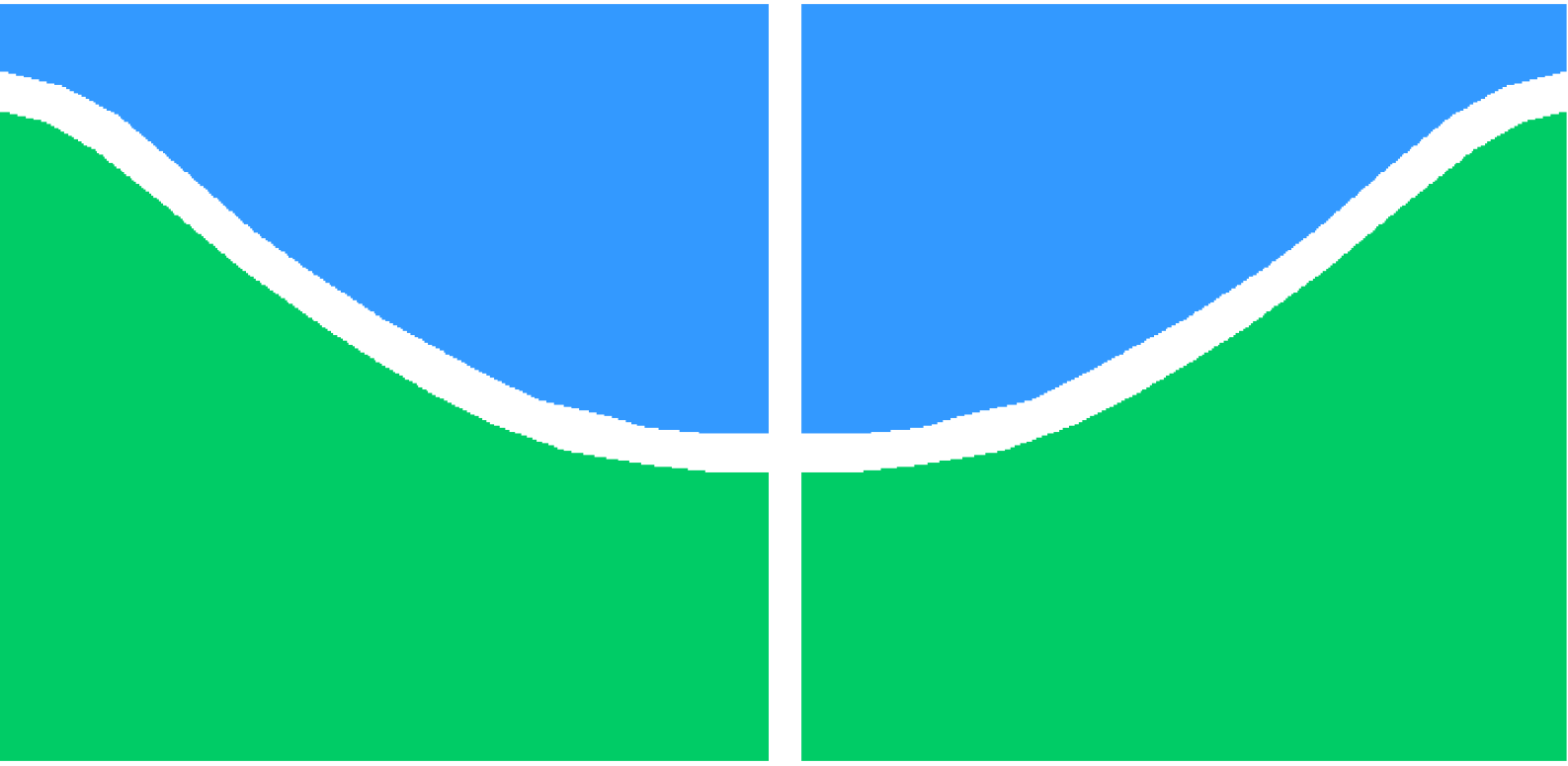

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Proposta De Guia Comparativo para Desenvolvimento de Aplicativos Móveis

Autor: Felipe César Silveira de Assis
Orientador: Prof. Dr. Paulo Meirelles

Brasília, DF
2018

Felipe César Silveira de Assis

Proposta De Guia Comparativo para Desenvolvimento de Aplicativos Móveis

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Paulo Meirelles

Coorientador: Profa. Dra. Carla Rocha

Brasília, DF

2018

Felipe César Silveira de Assis
Proposta De Guia Comparativo para Desenvolvimento de Aplicativos Móveis/ Felipe César Silveira de Assis. – Brasília, DF, 2018-
62 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Paulo Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Mobile. 2. Nativo. I. Prof. Dr. Paulo Meirelles. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Proposta De Guia Comparativo para Desenvolvimento de Aplicativos Móveis

CDU 02:141:005.6

Felipe César Silveira de Assis

Proposta De Guia Comparativo para Desenvolvimento de Aplicativos Móveis

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 04 de julho de 2018:

Prof. Dr. Paulo Meirelles
Orientador

Profa. Dra. Carla Rocha
Convidado 1

Rodrigo Siqueira
Convidado 2

Brasília, DF
2018

Este trabalho é dedicado a todos aqueles que acreditam que com trabalho duro podem superar seus demônios e seus deuses, mudando assim a sua própria história.

“I am thou and thou art I“
(Persona 4)

Resumo

O desenvolvimento de aplicativos e softwares para o ambiente mobile é uma área de conhecimento e atuação que tem crescido nos últimos anos. Para a realização deste desenvolvimento é necessário a escolha de uma abordagem de desenvolvimento para o projeto em questão, esta abordagem pode ser nativa (implementação individual em cada plataforma) ou multiplataforma (com grande ou total reutilização de código entre diferentes plataformas). Afim de prover uma visão mais clara sobre qual é a abordagem mais adequada para um projeto de desenvolvimento de software mobile, este trabalho de conclusão de curso foi elaborado, tendo como objetivo fornecer um guia comparativo para a escolha da melhor abordagem para um determinado contexto. Um diferencial para este estudo é que, além dos aspectos quantitativos e objetivos, também foram considerados aspectos qualitativos ou subjetivos os quais influenciam no desenvolvimento do software. Para elaborar este guia comparativo foram levantados dados através de três questionários aplicados em diferentes perfis de desenvolvedores e um exemplo de uso que busca evidenciar a necessidade da aplicação do guia comparativo a um caso experimental. Por fim foi elaborado o guia comparativo, no qual identificamos que os aspectos quantitativos e objetivos não podem ser exclusivos para a escolha da abordagem, fato que também foi observado na aplicação do guia no exemplo de uso proposto. Em geral, ao contrário do que as literaturas que consideram apenas aspectos quantitativos apontam, os desenvolvedores tendem para as abordagens nativas.

Palavras-chaves: desenvolvimento mobile. desenvolvimento multiplataforma. desenvolvimento nativo. desenvolvimento híbrido.

Abstract

The development mobile applications and softwares for mobile environment is an area that has grown in the past years. To develop a mobile application, it is necessary to choose a development approach to be used in a project, this approach can be native (realized for each platform separately) or multiplatform (recognized with common code among different platforms). This thesis was elaborated in order to provide a better understanding on what is the most appropriate approach for a mobile software development project, aiming to provide a guideline for choosing the best approach for a given context. A differential for this study is that, in addition to the quantitative aspects, qualitative or subjective aspects that influence the software development were also considered. In order to elaborate this guideline, data we collected data through three questionnaires, answered by different developer profiles, and an experiment that seeks to evidence the need of the guideline to a project. Finally, the guideline was elaborated, in which we identified that the quantitative aspects cannot be the only ones when selecting an approach. We also observed during the application of the guideline in the experiment, that in general developers tend towards native approaches, in counterpart to what other papers that consider exclusively quantitative aspects point out.

Key-words: mobile development. cross-platform development. native development. hibrid development.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Fig.	Area of the i^{th} component
app(s)	Aplicativo(s) *mobile*.
IBGE	Instituto Brasileiro de Geografia e Estatística *mobile*.

Sumário

1 Introdução

Atualmente a presença de smartphones e outros dispositivos *mobile* é mais significativa em comparação à ultima década. Segundo o IBGE 95% do acesso a internet do brasil é feito através de smartphones, este é apenas um exemplo da influência que o os Apps podem exercer no cotidiano e na própria economia empresarial quando a mesma está imersa neste contexto[1]. Observando com uma perspectiva global, a Tabela 1 mostra 80.7% dos dispositivos utilizam o sistema operacional Android, 17.7% utilizam o sistema iOS e apenas uma parcela inferior a 2% utilizam demais sistemas operacionais. Tais estatísticas são refletidas também no desenvolvimento de software mobile, pois, cerca de 23% dos desenvolvedores são desenvolvedores mobile, dos quais 64% trabalham com o sistema Android, 57% trabalham com iOS.[2].

Podemos observar que ambas as plataformas Android e iOS são alvos fundamentais para se obter uma maior abrangência de público ao se desenvolver um software para mobile, independente de sua categoria.

Frequentemente um app possui como um de seus requisitos a abrangência mínima sobre os dois sistemas operacionais mais utilizados atualmente, iOS e Android, tornando assim evidente a necessidade de se desenvolver para ambas as plataformas. Para se desenvolver para ambas as plataformas, muitas vezes dois produtos são implementados separadamente (com as mesmas *features* e estórias de usuário) e em linguagens diferentes(BERNARDES; MIYAKE, 2016), quando um aplicativo é desenvolvido diretamente para uma plataforma especifica utilizando as linguagens e Frameworks que acessam diretamente os recursos disponíveis pela mesma, nomeia-se **desenvolvimento nativo**(EL-KASSAS et al., 2015).

Partindo da premissa anteriormente apresentada, a reutilização de recursos durante

[1] Fonte:<https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2101543>
[2] Fonte:<Fonte:https://insights.stackoverflow.com/survey/2017>

Tabela 1 – Fração de mercado Mobile

Operating Sistem	**4Q15Units**	**4Q15 Share (%)**	**4Q14Units**	**4Q14 Share (%)**
Android	325,394.4	80.7	279,057.5	76.0
iOS	71,525.9	17.7	74,831.7	20.4
Windows	4,395.0	1.1	10,424.5	2.8
Blackberry	906.9	0.2	1,733.9	0.5
Others	887.3	0.2	1,286.9	0.4
Total	403,109.4	100.0	367,334.4	100.0

o desenvolvimento de um app se torna mais necessária, visto que, mesmo sendo desenvolvidos em linguagens diferentes (Java e Swift) e para plataformas diferentes (iOS e Android, respectivamente), ambos os aplicativos compartilham de *features* iguais, e para se desenvolver um aplicativo para múltiplas plataformas, mais recursos serão necessários considerando que em alto nível é gerado uma alta duplicação de código(MARTINEZ; LECOMTE, 2017).

