenta começa a demonstrar seu peso. Problemas de compatibilidade e poucas funcionalidades nativas são grandes inibidores, principalmente quando se precisa utilizar destas funcionalidades.

Já a Tabela 13 é subdividida entre **Web Apps** e **PWA**, já que são tecnologias parecidas, mas ainda diferentes entre si. Começando pelos Web Apps, que são basicamente páginas web comuns com capacidades responsivas para se adaptar a diferentes tamanhos de telas. Como esperado da abordagem Web ainda há desvantagens em seu uso, no entanto é uma ferramenta mais acessível por uso de tecnologias difundidas e por seu desenvolvimento barato. Já os PWAs compensam algumas das desvantagens do desenvolvimento web, como o funcionamento offline, dificuldade de acesso às APIs nativas e acesso exclusivo no navegador. No entanto ainda está em um estágio muito inicial de sua



Código	Qtd	Artefatos
Web Apps		
Acesso Dificultado	1	V4
Desempenho Inferior	1	V4
Naturalidade Inferior	1	V4
Desenvolvimento Barato	1	V4
Necessita de um Navegador	1	V4
Tecnologia Difundida	1	V4
Flexibilidade de Tecnologias	1	V4
PWAs		
Estabilidade Offline	1	E4
Acesso Limitado à APIs Nativas	1	V1
Desempenho Inferior	1	V1
Falta de Maturidade	1	V1
Custo de Desenvolvimento Baixo	1	V1

Tabela 13 – Quantidade de códigos encontrados para Web Apps e PWA

evolução, e mesmo com as melhorias ainda há críticas à sua eficiência.

Código	Qtd	Artefatos
Dificuldade de Instalação	1	E3
Desempenho Prejudicado	1	E3
Conhecimento Reaproveitado de React.js	1	V9
Versão iOS pode exigir trabalho extra	1	E6
Desenvolvimento de UI Facilitado	1	A16, A25 e V9
Testes Facilitados	1	V9
Dificuldade de Aprendizado	1	E3
Dificuldade de utilizar bibliotecas	1	E3
Maturo	2	V1 e V9
Viável a longo prazo	2	I5 e V6
Comunidade Ativa	3	I7, E3 e E6
Fácil acesso à APIs Nativas	3	A16, V1 e V10
Open Source	3	I6, I7 e V1
Desempenho	3	I10, V1 e V9
Facilidade de Desenvolvimento	3	E3, V1 e V9
Componentização	3	E3, E4 e E6
Uso de tecnologias difundidas	7	A16, E3, E6, I7, I10, V19 e V10

Tabela 14 – Quantidade de códigos encontrados para React Native

A ferramenta mais popular atualmente, **React Native** (Tabela 14) obteve poucas críticas quando comparado aos resultados do questionário (Seção 5.2). Talvez isso seja reflexo de uma demográfica diferente da que foi coletada no questionário. Mas independente de críticas, React Native se tornou a ferramenta mais popular por algum motivo. E este motivo possivelmente é sua facilidade de desenvolvimento, que incorpora de tecnologias

difundidas e uma comunidade ativa para introduzir novos desenvolvedores rapidamente.

<b>Código</b>	<b>Qtd</b>	<b>Artefatos</b>
Curva de aprendizado acentuada	1	E4
Baixo Custo	1	E4
Grande quantidade de código compartilhado	1	I6
Custo Alto	1	I6
Curva de Aprendizado	1	I8
Desempenho Inferior	1	V5
Maturo	1	A25 e V1
Suporta mais plataformas	1	I5
Desempenho	2	I6 e V1
Desenvolvimento de UI dificultado	2	A25 e E4
Open Source	2	A34 e V1
Proximo a plataforma nativa	2	A2 e V1
Uso de tecnologias difundidas	3	E4, I7 e I10
Desenvolvimento de UI facilitado	3	A34, E4 e V1
Versão Gratuita Limitada	5	A25, I1, I6, I8 e I10

Tabela 15 – Quantidade de códigos encontrados para Xamarin

Finalmente o **Xamarin** (Tabela 15), a ferramenta multiplataforma desenvolvida pela Microsoft e lançada em 2013. Apresentando características interessantes para empresas grandes, como a proximidade da plataforma nativa e o desempenho, sua principal utilização é justamente para esta demográfica. Mas ao mesmo tempo a Microsoft vem tentando introduzir novas tecnologias para possibilitar seu uso por equipes menores através do Xamarin.Forms.

A Tabela 16 mostra os principais pontos positivos e negativos de cada ferramenta observados durante a codificação. Foram selecionados os três códigos positivos e os três códigos negativos com maior ocorrência de cada ferramenta. Em casos de empate no número de ocorrências foram listados todas com mesmo valor.

<b>Ferramenta</b>	<b>Positivos</b>	<b>Negativos</b>
Flutter	Desempenho, Naturalidade, Viabilidade a longo prazo e Facilidade de Teste e Desenvolvimento	Maturidade
Ionic	Tecnologia Difundida, Desenvolvimento de UI, Viabilidade a Longo Prazo, Curva de Aprendizado e Comunidade	Desempenho, Controle de Status de Rede e de Câmera, Não ser nativo para iOS

MD <sup>2</sup>	Produção em massa, Maior nível de abstração, Manutenibilidade e Reusabilidade, Produtividade e Ajustes Finos	Documentação, Curva de Aprendizado, Instalação
Desenvolvimento Nativo	Desempenho, Naturalidade e Comunidade	Custo de Desenvolvimento
Phonegap	Tecnologia Difundida, Comunidade, Documentação e Componentização	Acesso às APIs Nativas, Não adequado para uso intensivo de gráficos, compatibilidade
PWA	Estabilidade Offline, Custo de Desenvolvimento	Acesso às APIs Nativas, Desempenho e Maturidade
React Native	Tecnologia Difundida, Componentização, Facilidade de Desenvolvimento, Desempenho, Open Source, Acesso às APIs Nativas e Comunidade	Dificuldade de utilizar bibliotecas, de Aprendizado e de Instalação, Desempenho e problemas com iOS
Desenvolvimento Web	Custo de Desenvolvimento, Tecnologia Difundida, Flexibilidade de Tecnologias	Acesso Dificultado, Desempenho, Naturalidade, Necessita de Navegador
Xamarin	Desenvolvimento de UI, Tecnologia Difundida, Próximo a Plataforma Nativa e Open Source	Versão Gratuita Limitada, Desenvolvimento de UI

Tabela 16 – Qualidades e Déficits de cada ferramenta presente na codificação

## 5.4 Pesquisa de Campo

Conforme discutido no Capítulo 3, foi realizada uma pesquisa de campo em uma empresa de agricultura de precisão, cujo objetivo foi atuar no processo de desenvolvimento multiplataforma e obter evidências empíricas dos detalhes e decisões que precisaram ser tomadas neste caso.

A pesquisa de campo compreendeu três processos que tem em comum o objetivo de transformar uma plataforma online em um aplicativo. O primeiro processo foi a trans-

formação desta plataforma para um *Web App*. Isto implicou em mudanças de layout e design para adequar às dimensões de *smartphones* e *tablets* responsivamente. O segundo processo foi a implementação do PWA no *Web App* produzido, exigindo a configuração de uma *Service Worker* e criação de algoritmos para lidar com os dados *offline*. Por fim o último processo foi a criação do aplicativo multiplataforma em React Native baseado na plataforma online. E como a plataforma online já estava em ReactJS foram exploradas a conversibilidade e reuso desta migração. Os três processos são descritos nas próximas subseções.

### 5.4.1 Transformação para *Web App*

A primeira etapa foi elaborar uma forma de alterar a estrutura da plataforma baseado nas dimensões da tela. No momento deste processo esta plataforma ainda era em *jQuery*, um framework JavaScript para manipulação do DOM da página. E então com poucas buscas já foi encontrada uma forma de controlar este parâmetro. Com o conhecimento em mente foi possível começar a atacar diretamente os problemas encontrados.

**Entrada do dia 1:** *“Primeiramente é necessário modificar a principal função do sistema até o momento: o menu de seleção de camadas de sobreposição do mapa. Para isto será necessário refazer o código de construção dinâmica deste componente para refletir ao novo método de exibição baseado em modais (até o momento era apenas um menu aberto ocupando a maior parte do mapa) ... A implementação do modal foi um sucesso, com poucas alterações foi possível reaproveitar um exemplo pronto do site do Bootstrap e adaptá-lo a minha necessidade”*

**Entrada do dia 2:** *“O próximo passo foi refazer a barra lateral do site, que estava fixado na lateral, ocupando um espaço de 50px horizontalmente que em dispositivos móveis é uma quantia considerável. Para tanto implementei um menu “hamburger” para exibir ou não o sidebar. A implementação foi simples, usando apenas lógica e jQuery. Com essas modificações o sistema web já se encontra perto do ideal de um web app, exibindo apenas uma navbar no topo da página, ocupando pouco espaço. Agora é necessário atacar apenas os problemas específicos de cada página”*

**Entrada do dia 3:** *“A última etapa foi as corrigir as páginas relacionadas ao login e outras 3 páginas de cadastro e edição que apresentavam problemas de responsividade. Utilizei de conhecimentos básicos de CSS e jQuery para adaptá-las sem muito esforço. Finalizando a primeira adaptação do sistema em 3 dias, com 6 horas de dedicação por dia.”*

No geral o processo ocorreu sem nenhuma dificuldade, mesmo com pouca experiência prévia do desenvolvedor. O sistema em si não seguia boas práticas nem possuía algum padrão de desenvolvimento para facilitar sua manutenção, o que pode ter feito o

processo levar mais tempo do que seria necessário. E no geral esta primeira adaptação se tornará cada vez menos precisa, devido aos novos frameworks utilizados no momento da escrita desta dissertação já possuírem um alto nível de responsividade integrada aos seus componentes básicos. De qualquer forma, esta experiência comprova que caso necessário a adaptação de um *website* convencional em um *Web App* não deverá ser um esforço demasiadamente grande.

### 5.4.2 Transformação para PWA

Logo após o término do último processo, começou-se o esforço para a transformação para PWA. O principal objetivo era conferir à plataforma atual certo nível de funcionamento offline. Outra qualidade desejada foi a redução de uso de conexão, já que a internet para os usuários típicos do sistema experiencia instabilidades de conexão e mesmo quando estável não é muito rápida.

**Entrada do dia 1:** *“Devido a minha falta de experiência com PWAs, iniciei procurando um bom guia para transformar seu web app em PWA. Rapidamente encontrei um guia que fornecia tudo o que seu Web App precisa para o processo. O autor tem êxito em listar o que é preciso, mas falta explicações de como implementar e o que faz cada parte, portanto fui atrás do guia oficial da Google. Este é um guia muito mais extenso, no entanto, como já sabia o que procurar, o processo foi muito mais rápido. E a implementação poderá começar amanhã após configurar o gulp.”*

**Entrada do dia 4:** *“Tive de modificar o gulp do site, já que ele precisa realizar uma etapa de pre caching, que tornaria os testes do PWA muito mais árduos se não fossem feitos automaticamente. Para isto segui o guia do sw-precache para gulp. Consegui resultados rápidos para configurá-lo, no entanto por problemas de compatibilidade com outras funções do gulp, e por falta de experiência acabei por gastar 3 dias para acertar todos os detalhes.”*

**Entrada do dia 5:** *“Após alguns ajustes finos na Service Worker, consegui executar o PWA em meu smartphone. Ele funciona muito bem, no entanto, percebo alguns problemas mínimos de desempenho que um app nativo não exibiria. Entre estes problemas: demora de transição de página ao clicar em botões (provavelmente não teria este problema se o sistema já estivesse em SPA); e arrastar a página para cima frequentemente causa o aplicativo a recarregar a página, função herdada do navegador.”*

**Entrada do dia 6:** *“Conseguindo configurar o PWA em uma semana com os problemas que tive foi um tempo muito bom. Mas durante testes mais sofisticados hoje notei problemas relacionados a lógica de caching que já vem pré-configuradas com o PWA. Portanto hoje tive de começar a implementar uma forma mais sofisticada lidar com o cache, principalmente sobre o que fazer com os requests que necessitam enviar uma informação*

*para o servidor e os requests que precisam ter um id em sua própria URL”*

**Entrada do dia 16:** *“Depois de duas semanas de várias iterações de desenvolvimento e testes finalmente tive um resultado satisfatório. O aplicativo pode ser utilizado mesmo sem acesso a internet, contanto que já tenha sido acessado pelo menos uma vez. Ele sincroniza informações de POST com a API assim que adquire conexão e envia uma notificação assim que consegue realizá-la. Através da lógica de pre-caching feito na Service Worker também é possível adicionar todos os endpoints da API que necessitam de um id. O único problema restante é com a exibição de camadas de sobreposição no mapa, já que para baixar todas elas demandaria de muito armazenamento no dispositivo. Portanto o aplicativo só exibirá camadas que já foram abertas enquanto havia conexão e portanto estão no cache.”*

Esta etapa durou ao todo três semanas, que pode ser considerado um tempo moderado para um iniciante na tecnologia implementar uma versão tão específica. Os benefícios ganhos com esta implementação foram a funcionalidade offline e melhora de desempenho e também a possibilidade de “instalar” o site como um aplicativo. Esta instalação não ocorre através da loja de aplicativos, mas sim através do próprio navegador ao acessar o site, notificando o usuário desta possibilidade. O aplicativo ainda se comporta um pouco como um site, não demonstrando uma fluidez nativa, mas ao menos é aberto em tela cheia, ganhando um pouco mais de tamanho vertical se comparado ao uso no navegador.

### 5.4.3 Criação do aplicativo multiplataforma

Primeiramente é preciso atualizar o contexto que a plataforma se encontrava durante este processo. A plataforma online foi migrada do jQuery para o ReactJS nos últimos meses. E no período de desenvolvimento do aplicativo, a migração já havia sido terminada e novas funcionalidades já estavam operando. A migração para o aplicativo ainda não envolve o projeto inteiro da plataforma, mas sim apenas algumas das funcionalidades que serão utilizadas por membros da empresa para coletar dados. Mas mesmo com esta mentalidade, o desenvolvimento foi feito com o plano de em breve se tornar uma versão completa igual a plataforma online.

**Entrada do dia 1:** *“Comecei o dia estudando a documentação e se havia alguma diferença sintática ou de arquitetura que deveria saber. Após ganhar um pouco de confiança pude começar a preparar o projeto. Utilizei as recomendações começando o projeto com o Expo e rapidamente consegui criar a base. Comecei a instalar as bibliotecas que precisaria com o npm e tudo funcionou perfeitamente. Estabeleci uma estrutura de arquivos idêntica ao projeto web e já que iria utilizar do Redux, decidi copiar as ações e reducers também.”*

**Entrada do dia 2:** *“Comecei a implementação e já percebi que não poderia reutilizar diretamente as views do projeto, pois a biblioteca que utilizava (Material-UI) não*

*é compatível com React Native. Pelo menos a estrutura hierárquica e lógica de controle podem ser reaproveitados. O projeto web está em processo de conversão para React Hooks, o que significa que até algumas das lógicas precisam ser convertidas para funcionar com High Order Components já que React Native ainda não possui Hooks.”*

**Entrada do dia 3:** *“Agora que terminei de implementar as views preciso começar a conectá-las com a API através do Redux. Felizmente muito do código das actions pode ser reutilizado, e todos os reducers também. Isto vai facilitar imensamente o processo de conexão.”*

**Entrada do dia 7:** *“Em apenas três dias consegui realizar todas as conexões encontrei um problema, uma das bibliotecas que utilizava em uma view não está funcionando na hora de produzir uma build final. Pra corrigir o problema terei de ejetar o projeto. Comecei a ejeção hoje e não estou tendo sucesso, continuarei tentando amanhã, caso não consiga terei de recriar o projeto já ejetado e transferir os arquivos para o outro projeto.”*

**Entrada do dia 18:** *“Como previsto tive de recriar o projeto para conseguir progredir. Demorou mais do que gostaria para obter sucesso, foram 4 dias gastos só para dar continuidade ao projeto. Mas pelo menos consegui produzir uma APK que poderá ser entregue aos usuários internos da empresa que são os principais usuários para o aplicativo. Ao todo o projeto demorou cerca de um mês de trabalho para poder ser disponibilizado, em um total de 18 dias de trabalho, já que foi realizado ao longo de dezembro com alguns feriados e recessos.”*

Começando pelo aplicativo resultante não foi observado impactos do desempenho e fluidez se comparado a um aplicativo nativo. Mas é importante reconhecer que ainda não há funções que poderiam causar grandes impactos do desempenho e que os dispositivos utilizados para os testes não estão na ponta inferior no quesito capacidade de hardware. Não houve dificuldades com acesso a nenhuma API nativa, no entanto só foi utilizado do GPS, Câmera e Acesso aos arquivos armazenados.