Para solucionar este problemas existe a abordagem de desenvolvimento multiplataforma (cross platform), a qual, diferentemente da abordagem nativa, tem como principal objetivo a máxima reutilização e centralização do código entre diferentes plataformas(BERNARDES; MIYAKE, 2016); a mesma abordagem pode ser dividida entre **desenvolvimento híbrido e desenvolvimento WebApp**. Apesar de suas vantagens, estas abordagem multiplataforma também possuem desvantagens como "acesso limitado a recursos nativos"e "maior consumo de RAM"(LATIF et al., 2016).

Podemos observar atualmente a presença significativa das tecnologias multiplataforma na industria de desenvolvimento de jogos eletrônicos, categoria de software a qual, o lançamento em diversas plataformas é fundamental para a abrangência do software. Em 2016 aproximadamente 59% de todos os jogos mobile foram desenvolvidos utilizando Frameworks de compilação multiplataforma. Frameworks e bibliotecas como Unity e Unreal, facilitam o reuso de código e a compilação não apenas entre diferentes plataformas mobile, mas também para consoles, navegadores de internet, Windows, MacOS, Linux e smart TVs[3].

Atualmente o desenvolvimento de aplicativos para mais de uma plataforma é um requisito fundamental para uma abrangência considerável ao publico alvo.Abordagens nativas e multiplataforma possuem vantagens e desvantagens entre si, e é necessário ponderar entre as mesmas antes de selecionar qual abordagem é a mais adequada. Entre diferentes contextos, tecnologias e requisitos, pode ser difícil determinar qual é a melhor abordagem para um dado projeto.

A literatura estudada considera a abordagem multiplataforma como a ideal, porém não existe solução universal para projetos mobile. Como observado anteriormente, cada uma das três abordagens apresentadas, possui suas vantagens e desvantagens, algumas diretamente mensuráveis como RAM, espaço em disco, consumo de bateria, tempo de resposta) e algumas não diretamente mensuráveis como requisitos do projeto, conformidade com UI/UX, curva de aprendizado da implementação, documentação e suporte oferecido pela tecnologia. Essas características, dados os diferentes requisitos de aplicativos e de projetos de desenvolvimento, devem ser consideradas antes de se escolher qual abordagem seguir.

[3] Fonte: <https://unity3d.com/public-relations>

Buscando apresentar uma ferramenta adicional para facilitar a escolha da abordagem para um projeto, foi elaborado um guideline o qual, apresenta as características técnicas de cada abordagem, tornando assim possível a comparação entre as mesmas em relação aos requisitos técnicos e não técnicos do aplicativo e do projeto de desenvolvimento. O guideline também faz um aparado e considerações finais sobre cada abordagem, já direcionando para qual tipo de projeto uma determinada tecnologia pode se adequar melhor.

Estudos anteriores possuem comparações entre aspectos qualitativos e quantitativos ao se adotar uma determinada abordagem (GAOUAR ABDELKRIM BENAMAR, 2015)(LATIF et al., 2016); entretanto, o estudo busca contribuir para uma comparação entre as abordagens nativa e multiplataforma (multiplataforma) de duas perspectivas diferentes, perspectiva técnica e não técnica, resultando no guideline mencionado.

O guideline é um resultado de um estudo que utilizou de três questionários, cada questionário foi direcionado a um publico alvo diferente: estudantes de graduação (diferentes níveis de experiencia em desenvolvimento), estudantes mestrandos (maior experiência em desenvolvimento) e profissionais da comunidade de desenvolvimento (diferentes níveis de experiencia, porém aplicados profissionalmente). O objetivo deste estudo, e também deste trabalho de conclusão de curso (TCC), é responder a questão: *"O que deve ser considerado para escolher qual a melhor abordagem (nativa ou multiplataforma) ao se desenvolver um aplicativo mobile?"*

Como mencionado anteriormente o guideline foi elaborado com base nos resultados obtidos a partir dos questionários; considerando que parte da orientação fornecida pelo guideline vem de opiniões empíricas e não técnicas dos desenvolvedores, o mesmo ainda pode ser atualizado, corrigido ou iterado. Buscando uma validação inicial das propostas descritas no guideline, também é proposto neste TCC, um exemplo de uso para o guideline. Este exemplo consiste na elaboração de um aplicativo utilizando ambas as tecnologias nativa e híbridas separadamente, observando suas diferenças técnicas (através de análise de benchmarking) e não técnicas, através de análises e entrevistas informais sobre o processo de desenvolvimento. Através deste exemplo de uso foi possível enfatizar algumas afirmações e orientações propostas pelos questionários anteriores e consequentemente o guideline.

O restante deste trabalho está organizado em: Capitulo 2, onde apresentaremos a fundamentação teórica correspondente às tecnologias de desenvolvimento mobile atuais, tal como a revisão da literatura utilizada para o estudo; Capítulo 3, onde descrevemos a fundo os questionários utilizados na elaboração do guideline e seu contexto experimental; Capítulo 4, onde apresentamos analisamos os resultados extraídos dos questionários; Capítulo 5, discorremos sobre as considerações geradas a partir dos questionários e experimentos aplicados gerando assim o guia comparativo; Capítulo 6, evidenciamos

a necessidade da aplicação do guia comparativo a um caso experimental, também onde discorremos sobre os resultados obtidos a partir do caso experimental e sua consistência perante as afirmações do guia comparativo; Capítulo 6.3, onde faremos as considerações finais considerando o trabalho como um todo e também sugestões de trabalhos futuros.

2 Fundamentação Teórica

Aplicativos para dispositivos moveis possuem um desenvolvimento com certas particularidades, isso ocorre devido ao seus requisitos de hardware, ciclos de *releases* e distribuição dos apps. Desenvolvedores de softwares para dispositivos moveis tem que considerar questões como capacidade de armazenamento, especificações de hardware exclusivas de um determinado aparelho, UI/UX, segurança e privacidade(EL-KASSAS et al., 2015). Também, aplicativos moveis tendem a ser dinamicamente desenvolvidas com pequenos ciclos de iterações para suas atualizações, o seu produto final em geral possui escopo restrito e baixo custo para o usuário final(CORRAL; JANES; REMENCIUS, 2012). Antes de escolher entre abordagem nativa ou multiplataforma, os desenvolvedores devem considerar suas respectivas vantagens e desvantagens(MARTINEZ; LECOMTE, 2017).

2.1 Desenvolvimento Mobile Nativo

Aplicativos nativos são caracterizados por serem desenvolvidos em linguagem padrão da plataforma em questão, em geral cada sistema operacional possui sua própria linguagem de programação, como podemos observar na Tabela 2, em outras palavras estes aplicativos não podem ser portados diretamente para outras plataformas pois será necessário uma nova implementação, maior tempo de desenvolvimento, orçamento e conhecimentos específicos em cada plataforma (HEITKÖTTER; HANSCHKE; MAJCHRZAK, 2013). Utilizando desta linguagem os aplicativos nativos utilizam frameworks que fazem acesso **direto** a chamadas de sistema e recursos específicos do aparelho no qual se encontra em operação, tais como sensor de luminosidade, sensor de proximidade, acelerômetro, giroscópio, magnetômetro bússola, câmeras, GPS (*Global Positioning System*), contatos e email do usuário(EL-KASSAS et al., 2015).

Tabela 2 – Plataformas mobile e suas tecnologias.

OS	**Language**	**Environment**	**App Store**
Apple's iOS	Objective-C/Swift	iOS SDK	Apple-iTunes
Blackberry	C++	BlackBerry Native SDK	BlackBerry World
Google's, Android	JAVA	Android SDK	Play Store
Microsoft's Windows Phone	C++, C, C#	Windows Phone SDK	Windows Phone Store

O desenvolvimento nativo tem acesso direto também aos componentes de Interface

de usuário (UI/UX) fornecidos pelos frameworks nativos proporcionando assim maior fidelidade com as diretrizes the design (design guidelines) da plataforma em questão [1] [2], como representado na Figura 1, proporcionando assim também melhor performance para seus aplicativos.

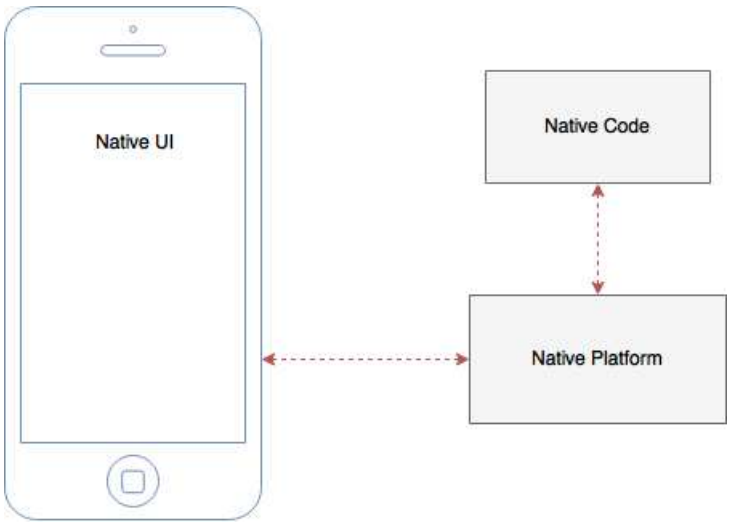

Figura 1 – Representação do acesso de tecnologias nativas.

Ao se acessar diretamente os frameworks do sistema operacional, algumas vantagens se tornam evidentes, podemos destacar a capacidade dos aplicativos de atingirem alta performance (menor consumo de bateria, memória RAM, recursos de rede celular, espaço em disco)(MALAVOLTA, 2016). Aplicativos Nativos são normalmente distribuídos para o público alvo através das respectivas lojas de aplicativos de cada plataforma (e.g. App Store, Play Store), podendo ser baixados e armazenados diretamente no dispositivo(HEITKÖTTER; HANSCHKE; MAJCHRZAK, 2013).