Partindo para o segundo tópico estudado, a facilidade de transição do ReactJS para React Native. Houve uma quantidade significativa de código reutilizado que de fato acelerou a produção do aplicativo. No entanto é importante considerar que o desenvolvedor não tinha experiência prévia com React Native. E durante este processo foi notado que muito do código que não foi reutilizado eram apenas problemas de compatibilidade de bibliotecas ou de APIs do React (Hooks e HOC). Portanto fica o questionamento de se isso não poderia ser driblado por um desenvolvedor mais experiente, e também se no futuro este “gap” entre os dois Reacts será reduzido.

Para finalizar, mesmo com o principal critério para a escolha do React Native ser a plataforma Web já estar em ReactJS, ainda houve os critérios que contribuíram ainda mais para esta decisão: era necessário um aplicativo que utiliza bastante as APIs nativas;

com um desempenho aceitável, já que será utilizado em serviço; e com capacidade de operar offline. Esses requisitos combinam bem com a proposta da abordagem generativa, e React Native pertence a esta abordagem.

Utilizar do React Native teve pontos positivos além do reuso de código. Um ponto positivo foi a questão de desempenho, que mesmo utilizando de vários recursos nativos não houve um impacto prejudicial a fluidez do aplicativo. Outro ponto positivo foi a escalabilidade, já que a arquitetura do projeto utiliza de alta componentização e uso de estados e ações que ajudam a conter o caos que projetos de grande complexidade podem adquirir. Já o único ponto negativo foi a questão da ejeção, começar um projeto utilizando do Expo como é recomendado no site do React Native pode causar contratemplos quando o projeto começa a expandir.

As lições aprendidas ficam com a questão de reuso de código do ReactJS, que pode economizar muito tempo de desenvolvimento se já houver um Web App desenvolvido com esta ferramenta. E também ao iniciar um projeto de React Native, já começá-lo sem o Expo, a menos que este seja um aplicativo muito simples.

## 5.5 Análise de correlação entre os diferentes resultados

Nesta seção é realizada uma análise para correlacionar todos os resultados obtidos através dos experimentos realizados durante a pesquisa. Assim como os objetivos do trabalho, esta seção se subdivide em duas partes, os resultados para os critérios e fatores de decisão, e a classificação das ferramentas. O objetivo desta seção é ilustrar de forma fácil de ser comparada todos os resultados obtidos durante a execução da pesquisa.

### 5.5.1 Sobre critérios e fatores de decisão

Na Tabela 17 estão os resultados de todos os experimentos em relação aos critérios. Também há uma coluna dedicada ao trabalho de Ahmad et al. (2018) como forma de comparar os resultados obtidos com os resultados de um trabalho similar já publicado. Para combinar os resultados cada critério é discutido em uma conclusão sobre a relevância do critério.

Acesso à recursos nativos
<b>Proof of Concept:</b> Há diferenças entre as ferramentas
<b>Questionário:</b> 83% de concordância e 6% de discordância
<b>Codificação:</b> Citado em 40,74% das fontes
<b>Pesquisa de Campo:</b> Foi necessário e não deu muito trabalho
<b>Ahmad et al. (2018):</b> 46% de ocorrência na literatura e 24% de concordância nos questionários



<p><b>Conclusão:</b> Os quatro métodos apontam que o critério é relevante, esta relevância é observada na análise da literatura feita por Ahmad, mas não foi observada em seus questionários. No geral o critério pode ser considerado essencial para a escolha de uma ferramenta</p>
<b>Aplicativo resultante</b>
<p><b>Proof of Concept:</b> Há pouca diferença entre as ferramentas  <b>Questionário:</b> 42% de concordância e 29% de discordância  <b>Codificação:</b> Citado em 1,85% das fontes  <b>Pesquisa de Campo:</b> Não foi necessário  <b>Ahmad et al. (2018):</b> Não avaliado  <b>Conclusão:</b> Não foi possível observar consenso de que este é um critério muito importante entre os métodos que o avaliaram</p>
<b>Desenvolvimento de UI</b>
<p><b>Proof of Concept:</b> Há muitas diferenças entre as ferramentas  <b>Questionário:</b> 87% de concordância e 5% de discordância  <b>Codificação:</b> Citado em 35,18% das fontes  <b>Pesquisa de Campo:</b> Foi necessário e não deu muito trabalho  <b>Ahmad et al. (2018):</b> Não foi avaliado  <b>Conclusão:</b> Os quatro métodos apontam que o critério é relevante na hora de escolher uma ferramenta</p>
<b>Distribuição</b>
<p><b>Proof of Concept:</b> Não foi avaliada  <b>Questionário:</b> 82% de concordância e 8% de discordância  <b>Codificação:</b> Citado em 11,11% das fontes  <b>Pesquisa de Campo:</b> Não foi necessário  <b>Ahmad et al. (2018):</b> Não foi avaliado  <b>Conclusão:</b> Dos três métodos que avaliaram este critério, um não o considera necessário, enquanto os outros consideram ele ligeiramente importante. No geral pode-se considerar que é um critério importante, mas que pode ser ignorado caso necessário</p>
<b>Escalabilidade</b>
<p><b>Proof of Concept:</b> Há diferença entre as ferramentas  <b>Questionário:</b> 81% de concordância e 6% de discordância  <b>Codificação:</b> Citado em 7,4% das fontes  <b>Pesquisa de Campo:</b> Foi necessário e não deu muito trabalho  <b>Ahmad et al. (2018):</b> 15% de ocorrência na literatura e 47% de concordância nos questionários  <b>Conclusão:</b> Os quatro métodos apontam sua necessidade, e no trabalho de Ahmad também fica confirmada esta aprovação</p>
<b>Facilidade de desenvolvimento</b>
<p><b>Proof of Concept:</b> Há muita diferença entre as ferramentas  <b>Questionário:</b> 91% de concordância e 1% de discordância  <b>Codificação:</b> Citado em 11,11% das fontes  <b>Pesquisa de Campo:</b> Foi necessário  <b>Ahmad et al. (2018):</b> 14% de ocorrência na literatura e 68% de concordância nos questionários  <b>Conclusão:</b> Unanimamente indicado como um requisito fundamental, inclusive pelo o trabalho de Ahmad</p>
<b>IDE</b>

<p><b><i>Proof of Concept:</i></b> Há pouca diferença entre as ferramentas  <b>Questionário:</b> 60% de concordância e 15% de discordância  <b>Codificação:</b> Citado em 20,37% das fontes  <b>Pesquisa de Campo:</b> Não foi necessário  <b>Ahmad et al. (2018):</b> 14% de ocorrência na literatura e 71% de concordância nos questionários  <b>Conclusão:</b> É um critério bastante polêmico, os questionários demonstraram certa aprovação e é bastante presente na codificação, já os outros dois métodos não indicam esta necessidade. No geral é um critério que pode ser relevante, mas dificilmente será essencial. O trabalho de Ahmad também indica esta polêmica</p>
<b>Licença e custos</b>
<p><b><i>Proof of Concept:</i></b> Não há diferença entre as ferramentas  <b>Questionário:</b> 53% de concordância e 20% de discordância  <b>Codificação:</b> Citado em 1,83% das fontes  <b>Pesquisa de Campo:</b> Foi necessário  <b>Ahmad et al. (2018):</b> Não foi avaliado  <b>Conclusão:</b> Os quatro métodos apontam que o critério pode ser desconsiderado</p>
<b>Linguagem de programação</b>
<p><b><i>Proof of Concept:</i></b> Há diferença entre as ferramentas  <b>Questionário:</b> 81% de concordância e 7% de discordância  <b>Codificação:</b> Citado em 29,62% das fontes  <b>Pesquisa de Campo:</b> Não foi necessário  <b>Ahmad et al. (2018):</b> Não foi avaliado  <b>Conclusão:</b> Resultados um pouco mistos, mas pendendo mais para o lado de ser um critério relevante</p>
<b>Manutenibilidade</b>
<p><b><i>Proof of Concept:</i></b> Há diferença entre as ferramentas  <b>Questionário:</b> 82% de concordância e 1% de discordância  <b>Codificação:</b> Citado em 5,55% das fontes  <b>Pesquisa de Campo:</b> Foi necessário e não deu muito trabalho  <b>Ahmad et al. (2018):</b> 27% de ocorrência na literatura e 74% de concordância nos questionários  <b>Conclusão:</b> Os quatro métodos apontam que o critério é relevante para a escolha da ferramenta, e o estudo de Ahmad também indica o mesmo</p>
<b>Naturalidade</b>
<p><b><i>Proof of Concept:</i></b> Há muitas diferenças entre as ferramentas  <b>Questionário:</b> 70% de concordância e 14% de discordância  <b>Codificação:</b> Citado em 7,4% das fontes  <b>Pesquisa de Campo:</b> Foi necessário e não deu muito trabalho  <b>Ahmad et al. (2018):</b> Não foi avaliado  <b>Conclusão:</b> Dos quatro métodos três indicam explicitamente sua necessidade, enquanto um indica certa polêmica. No geral é um critério relevante</p>
<b>Oportunidade para desenvolvimento futuro</b>
<p><b><i>Proof of Concept:</i></b> Não há diferenças entre as ferramentas  <b>Questionário:</b> 31% de concordância e 36% de discordância  <b>Codificação:</b> Citado em 1,85% das fontes  <b>Pesquisa de Campo:</b> Não foi necessário</p>

<p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Foi o critério com a desaprovação mais unânime, com certeza não é algo que interessa para o desenvolvedor na hora de escolher uma ferramenta</p>
<b>Plataformas suportadas</b>
<p><b>Proof of Concept:</b> Não há diferença entre as ferramentas</p> <p><b>Questionário:</b> 46% de concordância e 34% de discordância</p> <p><b>Codificação:</b> Citado em 27,77% das fontes</p> <p><b>Pesquisa de Campo:</b> Não foi necessário</p> <p><b>Ahmad et al. (2018):</b> 46% de ocorrência na literatura e 24% de concordância nos questionários</p> <p><b>Conclusão:</b> Dos quatro métodos apenas a codificação indica sua relevância, enquanto os outros demonstram que não é algo importante. No geral é um critério que muitas vezes deve ser desconsiderado</p>
<b>Responsividade</b>
<p><b>Proof of Concept:</b> Não há diferença entre as ferramentas</p> <p><b>Questionário:</b> 95% de concordância e 1% de discordância</p> <p><b>Codificação:</b> Citado em 7,4% das fontes</p> <p><b>Pesquisa de Campo:</b> Foi necessário e não deu muito trabalho</p> <p><b>Ahmad et al. (2018):</b> 54% de ocorrência na literatura e 38% de concordância nos questionários</p> <p><b>Conclusão:</b> Dos quatro métodos três apontam sua importância. No geral pode-se considerar como um critério relevante</p>
<b>Tecnologia Difundida</b>
<p><b>Proof of Concept:</b> Há diferença entre as ferramentas</p> <p><b>Questionário:</b> 50% de concordância e 24% de discordância</p> <p><b>Codificação:</b> Citado em 25,92% das fontes</p> <p><b>Pesquisa de Campo:</b> Não foi necessário</p> <p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Um critério muito polêmico, metade dos métodos indicam sua importância, enquanto a outra metade indica que não é tão necessário. No geral é um critério que pode ser descartado, mas que possui seu valor</p>
<b>Velocidade e custo de desenvolvimento</b>
<p><b>Proof of Concept:</b> Há diferença entre as ferramentas</p> <p><b>Questionário:</b> 73% de concordância e 5% de discordância</p> <p><b>Codificação:</b> Citado em 25,92% das fontes</p> <p><b>Pesquisa de Campo:</b> Foi necessário</p> <p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Os quatro métodos indicam sua relevância</p>
<b>Viabilidade de longo prazo</b>
<p><b>Proof of Concept:</b> Há diferença entre as ferramentas</p> <p><b>Questionário:</b> 92% de concordância e 3% de discordância</p> <p><b>Codificação:</b> Citado em 7,4% das fontes</p> <p><b>Pesquisa de Campo:</b> Foi necessário</p> <p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Os quatro métodos sua relevância</p>
<b>Comunidade</b>
<p><b>Proof of Concept:</b> Há muitas diferenças entre as ferramentas</p>

<p><b>Questionário:</b> 85% de concordância e 1% de discordância</p> <p><b>Codificação:</b> Citado em 16,66% das fontes</p> <p><b>Pesquisa de Campo:</b> Foi necessário</p> <p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Os quatro métodos indicam sua relevância</p>
<b>Documentação</b>
<p><b>Proof of Concept:</b> Há muitas diferenças entre as ferramentas</p> <p><b>Questionário:</b> 90% de concordância e 1% de discordância</p> <p><b>Codificação:</b> Citado em 11,11% das fontes</p> <p><b>Pesquisa de Campo:</b> Foi necessário</p> <p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Os quatro métodos indicam sua relevância</p>
<b>Desempenho</b>
<p><b>Proof of Concept:</b> Não há diferença entre as ferramentas</p> <p><b>Questionário:</b> 81% de concordância e 1% de discordância</p> <p><b>Codificação:</b> Citado em 51,85% das fontes</p> <p><b>Pesquisa de Campo:</b> Foi necessário</p> <p><b>Ahmad et al. (2018):</b> Não foi avaliado</p> <p><b>Conclusão:</b> Dos quatro métodos três indicam sua relevância, e na codificação foi o critério com a maior avaliação. No geral é um critério relevante</p>

Tabela 17 – Resumo e conclusão dos critérios

## 5.5.2 Sobre as ferramentas

A Tabela 18 contém todos os resultados de todos os métodos conduzidos para cada ferramenta avaliada. Para combinar os resultados de cada ferramenta em uma conclusão é feita uma discussão em cada ferramenta utilizando de todos os resultados encontrados para ela.