Como citado anteriormente uma consequência de se desenvolver um aplicativo de forma nativa é a necessidade de se implementar novamente a mesma solução em outra linguagem que acesse os frameworks nativos do aparelho, isso torna irrefutável o fato de que se aumenta os custos de produção e possivelmente o tempo de produção de um aplicativo[3](MARTINEZ; LECOMTE, 2017).

2.2 Desenvolvimento Mobile Web App

Umas das soluções multiplataforma para combater a duplicação de código e diminuir o tempo e custo de desenvolvimento de um aplicativo foram os WebApps. Web Apps são aplicativos desenvolvidos utilizando tecnologias web (HTML, CSS, Javascript, etc.). A

[1] Apple Design Guideline:<https://developer.apple.com/design/human-interface-guidelines/>
[2] Android Design Guideline:<https://developer.android.com/design/>
[3] Fonte:<https://www.comentum.com/mobile-app-development-cost.html>

primeira diferença entre os WebApps e apps nativos é o fato de que os WebApps não ficam armazenados diretamente no aparelho em que está sendo utilizados sua principal característica é a necessidade de estar armazenado em um servidor externo, acessado através de um navegador de internet (e.g. Chrome, Safari, Firefox) seguindo o modelo cliente-servidor representado na Figura 2.Por sua vez WebApps tem a vantagem de proporcionar as mesmas features, UI/UX entre as plataformas que o acessarem(CIMAN; GAGGI, 2017).

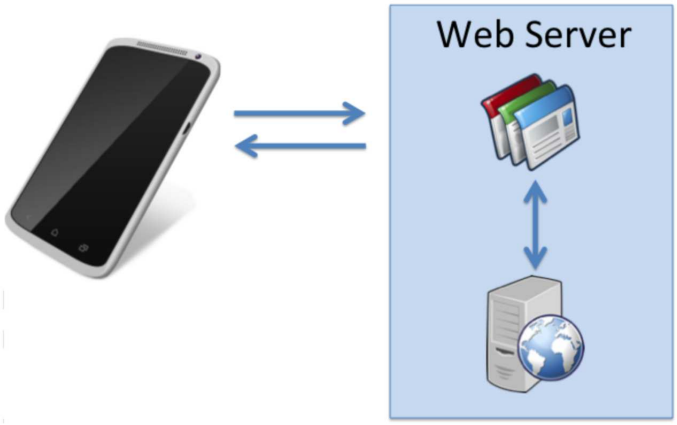

Figura 2 – Representação do acesso de tecnologias WebApp.

Em troca de sua "compatibilidade universal", WebApps possuem acesso limitado, ou impossibilitado, a recursos nativos dos dispositivos (notificações push, GPS, contatos, magnetômetro)(CORRAL; JANES; REMENCIUS, 2012) e por sua vez também consomem mais recursos de rede e bateria dos dispositivos(AHTI; HYRYNSALMI; NEVALAINEN, 2016).

2.3 Desenvolvimento Mobile Híbrido

Buscando mitigar os pontos negativos da abordagem nativa e também da abordagem de WebApps surgiram os aplicativos híbridos; podemos definir os aplicativos híbridos vulgarmente como o meio termo entre O desenvolvimento nativo e o desenvolvimento de WebApp. Ao contrário dos aplicativos nativos, os híbridos podem ser compilados com grande compartilhamento de código para diferentes plataformas mobile, por sua maioria utilizam de tecnologias web para serem implementados (HTML, CSS, javascript), porém ao contrário dos WebApps os híbridos podem ser armazenados localmente no dispositivo, não necessitando de conexão com internet e browser intermediário, utilizando do framework WebKit para seu funcionamento(P.K; M.KANNAN, 2017), como representado na Figura 3.

Os aplicativos híbridos utilizam de componentes web (renderizados pelo WebKit), para simular componentes nativos e proporcionar experiencia similar a experiencia dos

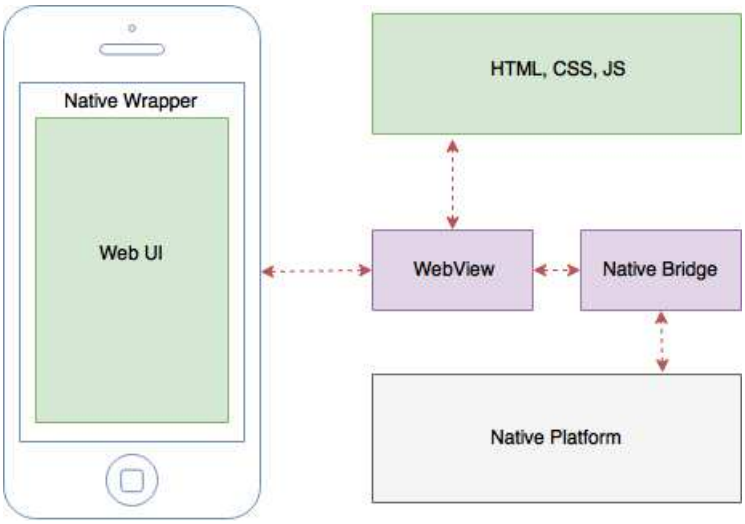

Figura 3 – Representação do acesso de tecnologias híbridas.

aplicativos nativos(JOHNSON et al., 2006). Outra melhoria notável em relação aos WebApps é o acesso aos recursos nativos do aparelho, embora a dificuldade de manipulação dos recursos seja maior, frameworks híbridos mais atuais utilizam de plugins para obter tais acessos. Embora pareçam ideais os frameworks híbridos possuem algumas limitações em troca de sua versatilidade; considerando que estes frameworks e seus plugins muitas vezes fazem acesso **indireto** aos recursos, provocando assim queda de performance geral e gráfica (menor fluidez das animações(P.K; M.KANNAN, 2017), maior tempo de resposta para telas de toque), maior consumo de RAM e maior consumo de bateria chegando a 250% em comparação com os aplicativos nativos(AHTI; HYRYNSALMI; NEVALAINEN, 2016).

2.4 Revisão da Literatura

Nós examinamos trabalhos de desenvolvimento mobile relacionados, aplicando um estratégia de mapeamento de estudo. Selecionamos artigos da IEEE, ACM, e da Science Direct, utilizando termos de pesquisa baseados em palavras-chave de trabalhos correlatos estudados preliminarmente: *cross-platform, development, framework, mobile, native, multiplatform, hybrid. Nós utilizamos os seguintes termos de pesquisa: "cross-platform AND mobile AND development", "hybrid AND mobile AND framework", "multiplatform AND mobile AND framework", "multiplatform AND mobile AND native"*. Em resumo, adotamos o seguinte protocolo:

1. Pesquisar artigos, utilizando os termos de busca definidos;

2. Filtrar de maneira automatizada, excluindo artigos duplicados;

3. Filtrar manualmente, eliminando patentes, duplicações, o que não for artigo e artigos não relacionados. Nós consideramos um artigo não relacionado aqueles que não forem relacionados ao nosso estudo (*mobile*, *cross-platform*, *hybrid development*);

4. Ler o abstracta e novamente remover artigos não relacionados ou irrelevantes ao nosso trabalho;

5. Avaliar os artigos restantes por relevância.

Foi coletado um total de 655 artigos na primeira parte do protocolo. Após aplicar a exclusão automática de artigos duplicados, um total de 620 trabalhos sobraram. O alto número de artigos na pesquisa inicial se deve às palavras-chave *"platform" e "hybrid"*. Nota-se que essas palavras-chave são amplamente utilizadas nos campos de física e química. Manter essas palavras-chave apesar de sua grande abrangência é crucial, dado que estão intimamente relacionadas aos artigos desejados para nosso trabalho. No terceiro passo, nós diminuímos os resultados ao aplicar critérios objetivos (o que não é artigo, patentes, duplicações, fonte inapropriada). Também eliminamos artigos baseado no seu título, onde excluímos aqueles que não eram relacionados a este estudo. Um total de 42 artigos sobraram. Então, nós avaliamos os 42 artigos remanescente através de seu abstracta e os classificamos de um a três, mensurando sua relevância para nossa pesquisa. Por fim, terminamos com o seguinte conjunto: 20 artigos avaliados com três, 5 artigos avaliados com dois, e 7 artigos avaliados com um.

Decidimos atribuir os artigos restantes uma categoria de Trabalho Legado (Seção 2.4.1) e Trabalho Recente (Seção 2.4.2). A seção de trabalho legado aborda artigos antigos (2006 a 2014) com ideias relevantes, mas que estão defasados do ponto de vista tecnológico. A seção de trabalho recente, por outro lado, tem uma coleção de trabalhos (de 2015) onde os estudos utilizam tecnologias recentes. Analisamos trabalhos legado que ainda eram relevante para este pesquisa mas que focavam nos dados e tecnologias dos últimos três anos como base de suas pesquisas. Não incluímos artigos que abordavam técnicas e novas tecnologias que não são bem empregadas pelo mercado. Um exemplo destas tecnologias é o Progressive Web Apps (PWA), um tipo especial de mobile web app; um servidor remoto provê acesso via HTTPS ao PWA, então o usuário pode optar por instalar o aplicativo no dispositivo, e portanto promovendo-o a um aplicativo parecido com o nativo (MALAVOLTA, 2016).

2.4.1 Trabalho Legado

Observamos nos últimos 12 anos uma ampla discussão, pesquisa e desenvolvimento de novas abordagens para aprimorar o desenvolvimento mobile (JOHNSON et al., 2006). Buscando por alternativas que pudessem reduzir os custos e o tempo de desenvolvimento

com o mínimo impacto em outros aspectos como performance do aplicativo, UI/UX, e segurança. Alguns desses estudos têm objetivos e metodologias similares a nossa. Dada a importância de trabalhos anteriores, nós comparamos as abordagens adotadas por outros pesquisadores com a nossa. Charland (CHARLAND; LEROUX, 2011) apresenta uma comparação teórica entre desenvolvimento multiplataforma (*cross-platform*) e nativo em aspectos como programação de UI, performance e conformidade com as convenções. Depois da investigação, o autor conclui que soluções híbridas têm várias vantagens sobre as nativas quando lidando com desenvolvimento multiplataforma. Seguindo a mesma motivação, também consideramos fatores não quantitativos em nossa pesquisa, em particular, empírica (requisitos de projeto, por exemplo).

Prezotto (PREZOTTO; BONIATI, 2014) faz uma distinção entre aplicativos nativos, web-apps e aplicativos híbridos similar a que estamos utilizando neste artigo. O autor fez uma prova de conceito ao programar um aplicativo híbrido. Após a implementação da solução, Prezotto notou comportamentos distintos entre as plataformas que executou a aplicação. Notamos que há uma preocupação em verificar a capacidade de aplicativos híbridos em relação a um abordagem nativa. Em nosso questionários, focamos em destacar essas diferenças para os desenvolvedores e analizar suas decisões.