<b>Flutter</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário: Qualidades:</b> Custo, Aplicativo Resultante, Distribuição, Desenvolvimento de UI, Responsividade e Desempenho. <b>Déficits:</b> Migração, Tecnologia Pouco Difundida, APIs Nativas, Facilidade de Desenvolvimento e Não a melhor escolha.</p> <p><b>Codificação: Positivos:</b> Desempenho, Naturalidade, Viabilidade a longo prazo e Facilidade de Teste e Desenvolvimento. <b>Negativos:</b> Maturidade.</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> É uma ferramenta muito completa e com bastante potencial, no entanto está no começo de seu desenvolvimento, sendo imatura e com alguns problemas decorrentes disto, como problemas de integração com APIs Nativas e Facilidade de Desenvolvimento</p>
<b>Ionic</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário: Qualidades:</b> Aplicativo Resultante, Distribuição, Manutenibilidade, Suporte a Plataformas, Documentação, Custo, Linguagem e Facilidade de Desenvolvimento. <b>Déficits:</b> APIs Nativas, Migração, Desempenho, Pouco Recomendada e Não a melhor escolha.</p>

<p><b>Codificação: Positivos:</b> Tecnologia Difundida, Desenvolvimento de UI, Viabilidade a Longo Prazo, Curva de Aprendizado e Comunidade. <b>Negativos:</b> Desempenho, Controle de Status de Rede e de Câmera, Não ser nativo para iOS.</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> Uma ferramenta muito diferente das outras que dominam o mercado atualmente, possuindo algumas vantagens específicas, mas apresentando alguns problemas, principalmente com o uso de APIs Nativas</p>
<b>Kotlin</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário: Qualidades:</b> Melhor Escolha e Recomendada, Aplicativo Resultante, APIs Nativas, Distribuição, Custo, Linguagem, Manutenibilidade, Responsividade e Desempenho. <b>Déficits:</b> Comunidade, Instalação, Desenvolvimento de UI, Naturalidade, Migração e Tecnologia Pouco Difundida.</p> <p><b>Codificação:</b> Não foi avaliado</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> Outra ferramenta muito nova no mercado, ainda começando a construir uma comunidade enquanto resolve problemas de instalação e de desenvolvimento de UI, mas que ainda pode demonstrar seu verdadeiro potencial</p>
<b>MD<sup>2</sup></b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário:</b> Não foi avaliado</p> <p><b>Codificação: Positivos:</b> Produção em massa, Maior nível de abstração, Manutenibilidade e Reusabilidade, Produtividade e Ajustes Finos. <b>Negativos:</b> Documentação, Curva de Aprendizado, Instalação.</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> Uma ferramenta altamente específica que dificilmente será adotada por profissionais da área, primariamente por possuir uma experiência muito diferente do habitual e pelas barreiras de documentação e curva de aprendizado, no entanto a ferramenta possui seu nicho, e é muito boa nele</p>
<b>Native Script</b>
<p><b>Proof of Concept:</b> Opção válida, mas não apresenta vantagens específicas sobre as outras ferramentas</p> <p><b>Questionário:</b> Não foi avaliado</p> <p><b>Codificação:</b> Não foi avaliado</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> No geral é uma ferramenta generativa que não possui muito destaque comparada a outras da mesma abordagem</p>
<b>Phonegap</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário:</b> Não foi avaliado</p> <p><b>Codificação: Positivos:</b> Tecnologia Difundida, Comunidade, Documentação e Componentização. <b>Negativos:</b> Acesso às APIs Nativas, Não adequado para uso intensivo de gráficos, compatibilidade.</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> Uma ferramenta sólida e amplamente utilizada ao longo dos anos, no entanto não envelheceu tão bem quanto outras, apresentando reclamações constantes com APIs nativas e compatibilidade</p>

<b>PWA</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário:</b> Não foi avaliado</p> <p><b>Codificação: Positivos:</b> Estabilidade Offline, Custo de Desenvolvimento. <b>Negativos:</b> Acesso às APIs Nativas, Desempenho e Maturidade.</p> <p><b>Pesquisa de Campo:</b> Facilidade de conversão</p> <p><b>Conclusão:</b> Uma tecnologia nova mas que tem potencial para revolucionar o desenvolvimento web, mas que ainda está no início de sua jornada, sendo imatura e apresentando problemas com APIs nativas e desempenho</p>
<b>Qt</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário: Qualidades:</b> Linguagem, Responsividade, Desempenho, Recomendação, Aplicativo Resultante e Viabilidade a longo prazo. <b>Déficits:</b> Migração, Naturalidade, APIs Nativas, Instalação e Custo.</p> <p><b>Codificação:</b> Não foi avaliado</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> Uma ferramenta altamente especializada com o objetivo de ser uma forma de aumentar o alcance de plataformas e ainda oferecer a possibilidade de otimizar o desempenho em um baixo nível, e esta capacidade vem com seus custos como problemas de naturalidade, acesso as APIs nativas e instalação</p>
<b>React Native</b>
<p><b>Proof of Concept:</b> Melhor naturalidade e de desenvolvimento menos custoso</p> <p><b>Questionário: Qualidades:</b> Tecnologia Difundida, Custo, Suporte a Plataformas, Distribuição, Responsividade e Facilidade de Desenvolvimento. <b>Déficits:</b> Migração, APIs Nativas, Não a melhor escolha, Desempenho e Documentação.</p> <p><b>Codificação: Positivos:</b> Tecnologia Difundida, Componentização, Facilidade de Desenvolvimento, Desempenho, Open Source, Acesso às APIs Nativas e Comunidade. <b>Negativos:</b> Dificuldade de utilizar bibliotecas, de Aprendizado e de Instalação, Desempenho e problemas com iOS.</p> <p><b>Pesquisa de Campo:</b> Desempenho, Facilidade de Desenvolvimento e Escalabilidade</p> <p><b>Conclusão:</b> A ferramenta mais utilizada no momento da escrita desta dissertação, sendo genericamente a mais poderosa, mesmo que apresente seus negativos. Os resultados apresentam certa incoerência sobre seus prós e contras, mas no geral é diferenciada por seu desempenho, componentização, responsividade e facilidade de desenvolvimento</p>
<b>Desenvolvimento Web</b>
<p><b>Proof of Concept:</b> Não foi avaliado</p> <p><b>Questionário: Qualidades:</b> Suporte a Plataformas, Desenvolvimento de UI, Linguagem, Documentação, Aplicativo Resultante, Custo e Instalação. <b>Déficits:</b> Não a melhor escolha, Desempenho, Migração, APIs Nativas e Naturalidade.</p> <p><b>Codificação: Positivos:</b> Custo de Desenvolvimento, Tecnologia Difundida, Flexibilidade de Tecnologias. <b>Negativos:</b> Acesso Dificultado, Desempenho, Naturalidade, Necessita de Navegador.</p> <p><b>Pesquisa de Campo:</b> Desempenho Prejudicado, mas com facilidade e velocidade de desenvolvimento</p>

<p><b>Conclusão:</b> O desenvolvimento web sempre será uma das formas mais acessíveis de disponibilizar um aplicativo para diferentes plataformas, mas também apresenta problemas por depender de uma conexão, no geral possui um dos desenvolvimentos menos custosos e permite grande flexibilidade de formas de desenvolvimento</p>
<b>Weex</b>
<p><b>Proof of Concept:</b> Opção válida, mas não apresenta vantagens específicas sobre as outras ferramentas</p> <p><b>Questionário:</b> Não foi avaliado</p> <p><b>Codificação:</b> Não foi avaliado</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> No geral é uma ferramenta generativa que não possui muito destaque comparada a outras da mesma abordagem</p>
<b>Xamarin</b>
<p><b>Proof of Concept:</b> Boa opção caso haja necessidade por alta customização das funcionalidades</p> <p><b>Questionário: Qualidades:</b> Suporte a Plataformas, Distribuição, Linguagem, Manutenibilidade e Tecnologia Difundida. <b>Déficits:</b> Migração, Não a melhor escolha, Desenvolvimento de UI, Aplicativo Resultante e Instalação.</p> <p><b>Codificação: Positivos:</b> Desenvolvimento de UI, Tecnologia Difundida, Próximo a Plataforma Nativa e Open Source. <b>Negativos:</b> Versão Gratuita Limitada, Desenvolvimento de UI.</p> <p><b>Pesquisa de Campo:</b> Não foi avaliado</p> <p><b>Conclusão:</b> Uma ferramenta eficiente e com um nicho parecido ao Qt, sendo possível realizar certas otimizações de desempenho, mas ao mesmo tempo sendo menos estrita, garantindo um desenvolvimento menos custoso, mas ainda não chega a velocidade de desenvolvimento do React Native e Ionic</p>

Tabela 18 – Resumo e conclusão das ferramentas





## 6 Conclusão

Este capítulo trata sobre a finalização do trabalho, condensando os resultados e discutindo sobre a pesquisa, e também autoavaliando as metodologias e resultados obtidos.

### 6.1 Contribuições

As principais contribuições obtidas com esta pesquisa foram:

- Foram identificados e analisados os principais critérios para os desenvolvedores, quando adotando uma solução multiplataforma. Essa identificação foi feita a partir de múltiplos métodos, considerando múltiplos pontos de vista, inclusive de profissionais, algo que em geral é ignorado na literatura acadêmica; e
- Foi feita uma análise objetiva de diferentes ferramentas, com base nos critérios identificados, e considerando diferentes pontos de vista, tanto prático com base na *Proof of Concept*, como de opiniões de profissionais. Aqui também o trabalho difere da literatura, que normalmente foca em informações da documentação e do fabricante e em opiniões/experiência subjetiva do pesquisador.

### 6.2 Limitações

As limitações identificadas nesta pesquisa foram:

- Não foram incluídas todas as ferramentas, pois há muitas ferramentas, e muitas das ferramentas se tornam obsoletas com o tempo. O maior exemplo disto é o Phoneygap, que era uma das ferramentas mais utilizadas no período de 2011-2014 (RIBEIRO; Da Silva, 2012; PALMIERI; SINGH; CICHETTI, 2012; RAJ; TOLETY, 2012) e agora é discutida na literatura e entre os profissionais. E mesmo entre apenas as atuais e populares ainda há um grande número que não foi incluído, e isto ocorreu por falta de participantes nos questionários e entrevistas que utilizem desta ferramenta e por falta de discussão na literatura;
- Não fizemos uma análise comparativa aprofundada entre academia X indústria. Isto não ocorreu principalmente por limitações de tempo, devido o escopo do trabalho ter crescido além do previsto. Isto ocorreu por conta do progresso do projeto acontecer de forma muito reativa; e

- Não entramos no mérito de diferentes contextos para os critérios. Sabemos que, dependendo do contexto, um critério A pode ser mais importante que o critério B. Mas no geral, tentamos listar os desdobramentos de detalhes importantes, a ponto de o leitor conseguir tomar sua própria decisão. (ou seja, não fizemos um modelo de decisão detalhado, que era a ideia inicial).

## 6.3 Ameaças à validade

Os pontos aonde ocorrem ameaças à validade são nos experimentos conduzidos. Portanto nesta seção serão agrupados e apresentados os problemas que cada experimento pode ter sido afetado.

### 6.3.1 *Proof of Concept*

As ameaças à validade encontradas na *Proof of Concept* são:

- Exemplo pequeno, pode ter causado um excesso de simplificação;
- Não testamos todas as ferramentas;
- Análise foi feita por um pesquisador só; e
- Foi feita durante o aprendizado, talvez um programador experiente saiba contornar os pontos identificados.

### 6.3.2 Questionário

As ameaças à validade encontradas no questionário são:

- Boa abrangência, diversos países, diversos perfis, mas poderia ter mais;
- O local de divulgação dos questionários pode afetar a visão dos participantes. Por exemplo, um fórum pode ter desenvolvido uma visão muito pessimista sobre a ferramenta, criando um viés particular para esta; e
- Poucas respostas para algumas ferramentas, podemos ter tido uma visão enviesada.

### 6.3.3 Codificação

As ameaças à validade encontradas na codificação são:

- Entrevistas foram difíceis de arranjar. Acabamos pegando diversos perfis, mas no geral com baixa experiência. O ideal seria profissionais com mais anos de prática.

Pode ter influenciado as questões de tarefas mais avançadas e possibilidades que esses profissionais ainda não vivenciaram;

- Codificação é subjetiva. Fizemos, para 6% das fontes uma codificação em duas pessoas, mas poderia ser melhor (100%);
- Não há como garantir que todos os artigos científicos sobre o tema tenham sido encontrados;
- O processo de encontrar fontes na literatura cinza é complicado. Tentamos escolher apenas fontes com alta credibilidade, e alguns critérios, mas há sempre o risco de material enviesado;
- Poucas fontes da literatura cinza, ainda que diversificada e algumas fontes importantes.

#### 6.3.4 Pesquisa de campo

As ameaças à validade encontradas na pesquisa de campo são:

- Processo aplicado pelo próprio autor desta pesquisa, portanto há um certo viés, uma tendência em observar melhor alguns critérios e descartar outros;
- Aplicação restrita a uma única tecnologia, portanto conseguimos algumas lições aprendidas sobre React Native X Web e PWA, mas pouca base de comparação com outras;
- Coleta de dados subjetiva (diário de bordo).

## 6.4 Trabalhos Futuros

Outro subproduto desta pesquisa é a possibilidade de expansão do tema, seja complementando o que foi realizado nesta, reproduzindo e confirmando os métodos ou até iniciando novas investigações. Desta forma as possibilidades sugeridas são:

- **Reprodução dos métodos para uma escala maior de contribuidores:** Como já mencionado, aumentar o número de entrevistados contribuirá com um aumento de confiança nos dados obtidos. E um trabalho deste poderia ocorrer com nenhuma ou poucas alterações na metodologia aplicada em escalas muito menores de tempo de dedicação;

- **Novas comparações entre ferramentas:** Com os critérios propostos e exemplo de utilização destes na Seção 18 novos pesquisadores podem utilizá-los para construir novas comparações entre ferramentas. Sejam por ferramentas que não foram avaliadas ou até mesmo por ferramentas que não existiam durante a elaboração deste trabalho.
- **Aprimoramento do mecanismo de comparação:** Outra possibilidade seria uma nova forma de realizar as comparações entre as ferramentas. Por mais intuitivo que seja o uso de tabelas comparativas, há muita ineficiência neste modelo. Talvez elaborar um mecanismo de decisão automatizado para sugerir a melhor escolha para um desenvolvedor baseado em entradas deste sobre o que precisa no projeto seria uma excelente forma. Isto poderia ser alcançado com simples algoritmos em uma série de perguntas com escolhas pré-definidas, ou até mesmo com uma inteligência artificial extraindo o peso para os critérios através da leitura de um parágrafo.

## 6.5 Considerações Finais

Desenvolvimento Multiplataforma é um tópico muito recente e volátil, mas que vem demonstrando sua importância conforme as ferramentas evoluem. Portanto é vital para esta evolução que as ferramentas possam ser avaliadas de acordo com as necessidades dos principais interessados.

No geral ferramentas de desenvolvimento multiplataforma contribuem positivamente na produtividade de equipes de desenvolvimento, principalmente as pequenas. Mas há muito mais nuances que nem sempre são exploradas nas comparações realizadas na academia e na indústria.

Um diferencial deste trabalho em relação à maioria dos que são publicados, na academia e na indústria, é a forma **duradoura, flexível e reutilizável para estabelecer as métricas de comparação de ferramentas de desenvolvimento multiplataforma.**