Dalmasso (DALMASSO et al., 2013) apresentou várias ferramentas de desenvolvimento de aplicativos híbridos disponíveis no mercado (ex.: PhoneGap, Titanium, Rhomobile, JQuery Mobile) e fez comparações através de um questionário. O trabalho de Dalmasso abordava um estudo qualitativo desses frameworks para apresentar um breve panorama geral para desenvolvedores mobile. Nossa pesquisa pretende cobrir não apenas as abordagens multiplataforma, mas também adicionar uma comparação mais empírica em relação a abordagem nativa.

2.4.2 Trabalho Recente Relacionado

Majchrzak (MAJCHRZAK; GRØNLI, 2017) discute sobre como escolher entre diferentes tecnologias híbridas. O autor adotou o Ionic, React e Fuse como objetos de seu estudo, considerando aspectos estritamente técnicos e em maior parte quantitativos. Em nossa pesquisa, nos também discutimos questões não técnicas para determinar a abordagem ideal para um projeto. Nós também queremos prover diretrizes de decisão para escolher uma abordagem mobile. Não consideramos tecnologias individualmente para essa decisão, mas sim a abordagem tecnológica (nativo, web-app, ou híbrido). Como já mencionamos, diferentemente do autor, também consideraremos fatores práticos que influenciam o desenvolvimento de software ao escolher uma das abordagens.

Latif (LATIF et al., 2016) fez um estudo de caso sobre como várias tecnologias híbridas funcionam (linguagem, arquitetura e performance) e destacou suas vantagens e desafios. Gaouar (GAOUAR ABDELKRIM BENAMAR, 2015) conduziu um trabalho

parecido, onde o autor mapeou os requisitos técnicos para multiplataforma em seis categorias: plataformas de suporte, documentação, segurança, acesso a recursos nativos, UI/UX, e consumo de recursos. Diferente destes autores, não focamos em mapear as características de cada abordagem ou tecnologia, mas em utilizar esses tópicos já destacados e mapeados para prover, da perspectiva dos desenvolvedores, uma abordagem recomendada (nativa, multiplataforma, ou web-app) para um determinado caso de uso ou projeto. Como visto na seção IV, o objetivo não está relacionado apenas aos SDKs multiplataforma, mas também comparamos o desenvolvimento multiplataforma com o desenvolvimento nativo de cada sistema operacional.

Há artigos recentes que fizeram uma análise comparativa depois de implementar o mesmo caso de uso com diferentes tecnologias. Carlstrom (AXELSSON; CARLSTRöM, 2016) conduziu um estudo mais próximo ao nosso objetivo: um experimento prático que separasse os resultados de cada estudo de caso pré-definido. Em geral, o tempo gasto no desenvolvimento de aplicações multiplataforma eram menores quando comparados separadamente ao desenvolvimento nativo. Shan Jiang (JIANG, 2016) considerou aspectos técnicos, empíricos e práticos das diferentes soluções. Em resumo, o autor concluiu que não há uma solução universal, dado que não há nenhuma diferença gritante entre processos de desenvolvimento e produtos para decidir uma abordagem definitiva para o desenvolvimento de aplicativos mobile. É possível notar a intercessão com a nossa pesquisa no Capítulo 3, n qual antes de responder ao questionário, os desenvolvedores implementaram a mesma solução em duas tecnologias distintas, nativo (Android) e híbrido (Ionic). De uma maneira distinta dos trabalhos de Carlstrom e Jiang, estamos focando nas características não quantitativas do ponto de vista do desenvolvedor.

3 Metodologia

Nosso objetivo é investigar as abordagens nativa, híbrida e web app em relação ao desenvolvimento de aplicativos mobile. O guia comparativo (guideline) o qual será testado no Capítulo 6 deste trabalho foi elaborado anteriormente utilizando de dados provenientes da revisão bibliográfica e da análise sobre o que as pesquisas mais recentes apontavam em relação a escolha entre desenvolvimento nativo ou multiplataforma.

Em geral a literatura e artigos estudados apontavam para o uso de aplicativos híbridos como o mais vantajosos para aplicativos quando comparado com as outras abordagens(LATIF et al., 2017), porém, também se faz necessário se considerar ambos os aspectos não técnicos e qualitativos de cada abordagem. Nos direcionamos este estudo para responder a seguinte pergunta:

Questão 1: *O que deve ser levado em consideração ao escolher a melhor abordagem (nativa, híbrida ou WebApp) para se desenvolver um aplicativo mobile?*

Para nos ajudar a responder a Questão 1, as seguinte questão de pesquisa surgiu para investigar mais a fundo o contexto quando está relacionado a aspectos qualitativos:

Questão 1.1: *Quais são as vantagens técnicas das abordagens e frameworks multiplataforma (híbridas) da perspectiva de desenvolvedores mobile menos experientes? E quais as mesmas vantagens para os desenvolvedores especialistas?*

Nossa hipótese para a pergunta anterior é que desenvolvedores de aplicativos mobile com menor experiencia destacariam "custo por plataforma"como a maior vantagem técnica entre as plataformas, ou seja, sua escolha tende às tecnologias híbridas.

Em contraste com a hipótese anterior, esperamos que os especialistas não corroborem em totalidade com as afirmações feitas. Criamos essas expectativas devido a outros fatores técnicos como o consumo de RAM, tamanho do arquivo binário gerado pela compilação, capacidade de otimização de código, nível de consumo de bateria, entre outros. Ainda assim se faz necessário analisar outros fatores práticos, e isso nos levou à segunda questão de pesquisa.

Questão 1.2: *Quais são os fatores não técnicos que influenciam na escolha da abordagem e/ou framework utilizados para se desenvolver uma aplicação mobile?*

Nossa hipótese é que vários aspectos subjetivos, tais como "requisitos do projeto", "curva de aprendizado"e "conhecimento prévio na tecnologia", não são apenas que devem influenciar mas também podem ser determinantes para a escolha da abordagem. Nossa expectativa é de que os desenvolvedores determinem os aspectos não técnicos como mais impactantes do que os aspectos técnicos durante a tomada de decisão entre as abordagens.

Foram elaborados três questionários para coletar dados que buscam responde as três questões de pesquisa. A partir dos dados coletados e analisados destes questionários, nos elaboramos o guia comparativo (guideline) buscando facilitar a melhor decisão ao se escolher qual abordagem é a mais adequada para um determinado desenvolvimento de software mobile.

Para se obter mais informações sobre estas características não mensuráveis e aspectos empíricos, foi elaborado a estratégia ilustrada na Figura 4.

Figura 4 – Representação das etapas da pesquisa.

Foi realizado um primeiro experimento com alunos mestrandos da Universidade de São Paulo (USP), neste experimento os alunos tiveram de desenvolver uma aplicação em duas tecnologias diferentes, primeiramente Android (nativo) e em seguida Ionic (híbrido) [1].

[1] Fonte:<https://ionicframework.com/>

3.1 Primeiro Questionário - Percepção dos desenvolvedores ao implementar utilizando de uma abordagem nativa e uma multiplataforma

O primeiro questionário coletou a percepção de estudantes avançados de graduação e pós-graduação em Ciência da Computação da Universidade de São Paulo (USP). Anteriormente a aplicação do questionário os alunos desenvolveram funcionalidades básicas e simples, através de uma disciplina de 4 meses de duração, utilizando abordagens nativas(Anroid) e híbridas (Ionic).

Esta pesquisa tem o objetivo de avaliar as principais vantagens e dificuldades enfrentadas pelos desenvolvedores, os quais consideramos de nível intermediário (em relação a sua experiência de desenvolvimento), ao implementar um aplicativo mobile em ambas as abordagens, e se as dificuldades são percebidas de forma diferente em abordagens híbridas e nativas. Ainda nesta pesquisa, poderíamos avaliar a curva de aprendizagem de em ambas as tecnologias, e consequentemente, responder a Questão 1.1.

Neste experimento não consideramos os aspectos de implementação relacionados á Interface de usuário (animações de UI/UX, quadros por segundo) entre as abordagens. As funcionalidades não envolviam o uso de nenhum recurso de animação ou recurso específico do aparelho (GPS, acelerômetro), apenas funcionalidades as quais consideramos simples, por exemplo, entrada de texto, requisições web (GET, POST, PUT, DELETE) e apresentação de informações.

As seções do questionário são:

- **Seção 1 - Perfil dos participantes:** idade, sexo, conhecimento prévio de linguagens de programação (como C, C ++, Python, PHP e Ruby). Questão exemplo: "Você já já desenvolveu algum software que está sendo usado por outros usuários?;

- **Seção 2 - Experiência prévia em desenvolvimento para mobile:** nível de afinidade com linguagens de programação e tecnologias como Java, Android (Java para Android), Android Studio, HTML, Javascript, Angular 2 e Ionic Questão Exemplo: "Quantas outras plataformas de desenvolvimento você já usou?;

- **Seções 3 and 4 - Percepções sobre o Android Studio e Ionic:** dificuldade durante a configuração do ambiente de desenvolvimento, utilização de Web Services para implementar o login, desenvolvimento da funcionalidade de cadastro de estudante, implementação de Comunicações Bluetooth e Wi-Fi, captura de imagens utilizando a câmera, leitura de QR-Code e upload de arquivos usando o Web Services.

Todas as respostas foram baseadas na escala de Likert com os seguintes níveis: (1) Discordo totalmente; (2) Discordo; (3) Indiferente; (4) Concordo; (5) Concordo totalmente. Finalmente, o aluno pode responder duas perguntas livres para escrever comentários a respeito do desenvolvimento usando o Android Studio e usando o Ionic.

3.2 Segundo e Terceiro Questionários - Percepção dos desenvolvedores sobre os fatores que interferem a escolha da tecnologia mobile

Os dois questionários a seguir pretendiam responder as questões 1 e 1.2. Em busca destas respostas realizamos pesquisas em dois contextos distintos. Primeiramente, queríamos entender os aspectos que os desenvolvedores com menor experiência, ou totalmente inexperientes no desenvolvimento de dispositivos móveis, levam em consideração em seus primeiros projetos para dispositivos móveis. Em segundo lugar, queríamos entender como o mesmo conjunto de fatores afeta os desenvolvedores avançados e até mesmo especialistas em dispositivos móveis.