## Referências

- ABID, F. B. A.; KARIM, A. N. Cross-platform development for an online food delivery application. *Proceedings of the IEEE International Conference on Computing, Networking and Informatics, ICCNI 2017*, v. 2017-January, p. 1–4, 2017. Citado 2 vezes nas páginas 58 e 85.
- AGGARWAL, R. S. *React Native Vs Xamarin. What to choose for cross-platform app development?* 2019. Disponível em: <<https://codeburst.io/react-native-vs-xamarin-what-to-choose-for-cross-platform-app-development-226ac2ef35c9>>. Citado na página 86.
- AHMAD, A. et al. Challenges of mobile applications development: Initial results. In: IEEE. *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. [S.l.], 2017. p. 464–469. Citado na página 85.
- AHMAD, A. et al. An empirical study of investigating mobile applications development challenges. *IEEE Access*, IEEE, v. 6, p. 17711–17728, 2018. Citado 10 vezes nas páginas 52, 57, 58, 59, 84, 102, 103, 104, 105 e 106.
- 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*. [S.l.: s.n.], 2016. p. 41–48. Citado na página 84.
- ALTEXSOFT. *Xamarin vs React Native vs Ionic vs NativeScript: Cross-platform Mobile Frameworks Comparison*. 2018. Disponível em: <<https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-vs-nativescript-cross-platform-mobile-frameworks-comparison/>>. Citado na página 86.
- BALDWIN, C. *Cross Platform Mobile Development Tools: Ending the iOS vs. Android Debate*. 2019. Disponível em: <<http://thinkapps.com/blog/development/develop-for-ios-v-android-cross-platform-tools/>>. Citado na página 86.
- BERNARDES, T. F.; MIYAKE, M. Y. Cross-platform mobile development approaches: A systematic review. *IEEE Latin America Transactions*, IEEE, v. 14, n. 4, p. 1892–1898, 2016. Citado 10 vezes nas páginas 23, 29, 30, 31, 32, 33, 34, 35, 52 e 85.
- BIORN-HANSEN, A.; GRONLI, T.-M.; GHINEA, G. A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 5, p. 1–34, 2018. Citado 2 vezes nas páginas 59 e 84.
- BIORN-HANSEN, A.; MAJCHRZAK, T. A.; GRONLI, T.-M. Progressive web apps for the unified development of mobile applications. In: SPRINGER. *International Conference on Web Information Systems and Technologies*. [S.l.], 2017. p. 64–86. Citado 3 vezes nas páginas 58, 59 e 85.

- BOTELLA, F.; ESCRIBANO, P.; PEÑALVER, A. Selecting the best mobile framework for developing web and hybrid mobile apps. *ACM International Conference Proceeding Series*, 2016. Citado 2 vezes nas páginas 52 e 86.
- BRUCKER, A. D.; HERZBERG, M. On the static analysis of hybrid mobile apps. In: SPRINGER. *International Symposium on Engineering Secure Software and Systems*. [S.l.], 2016. p. 72–88. Citado na página 85.
- BUSCH, M. *Mobile development in 2019: native versus cross-platform*. 2019. Disponível em: <<https://www.youtube.com/watch?v=45JWJfss9GA>>. Citado na página 87.
- CABALLERO, J.; BODDEN, E.; ATHANASOPOULOS, E. Engineering secure software and systems: 8th International Symposium, ESSoS 2016 London, UK, April 6–8, 2016 proceedings. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9639, p. 72–88, 2016. ISSN 16113349. Citado na página 52.
- CHARKAOUI, S.; ADRAOUI, Z.; Habib Benlahmar, E. Cross-platform mobile development approaches. *Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in*, p. 188–191, 2014. ISSN 23271884. Citado 2 vezes nas páginas 23 e 85.
- CHARMAZ, K. *Constructing grounded theory*. [S.l.]: Sage, 2014. Citado 3 vezes nas páginas 40, 41 e 44.
- CHOUDHARY, H. *Hybrid App vs native app*. 2017. Disponível em: <<https://www.youtube.com/watch?v=jE5W89GslX8>>. Citado na página 86.
- CODEDAMN. *Ionic 4 v/s React Native - A detailed analysis of both*. 2018. Disponível em: <<https://www.youtube.com/watch?v=bKVt0rGIx9s>>. Citado na página 86.
- CORBALAN, L. et al. Development frameworks for mobile devices: A comparative study about energy consumption. *Proceedings - International Conference on Software Engineering*, p. 191–201, 2018. ISSN 02705257. Citado 2 vezes nas páginas 58 e 85.
- CRUXLAB. *Xamarin vs Ionic vs React Native: differences under the hood*. 2017. Disponível em: <<https://cruxlab.com/blog/reactnative-vs-xamarin/>>. Citado na página 86.
- DALMASSO, I. et al. Survey, comparison and evaluation of cross platform mobile application development tools. *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, n. July, p. 323–328, 2013. ISSN 2376-6492. Citado 9 vezes nas páginas 15, 23, 34, 52, 53, 61, 69, 70 e 86.
- DHILLON, S.; MAHMOUD, Q. H. An evaluation framework for cross-platform mobile application development tools. *Software: Practice and Experience*, Wiley Online Library, v. 45, n. 10, p. 1331–1357, 2015. Citado na página 84.
- DOGTIEV, A. *Cross Platform Mobile App Development Guide (2019)*. 2019. Disponível em: <<https://www.businessofapps.com/guide/cross-platform-mobile-app-development/>>. Citado na página 86.

- DYVLIASH, V. *React Native vs Xamarin vs Ionic*. 2019. Disponível em: <<https://belitsoft.com/react-native-development/react-native-vs-xamarin-vs-ionic>>. Citado na página 86.
- EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. *Ain Shams Engineering Journal*, Elsevier, v. 8, n. 2, p. 163–190, 2017. Citado 2 vezes nas páginas 59 e 86.
- FAN, W.; YANG, J. Design and implementation of cross-platform friends-positioning mobile APP based on phonegap and HTML5. *2017 2nd IEEE International Conference on Computational Intelligence and Applications, ICCIA 2017*, v. 2017-January, p. 239–242, 2017. Citado 2 vezes nas páginas 58 e 85.
- FERREIRA, C. M. et al. An evaluation of cross-platform frameworks for multimedia mobile applications development. *IEEE Latin America Transactions*, v. 16, n. 4, p. 1206–1212, 2018. ISSN 15480992. Citado 3 vezes nas páginas 52, 57 e 84.
- FLAHERTY, K. Diary studies: Understanding long-term user behavior and experiences. *Retrieved January*, v. 7, p. 2018, 2016. Citado na página 41.
- GAOUAR, L.; BENAMAR, A.; BENDIMERAD, F. T. Model driven approaches to cross platform mobile development. In: ACM. *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*. [S.l.], 2015. p. 19. Citado 2 vezes nas páginas 36 e 85.
- GAUNT, M. *Service Workers: an Introduction*. Google, 2018. Disponível em: <<https://developers.google.com/web/fundamentals/primers/service-workers/?hl=en-us>>. Citado na página 31.
- HALLIDAY, P. *What Mobile Development Framework Should I Learn in 2017?* 2017. Disponível em: <<https://www.youtube.com/watch?v=D20o8rg-p-4>>. Citado na página 87.
- HEITKOTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. In: SPRINGER. *International Conference on Web Information Systems and Technologies*. [S.l.], 2012. p. 120–138. Citado 6 vezes nas páginas 23, 29, 52, 53, 69 e 70.
- HUDLI, A.; HUDLI, S.; HUDLI, R. An evaluation framework for selection of mobile app development platform. In: *Proceedings of the 3rd International Workshop on Mobile Development Lifecycle*. [S.l.: s.n.], 2015. p. 13–16. Citado na página 84.
- JIA, X.; EBONE, A.; TAN, Y. A performance evaluation of cross-platform mobile application development approaches. *Proceedings - International Conference on Software Engineering*, ACM, p. 92–93, 2018. ISSN 02705257. Citado 3 vezes nas páginas 58, 84 e 90.
- KOZIOKAS, P. T.; TSELIKAS, N. D.; TSELIKIS, G. S. Usability testing of mobile applications: Web vs. Hybrid Apps. *ACM International Conference Proceeding Series*, Part F132523, p. 9–10, 2017. Citado 2 vezes nas páginas 58 e 86.
- KS, A. *Top 10 Cross-Platform Mobile Development Tools*. 2018. Disponível em: <<https://www.hongkiat.com/blog/cross-mobile-platform-framework-wora/>>. Citado na página 86.

- LATIF, M. et al. Cross platform approach for mobile application development: A survey. *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, p. 1–5, 2016. Citado 8 vezes nas páginas 23, 29, 36, 52, 54, 57, 61 e 85.
- LUCRÉDIO, D. Uma abordagem orientada a modelos para reutilização de software. *INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO UNIVERSIDADE DE SÃO PAULO*, p. 37, 2009. Citado 2 vezes nas páginas 13 e 36.
- MAJCHRZAK, T. A.; BIORN-HANSEN, A.; GRONLI, T.-M. Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks. *Proceedings of the 50th Hawaii International Conference on System Sciences*, p. 6162–6171, 2017. Citado 3 vezes nas páginas 23, 52 e 85.
- MAJCHRZAK, T. A.; BIORN-HANSEN, A.; GRONLI, T.-M. Progressive Web Apps: the Definite Approach to Cross-Platform Development? *Proceedings of the 51st Hawaii International Conference on System Sciences*, p. 5735–5744, 2018. Citado na página 85.
- MALAVOLTA, I. Beyond Native Apps: Web Technologies to the Rescue. *Mobile! 2016 - Proceedings of the 1st International Workshop on Mobile Development, co-located with SPLASH 2016*, p. 1–2, 2016. Citado 2 vezes nas páginas 58 e 84.
- MEDIA, T. *Mobile Apps - Web vs. Native vs. Hybrid*. 2017. Disponível em: <<https://www.youtube.com/watch?v=ZikVtdopsfY>>. Citado na página 87.
- MEIRELLES, P. et al. A students' perspective of native and cross-platform approaches for mobile application development. In: SPRINGER. *International Conference on Computational Science and Its Applications*. [S.l.], 2019. p. 586–601. Citado 3 vezes nas páginas 52, 57 e 84.
- MERCADO, I. T.; MUNAIAH, N.; MENEELY, A. The impact of cross-platform development approaches for mobile applications from the user's perspective. In: ACM. *Proceedings of the International Workshop on App Market Analytics*. [S.l.], 2016. p. 43–49. Citado na página 23.
- MISCHOOK, S. *The Future of Mobile Development*. 2019. Disponível em: <[https://www.youtube.com/watch?v=Nh0F\\_okJAKA](https://www.youtube.com/watch?v=Nh0F_okJAKA)>. Citado na página 87.
- MOUMNI, H. *Flutter, Ionic, React Native or Xamarin? What you use and why?* 2018. Disponível em: <<https://dev.to/heithemmoumni/what-is-the-best-flutter-ionic-or-react-native-503d>>. Citado na página 86.
- NUNKESSER, R. Beyond web-native-hybrid: a new taxonomy for mobile app development. In: IEEE. *2018 IEEE-ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. [S.l.], 2018. p. 214–218. Citado 2 vezes nas páginas 59 e 85.
- PALMIERI, M.; SINGH, I.; CICCHETTI, A. Comparison of cross-platform mobile development tools. *2012 16th International Conference on Intelligence in Next Generation Networks, ICIN 2012*, p. 179–186, 2012. ISSN 00010782. Citado 8 vezes nas páginas 23, 52, 54, 61, 69, 70, 85 e 111.



- PARANHOS, R. et al. Uma introdução aos métodos mistos. *Sociologias*, SciELO Brasil, v. 18, n. 42, p. 384–411, 2016. Citado 2 vezes nas páginas 27 e 50.
- PAWAR, A. P.; JAGTAP, V. S.; BHAMARE, M. S. Survey on Techniques for Cross Platform Mobile Application Development. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, v. 3, n. 10, p. 3551–3558, 2014. Citado 3 vezes nas páginas 52, 55 e 86.
- PINTO, C. M.; COUTINHO, C. From Native to Cross-platform Hybrid Development. *9th International Conference on Intelligent Systems 2018: Theory, Research and Innovation in Applications, IS 2018 - Proceedings*, p. 669–676, 2018. Citado 2 vezes nas páginas 52 e 85.
- RAJ, C. R.; TOLETY, S. B. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. *2012 Annual IEEE India Conference, INDICON 2012*, p. 625–629, 2012. ISSN 0098-5589. Citado 12 vezes nas páginas 13, 21, 23, 29, 30, 32, 33, 35, 52, 55, 84 e 111.
- REDDA, Y. A. Cross platform Mobile Applications Development Mobile Apps Mobility. n. June, p. 1–77, 2012. Citado na página 85.
- RIBEIRO, A.; Da Silva, A. R. Survey on cross-platforms and languages for mobile apps. *Proceedings - 2012 8th International Conference on the Quality of Information and Communications Technology, QUATIC 2012*, n. September, p. 255–260, 2012. ISSN 00010782. Citado 6 vezes nas páginas 23, 52, 56, 57, 86 e 111.
- RICHARD, J. *9 Best Cross-Platform Mobile App Development Tools*. 2017. Disponível em: <<https://dzone.com/articles/9-best-cross-platform-mobile-app-development-tools-1>>. Citado na página 86.
- RICHTER, F. *The Smartphone Duopoly*. Statista, 2018. Disponível em: <<https://www.statista.com/chart/3268/smartphone-os-market-share/>>. Citado na página 21.
- RIEGER, C.; MAJCHRZAK, T. A. Weighted evaluation framework for cross-platform app development approaches. In: SPRINGER. *EuroSymposium on Systems Analysis and Design*. [S.l.], 2016. p. 18–39. Citado na página 86.
- SOLUTIONS, E. L. *Top 10 Mobile Frameworks For App Development | Mobile App Development Frameworks*. 2017. Disponível em: <<https://www.youtube.com/watch?v=zhM4XxD6hE>>. Citado na página 87.
- SOLUTIONS, E. L. *The Mobile Frameworks Battle | Flutter Vs React Native Vs Native App Development*. 2018. Disponível em: <<https://www.youtube.com/watch?v=cR8sHoyKRNA>>. Citado na página 87.
- SOUZA, C. R. B. d.; STEINMACHER, I.; PRIKLADNICKI, R. *IX Brazilian Conference on Software*. 2018. Citado na página 41.
- STARTUPS, M. for. *Choosing the best mobile app framework*. 2019. Disponível em: <<https://www.youtube.com/watch?v=4m7msadL5iA>>. Citado na página 86.

STATISTA; APPLE. *Number of available apps in the Apple App Store from July 2008 to January 2017*. AppleInsider, 2017. Disponível em: <<https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/>>. Citado 2 vezes nas páginas 13 e 22.

STATISTA; EMARKETER. *Number of smartphone users worldwide 2014-2020*. 2016. Disponível em: <<https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>>. Citado 2 vezes nas páginas 13 e 22.

STATISTA et al. *Number of available applications in the Google Play Store from December 2009 to December 2017*. AppBrain, 2017. Disponível em: <<https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>>. Citado 2 vezes nas páginas 13 e 22.

STATISTA; IDC. *Smartphone OS market share forecast 2014-2022 | Statistic*. IDC, 2018. Disponível em: <<https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/>>. Citado 2 vezes nas páginas 13 e 21.

TECHLEAD. *React Native vs Flutter vs WebView - Hybrid Mobile App Development for 2018*. 2018. Disponível em: <<https://www.youtube.com/watch?v=-n5G48o2bxQ>>. Citado na página 87.

THOMAS, C. *Cross Platform Mobile Development in 2018 A Beginner's Guide*. 2018. Disponível em: <<https://trifinlabs.com/cross-platform-mobile-development/>>. Citado na página 86.

TUFAIL, H. et al. Model-driven development of mobile applications: A systematic literature review. In: IEEE. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. [S.l.], 2018. p. 1165–1171. Citado 2 vezes nas páginas 52 e 85.

VILCEK, T.; JAKOPEC, T. Comparative analysis of tools for development of native and hybrid mobile applications. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017 - Proceedings*, p. 1516–1521, 2017. Citado 2 vezes nas páginas 52 e 85.

VISHAL, K.; KUSHWAHA, A. S. Mobile Application Development Research Based on Xamarin Platform. *Proceedings - 4th International Conference on Computing Sciences, ICCS 2018*, IEEE, p. 115–118, 2019. Citado 2 vezes nas páginas 52 e 85.

WANG, J. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, ACM, v. 29, n. 5, p. 36–46, 1999. Citado 2 vezes nas páginas 31 e 56.

XANTHOPOULOS, S.; XINO GALOS, S. A comparative analysis of cross-platform development approaches for mobile applications. In: ACM. *Proceedings of the 6th Balkan Conference in Informatics*. [S.l.], 2013. p. 213–220. Citado 6 vezes nas páginas 29, 36, 52, 56, 61 e 84.