O segundo questionário coletou a perspectiva dos alunos de graduação do curso de Engenharia de Software da Universidade de Brasília. Coletamos dados de alunos que se participaram da disciplina de projeto de software, onde os alunos devem escolher as tecnologias que vão usar (sem restrições) e justificar sua escolha. Diferentemente, dos alunos avaliados na primeira pesquisa, esses alunos desenvolveram uma aplicação móvel inteira com apenas a abordagem selecionada (nativo ou multiplataforma).

O terceiro questionário coletou o ponto de vista de desenvolvedores mobile experientes. Esta versão foi direcionada á comunidade de desenvolvedores mobile no geral nos canais de comunicação:

- Grupo de desenvolvedores de iOS no Brasil: <https://github.com/iOSDevBR>
- Grupo de desenvolvedores de Android no Brasil: <https://groups.google.com/forum/#!forum/androidbrasil-dev>
- Grupo de desenvolvedores de Android no Brasil: <https://groups.google.com/forum/#!forum/android-brasil\T1\textendashprojetos>
- Grupo de desenvolvedores de Android no Brasil:<http://slack.androiddevbr.org>
- Grupo internacional de desenvolvedores iOS: <https://plus.google.com/communities/112026628790708717979>

Ambos os questionários coletaram informações sobre sobre o conhecimento atual do participante na área, também buscaram coletar informações sobte os fatores primários considerados quando escolhendo entre diferentes abordagens, frameworks e plataformas de desenvolvimento. Consideramos especialistas aqueles que se encaixam nos seguintes requisitos:

- 3 ou mais anos de experiencia profissional em desenvolvimento de software.
- Trabalhou em pelo menos dois produtos de software publicados.
- Trabalhou com tecnologias de desenvolvimento mobile recentes (a partir de 2015).

Os questionários foram estruturados da seguinte maneira:

- **Seção 1 - Perfil dos participantes e experiência prévia:** idade, gênero, conhecimento prévio de linguagens de programação (C, C ++, Java, Javascript, Python, Objective-C e Swift). Questão exemplo: "Quantos projetos foram publicados, quantos ainda estão em produção?"
- **Seções 2 e 3 - preferências arbitrárias em relação ao desenvolvimento de aplicativos móveis:** cobrem os idiomas e frameworks mais usados atualmente no desenvolvimento de dispositivos móveis. Por exemplo, Xcode (Objective-C e Swift) e Android Studio (Java) para desenvolvimento nativo, e framework Ionic e React-Native para abordagens híbridas;
- **Seções 4 - Fatores que interferem na seleção da tecnologia móvel:** a lista dos fatores avaliados está descrita na lista:

  1. Conhecimento prévio sobre a plataforma
  2. Curva de aprendizagem
  3. Requisitos do projeto de desenvolvimento
  4. Preferências das partes interessadas
  5. Reutilização de código
  6. Qualidade do suporte (Documentação, API, Comunidade, Wiki)
  7. Recomendações de especialistas da área
  8. Tendência de pesquisa do Google
  9. Desempenho final do produto (Consumo de bateria, tempo de resposta, desempenho gráfico)
  10. Não posso dizer, prefiro não responder.

Quando perguntados sobre cada um desses fatores, os respondentes têm diferentes opções. Eles podem selecionar os 3 principais fatores considerados ao iniciar um projeto de software.

4 Análise Explanatória

Esta seção apresenta os resultados relativos aos questionários do Capítulo 3 assim como uma análise explanatória mediante o resultado obtido.

Foram realizadas ao total três pesquisas para avaliar aspectos distintos sobre as abordagens mobile, seja nativa ou híbrida. Primeiramente, analisamos a curva de aprendizagem de ambas as abordagens, graduandos e mestrandos em ciências da computação implementaram pela primeira vez, funcionalidades consideradas simples tanto no Android nativo quanto no framework híbrido Ionic. Analisamos suas percepções e dificuldades ao configurar o ambiente de desenvolvimento, ao utilizar IDEs, linguagem de programação, testando e implementando um aplicativo em ambas as abordagens. No segundo questionário, analisamos quais fatores os desenvolvedores com menor experiencia levam em consideração na escolha entre abordagens nativas ou híbridas ao desenvolver um aplicativo. Por fim, avaliamos quais fatores os desenvolvedores mais avançados se baseiam ao escolher a abordagem mais adequada para se desenvolver um aplicativo mobile.

Tabela 3 – Sumário dos questionários.

Questionário	**Respostas**	**Perfil do Píblico Alvo**	**Objetivo do Questionário**
#1	40	Estudantes Graduandos e Mestrandos em Ciência da Computação	Observar a comparação técnica entre nativo (android) e híbrido(Ionic)
#2	96	Estudantes Graduandos em Engenharia de Software	Avaliar os fatores que um desenvolvedor com pouca experiencia considera ao escolher uma tecnologia para um projeto mobile.
#3	30	Desenvolvedores Mobile Experientes	Avaliar os fatores que um desenvolvedor experiente/especialista considera ao escolher uma tecnologia para um projeto mobile.

A Tabela 3 apresenta uma visão geral superficial das três pesquisas. A partir dos dados adquiridos, extrairemos aspectos não técnicos e técnicos que podem orientar os desenvolvedores a escolher entre abordagens nativas e híbridas ao iniciar um novo projeto mobile.

4.1 Primeiro Questionário

Realizamos os experimentos relacionados ao Android (nativo) e ao Ionic (híbrido) na disciplina de computação mobile da USP. No final da disciplina, um total de 40 de 53 alunos responderam ao questionário. Como observado na Figura 5 A maioria dos alunos tinha uma boa experiência nas linguagens de programação C, Java e Python. Na linguagem C, 80% declararam ter habilidades "boas" ou "excelentes". Para Python, 42% relataram ter o conhecimento "médio". 32,5% dos respondentes têm domínio "bom" a "excelente" sobre a linguagem. Mais de 65% declararam ter conhecimento "zero" ou "mínimo" de JavaScript (usado principalmente por Ionic), PHP e linguagens de programação Ruby.

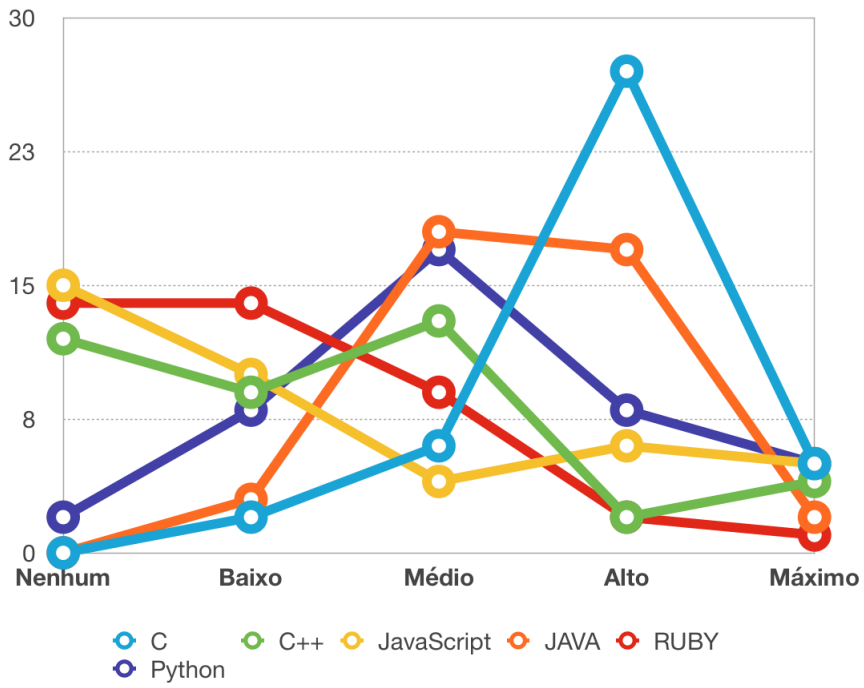

Figura 5 – Gráfico de perfil de linguagens de programação USP.

Em relação ao IDE e ao Framework, quase 87,5% dos entrevistados responderam que não tinham experiência com o Android Studio e 80% afirmaram o mesmo para o Ionic. Após o desenvolvimento utilizando ambas as tecnologias (Android e Ionic), os alunos foram questionados sobre a questão "Quão fácil é configurar o ambiente de desenvolvimento?", como observado na Figura 6, 67,5% dos participantes disseram que é fácil configurar o Android Studio, enquanto apenas 35% tinham a mesma opinião sobre o ambiente iônico. Mais da metade dos estudantes (55%) afirmou que o ambiente jônico não é natural de se configurar.

Em geral, os resultados mostram algumas evidências de que o desenvolvimento de tarefas simples, tanto o Android quanto o Ionic, são bastante semelhantes, dados evidenciados nas Figuras 7 e 8.

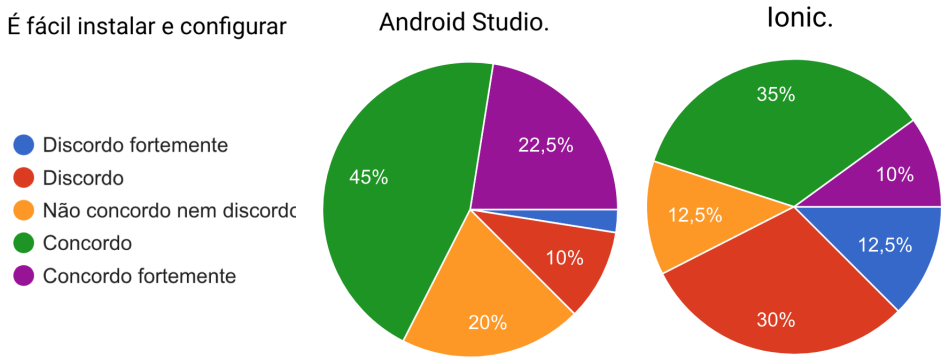

Figura 6 – Gráfico de respostas sobre facilidade de configuração Android e Ionic.

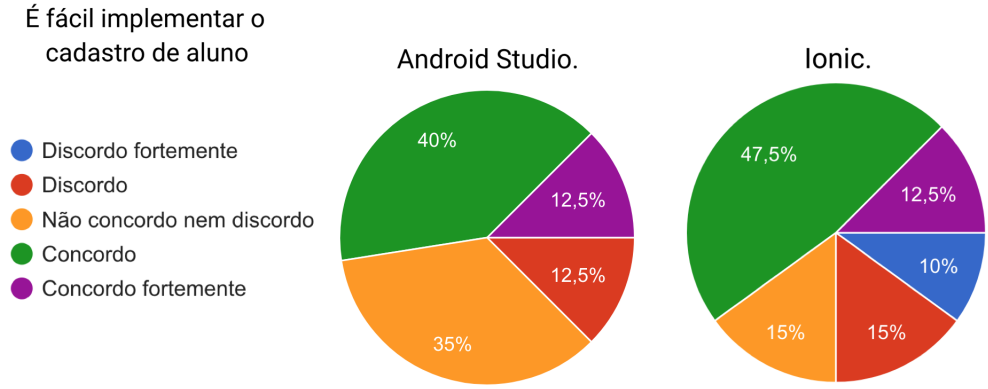

Figura 7 – Gráfico de respostas sobre facilidade de implementar Web Service em Android e Ionic.

Verificamos a implementação que utilizam os recursos de Bluetooth, Wifi e câmera em ambas as plataformas. Nossos resultados indicaram que os desenvolvedores consideram mais complicado acessar o hardware quando utilizando Ionic do que utilizando Android Studio como representado nas Figuras 9, 10, 11.

Neste questionário existem duas perguntas a respeito da percepção das vantagens e desvantagens, uma para Android Studio (21 respostas) e a segunda sobre os aspectos do Ionic (24 respostas). Em geral, de acordo com as respostas destas perguntas, os alunos apresentaram boa satisfação e maior facilidade em relação à IDE Android Studio; a maioria dos comentários foi positiva, comentários como: “IDE bastante útil”, “excelente depurador” e “boa ferramenta de terminal, log e ferramenta de refatoração”. No entanto, muitos deles se queixaram sobre a curva de aprendizado que era bastante acentuada, das dificuldades de configuração do ambiente de desenvolvimento e de métodos obsoletos (*deprecated*).

Ao que tange o framework Ionic, os estudantes foram bastante consistentes em seu uso para aplicações diretas. Houveram comentários como “Fácil de fazer interfaces HTML e navegar entre páginas” e “Fácil para desenvolvedores com conhecimento em HTML e

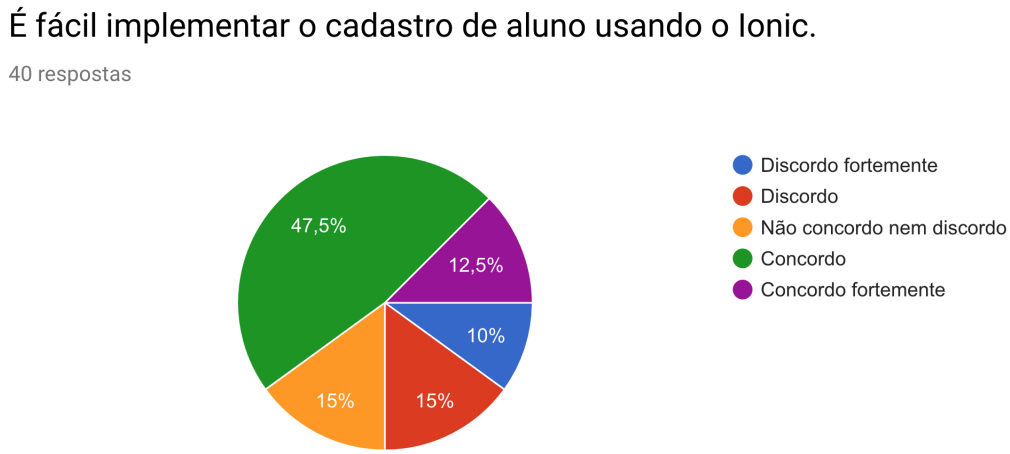

Figura 8 – Gráfico de respostas sobre facilidade de implementar cadastro de usuário em Android e Ionic.

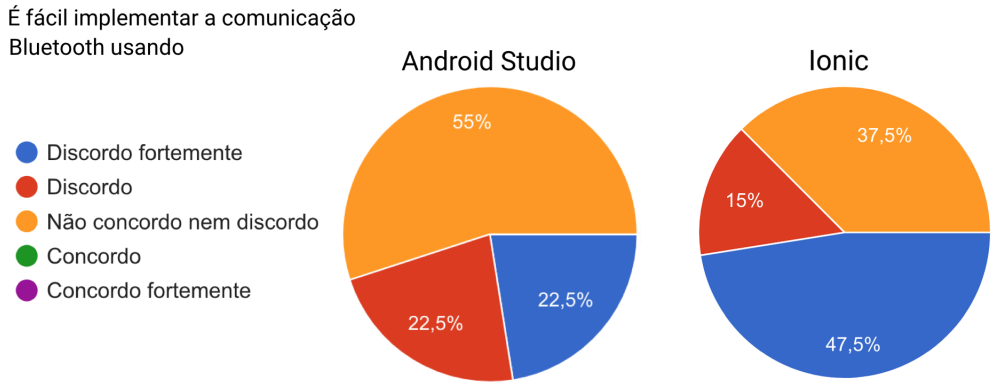

Figura 9 – Gráfico de respostas sobre facilidade de implementar uma funcionalidade com Bluetooth em Android e Ionic.

Javascript". No entanto, haviam diversas críticas relacionadas a vários tópicos como: "Não é bom usar os recursos nativos do dispositivo", "difícil encontrar as bibliotecas corretas", "grande dependência de *plugins* de terceiros". Os alunos também se queixaram do fato de o framework ser bastante recente, com documentação insuficiente, a "falta de documentação abrangente e falta de suporte", "difícil de configurar", "a sensação de comprar um aplicativo Frankenstein com diferentes partes funcionais", "a final o aplicativo era mais lento que o nativo", "difícil de depurar" e "É frustrante fazer com que as solicitações HTTPS funcionem".

4.2 Segundo Questionário

O segundo questionário resultou em 96 respostas de alunos de graduação em Engenharia de Software da Universidade de Brasília (UNB). Esta pesquisa tinha como objetivo

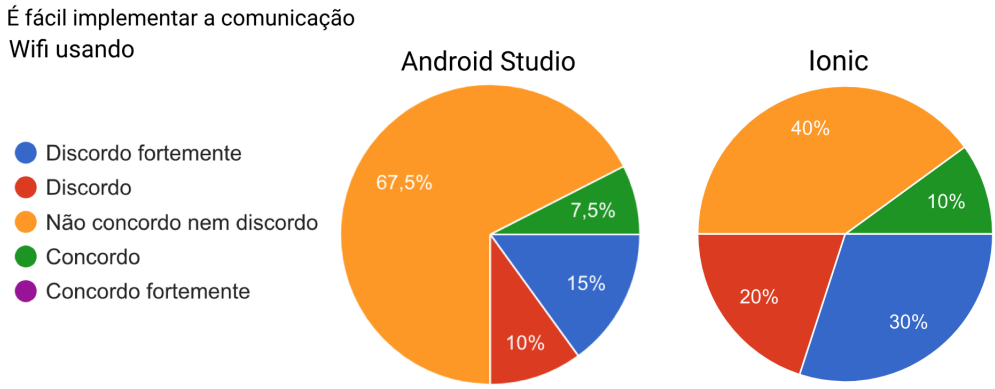

Figura 10 – Gráfico de respostas sobre facilidade de implementar uma funcionalidade com WiFi em Android e Ionic.

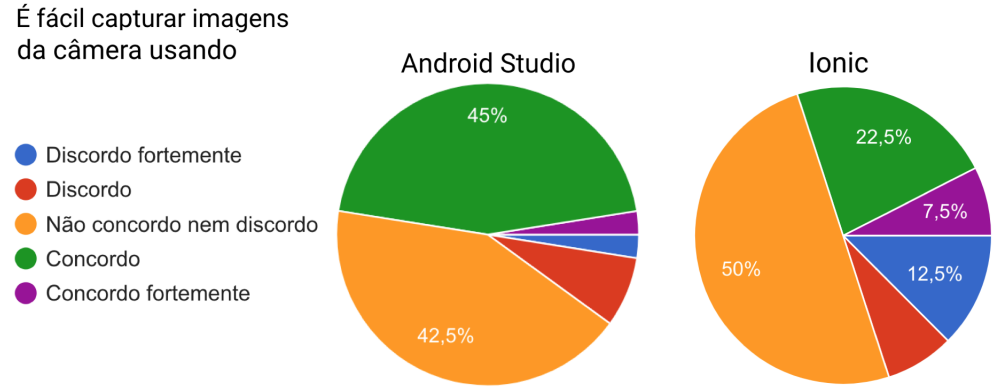

Figura 11 – Gráfico de respostas sobre facilidade de implementar uma funcionalidade com a câmera do dispositivo em Android e Ionic.

entender como os desenvolvedores, com pouca experiencia em desenvolvimento de software para dispositivos móveis, escolhem as técnicas e ferramentas de desenvolvimento para seus primeiros projetos mobile. Os estudantes avaliados são em sua maioria homens (85%) e tem faixa etária entre 18 e 25 anos.

Quando questionados sobre seu conhecimento atual sobre linguagens de programação, 74%, 58,4% e 52,1% dos alunos declararam, respectivamente, experiencia média ou alta em C / C ++, Java e Python. Em contraste, as linguagens da plataforma iOS, Swift e Objective-C, foram as mais obscuras entre todos os estudantes, apresentando 88,5% das respostas com "conhecimento zero na linguagem". Observamos que o Javascript apresentou um espaço amostral bem distribuído, com 29,2% dos entrevistados com "conhecimento baixo ou zero", 44% com "conhecimento médio" e 26,1% com "conhecimento elevado", podemos evidenciar mais claramente o perfil geral dos candidatos na Figura 12.

Perguntamos sobre quanto tempo os alunos estudaram ou trabalharam com o desenvolvimento de software como um todo e também o tempo com o desenvolvimento de

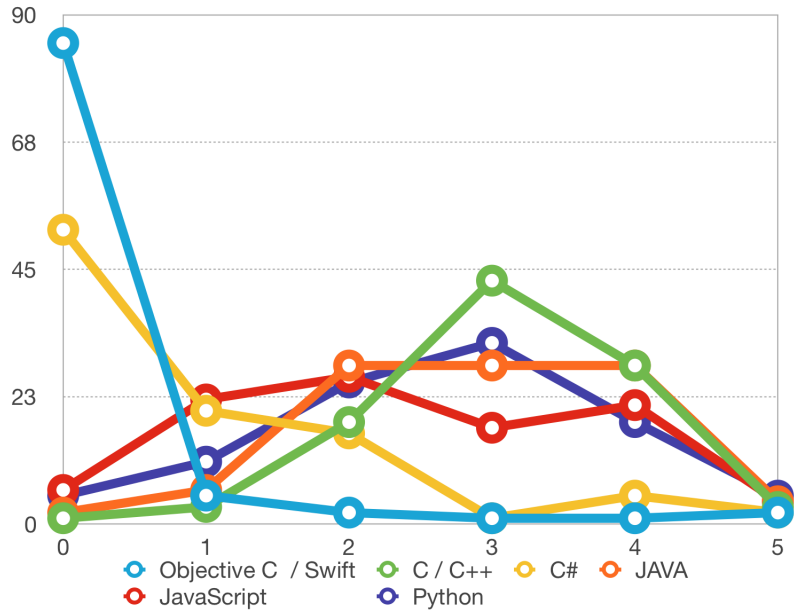

Figura 12 – Gráfico de perfil de linguagens de programação UNB.

software mobile. Como observado nas Figuras 13 e 14, 53,1% dos respondentes têm entre um e três anos de prática e 77,1% dos alunos têm 0 a 1 ano de prática em desenvolvimento de software móvel.

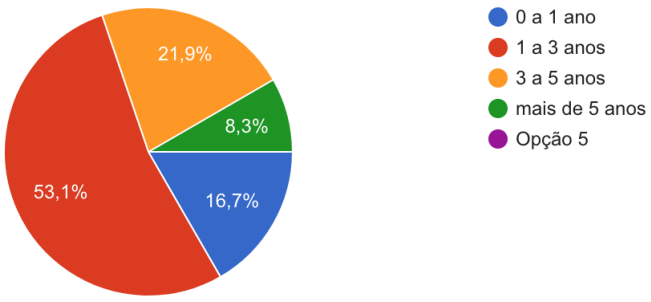

Figura 13 – Gráfico de tempo de experiência em desenvolvimento de software dos estudantes UNB.

Os candidatos foram questionados sobre frameworks, plataformas e IDEs específicos que eles usam durante o desenvolvimento híbrido; 42,7% dos alunos têm prática pelo menos em Ionic, React Native, PhoneGap, Xamarin ou Rails, representado no gráfico da Figura 14. Quanto às abordagens nativas, o Android foi a plataforma nativa mais conhecida para desenvolvimento entre os estudantes com 49% das respostas. De acordo com esses dados, podemos observar que os alunos têm prática com tecnologias multiplataforma.

Os alunos também selecionaram três fatores (listados na Seção 3.2) que seriam decisivos ao escolher que plataforma, framework, tecnologia ou abordagem para desen-

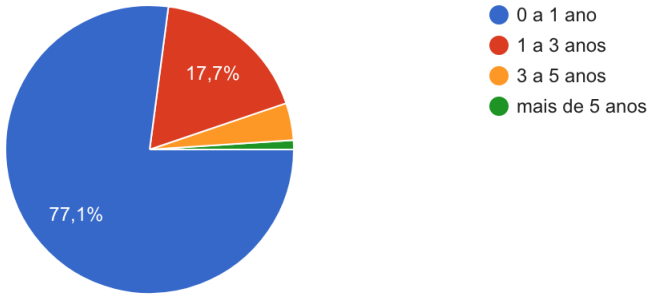

Figura 14 – Gráfico de tempo de experiência em desenvolvimento de software mobile dos estudantes UNB.

volver um projeto de aplicativo mobile. A Figura 15 descreve os resultados, onde 67,7% dos respondentes indicaram que "qualidade do suporte (wiki, comunidade, documentação, API)" e os "requisitos do projeto" como os fatores com a maior importância. Em segundo lugar, ficaram os fatores "conhecimento prévio" e "curva de aprendizado" (ambos tópicos não técnicos), com 53,1% e 44,8% de respostas, respectivamente. É evidente que para os iniciantes, os fatores mais importantes para se escolher qual abordagem é mais adequada são o suporte técnico, documentação e comunidade disponível.

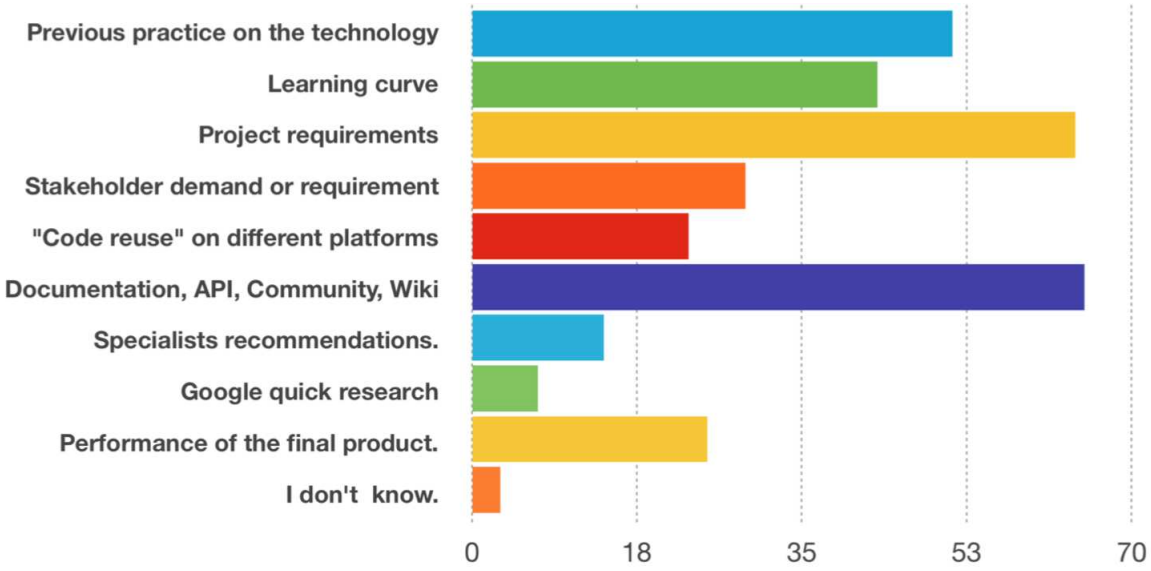

Figura 15 – Gráfico de fatores que influenciam na escolha da abordagem segundo desenvolvedores com menor experiência.

Isso significa que desenvolvedores inexperientes de celulares provavelmente não serão os primeiros a adotar tecnologias emergentes móveis, uma vez que aumentam os riscos associados ao desenvolvimento de projetos. Igualmente crucial para os iniciantes são os requisitos do projeto e a experiência anterior na tecnologia. No entanto, quando se trata de requisitos de projetos, eles tendem a escolher a abordagem híbrida quando é necessária

disponibilidade e portabilidade para múltiplas plataformas. Esse fato é devido a prazos rígidos, orçamentos limitados e gerenciamento de riscos. Consequentemente, se o requisito do projeto é ter o aplicativo disponível em apenas uma plataforma, os desenvolvedores inexperientes tendem a escolher a abordagem nativa.

4.3 Terceiro Questionário

Como mencionado anteriormente o terceiro questionário teve como foco especialistas e desenvolvedores experientes em aplicativos mobile. Esse questionário resultou em 30 respostas de desenvolvedores avançados que participam ativamente da comunidade de desenvolvedores. Os profissionais avaliados são em sua maioria homens, com faixa etária de 22 a 35 anos (80 %).

Como observado na Figura 16, quando questionados sobre seu conhecimento atual sobre linguagens de programação, a maior parte declarou ter média ou boa experiência em Objective-C/Swift (56,7%), Java (66,6%), Javascript (53,5%); as mesmas se destacaram mediante outras linguagens como PHP, Ruby, C#, Python e C/C++.

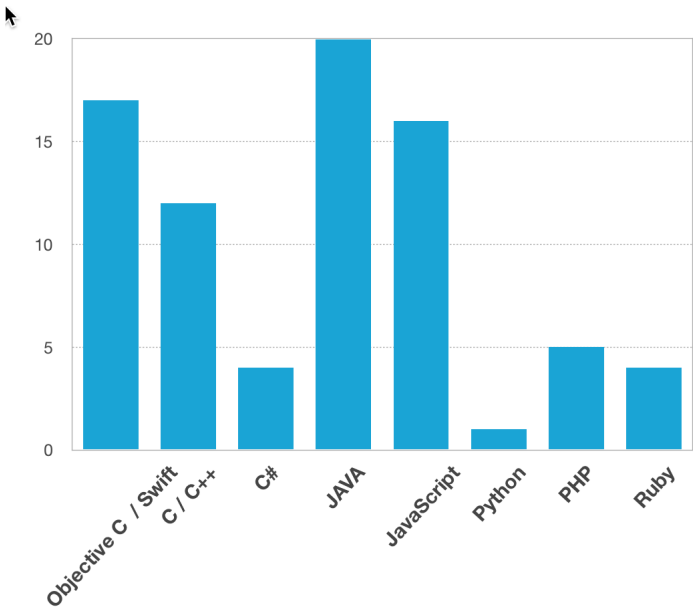

Figura 16 – Gráfico de experiência média ou superior em linguagens de programação para especialistas da área

Quando discutimos sobre suas preferências empíricas referentes a cada tecnologia (fora de qualquer caso de uso específico), podemos observar na Figura 17 que entre as tecnologias de desenvolvimento multiplataforma, a comunidade de desenvolvedores mais representativa são referentes às tecnologias Xamarin e React Native, e WebApps, todos com pouco mais de 30% de qualquer nível de aprovação entre baixo e máximo (medi-

ante todas as outras tecnologias).A preferência por alternativas nativas (iOS e Android) permanece em 90% de qualquer nível de preferência.

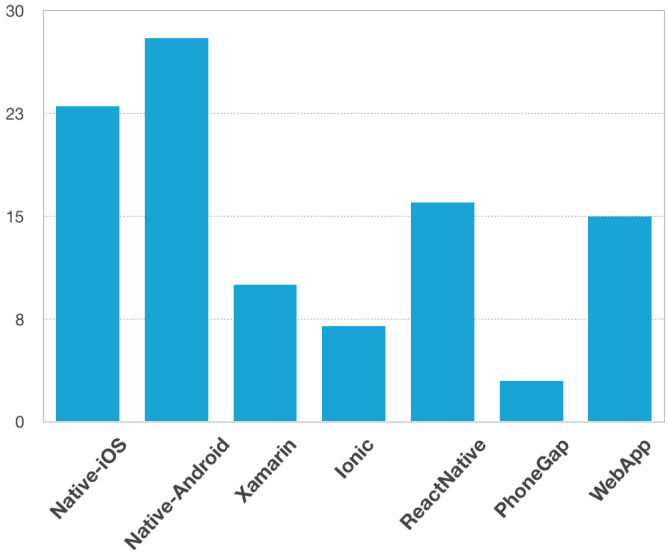

Figura 17 – Gráfico de aceitação mínima sobre diferentes tecnologias de desenvolvimento mobile.

Assim como no segundo questionário, os desenvolvedores tiveram que escolher entre uma lista de fatores, até 3 fatores que influenciam a escolha do framework ou tecnologia ao iniciar um projeto de desenvolvimento de software mobile. Observamos na representação grafica na Figura 18 que assim como os estudantes, “qualidade do suporte” (documentação, Wiki, APIs) com 46,7% foi o aspecto mais relevante para a escolha de uma tecnologia, mas também o aspecto “Desempenho do produto final” também 46,7% em contraste com pesquisas anteriores . Curva de aprendizado (36,7%) e conhecimento prévio de tecnologia (33,7%) também foram considerados novamente, se assemelhando aos resultados de questionários anteriores.

No questionário, apresentamos aos participantes aplicativos mobile que hipoteticamente necessitam ser desenvolvidos para ambas as plataformas iOS e Android. Em cada opção, eles devem escolher qual tecnologia (nativa, híbrida, WebApp) seria a sua preferência pelo desenvolvimento de software. Os tipos de aplicativos foram: (1) “Jogo eletrônico”, (2) "Aplicativo baseado em CRUD", (3) “Aplicação com elementos complexos de animações e UI/UX"e (4) "Aplicativo que depende muito dos recursos do dispositivo (GPS, acelerômetro, giroscópio, iCloud, login nativo, Touch ID, Notificações push, etc.)". Para as alternativas (3) e (4), uma preferência significativa em relação à abordagem nativa foi evidente (90%), para aplicações baseadas em CRUD, a maioria teve preferência por abordagens multi-plataforma, com 40% híbridos e 13,13% Web. -Aplicativo. Finalmente, para os jogos eletrônicos, houve uma preferência de pouco mais de 50% para a abordagem nativa e 3% para abordagens híbridas (Unity, Unreal Engine e Game Engine), podemos

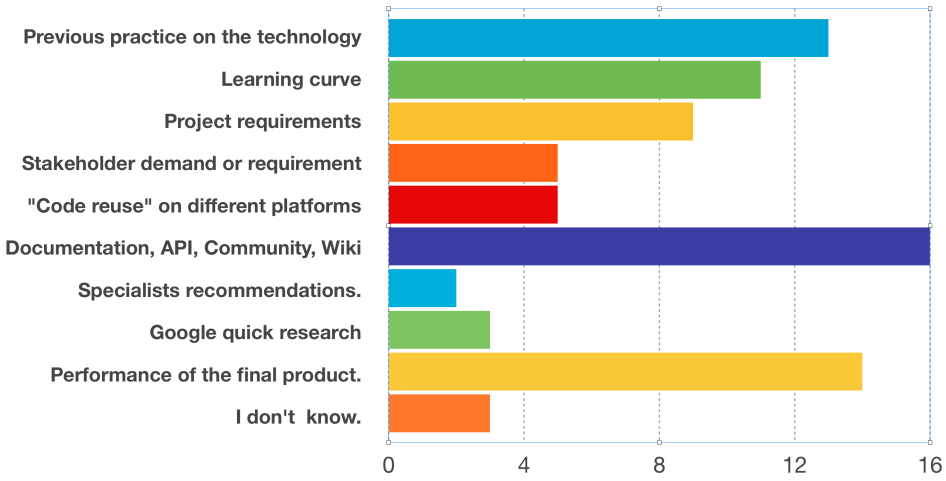

Figura 18 – Gráfico de fatores que influenciam na escolha da abordagem segundo desenvolvedores especialistas.

visualizar graficamente os resultados na Figura 19.

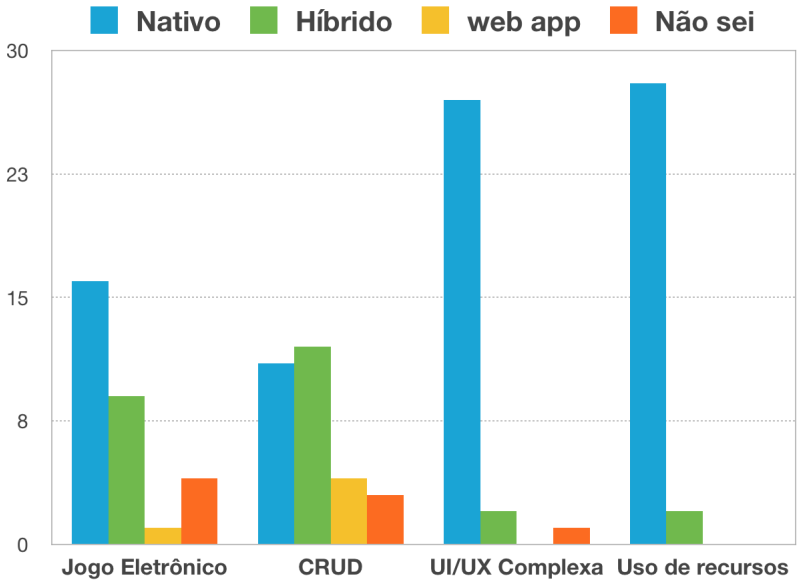

Figura 19 – Gráfico de preferências de abordagem para tipos diferentes de aplicativo.

5 Proposta de guia comparativo

Nossos resultados mostram que, do ponto de vista do profissional, o aspecto mais crítico a ser considerado na escolha de uma abordagem de desenvolvimento móvel é “quão bem estabelecida e consolidada é a tecnologia”. A resposta a essa pergunta depende muito do tipo de aplicativo móvel, o que corrobora nossa hipótese de que não há uma escolha ideal para todas as situações. Isso significa que os desenvolvedores devem verificar quão boa ou extensa é a documentação técnica (por exemplo, documentação de código, wiki, fóruns) e quão ativos são seu suporte e comunidade. Esse aspecto está diretamente ligado à estabilidade e consolidação da tecnologia mobile escolhida na comunidade de desenvolvedores em geral. A experiência prévia e a curva de aprendizado da equipe de desenvolvimento em relação à tecnologia candidata também influenciam essa decisão.

Podemos observar que algumas das afirmações que nós teorizamos provaram ser verdadeiras. Esperávamos que: ao contrário do que a literatura e estudos recentes apontam, abordagens híbridas seriam a melhor escolha para a maioria dos casos. Essa afirmação parece ser verdadeira mesmo que várias tecnologias híbridas já estejam bem consolidadas. Podemos observar que a escolha tende a abordagens nativas quando aplicativo necessita de elementos complexos de UI / UX e animações. A mesma opção também é valida para aplicativos que tem como requisito, a utilização de recursos nativos não comuns a outros dispositivos. Em relação aos jogos eletrônicos para celular, a prevalência de preferência ainda é a abordagem nativa, mas há uma crescente presença de abordagens híbridas que estão atualmente bem estabelecidas (por exemplo, Unity, Unreal Engine, Game Engine).

Desde que a abordagem selecionada não entre em conflito com os requisitos do projeto, é apenas uma questão de selecionar uma abordagem mais adequada. Concluímos a partir dos dados coletados que não há escolha ideal para todas as situações, onde cada projeto e cada tipo de software tem suas particularidades. A partir dos resultados obtidos a partir da análise dos dados, propusemos um conjunto geral de recomendações descritas na Tabela 4 e descritas a seguir.

Abordagem Nativa: A abordagem nativa apresenta o melhor desempenho gráfico de UI/UX e maior facilidade para se implementar animações complexas. Esta abordagem proporciona o máximo de conformidade com os padrões de design da plataforma em questão (Android ou iOS). As tecnologias nativas mais utilizadas se destacam em sua vasta documentação, suporte técnico para correções de bugs e extensa comunidade de desenvolvedores já com projetos em andamento; em geral os fornecedores (Apple para iOS e Google para Android) oferecem uma variedade de ferramentas de desenvolvimento bem estabelecidas.Com a implementação adequada é possível se obter máxima otimização para

Tabela 4 – Resumo do guideline proposto.

	Native	**Hybrid**	**WebApp**
UI/UX performance	High	Low	Low
UI/UX customization	High	Medium to Low	Relative to web technology adopted.
UI/UX platform guideline compliance	Maximum	Medium to High	None
Documentation, support, community	High	Low to Medium . (relative to the selected hybrid technology)	Relative to web technology adopted. But low in the context of web app
Hardware stress (battery consumption, RAM, GPU, response time)	Low	Medium to High, (Very perceptive in some cases)	High to Maximum
Optimized product output (binary or generated source)	Can be very optimized (relative to developer)	Not optimized	Not optimized
Relative cost per number of platform.	Maximum cost per platform.	medium or low . (relative to the selected hybrid technology)	Minimum cost per platform
Other comments	Easy to setup working environment, high number of developer tools; maximum optimization for game softwares.	Usually longer to setup working environment (with exceptions);	Limited access to device resources
Recommended for	For projects with a lower priority in budget, projects that don't need to be deployed on multiple platforms, apps that need high or maximum performance are required as well as a need of a good support and learning curve in the development team.	For projects targets two or more platforms, require access to basic device resources, projects that don't require excellency in graphical performance or compliance and have a high priority of a lower budget.	For apps that have simple UI/UX requirements don't need to be stored in the device, apps that need to run in an undetermined numberof devices and applications that don't require device specific features (camera, accelerometer, touch-id, gyroscope, compass, proximity, microphone).

baixo consumo de bateria, uso de RAM, memória de vídeo e latência. Recomendamos a abordagem de desenvolvimento nativo para:

- Projetos que precisam de alto desempenho;
- Projetos que requerem suporte técnico dos provedores da tecnologia;
- Projetos que precisam de uma curva de aprendizado mais rápida para desenvolvedores experientes.

Abordagem Híbrida:A abordagem híbrida apresenta uma conformidade com os padrões de UI/UX aceitável, em geral conseguem reproduzir os elementos básicos da respectiva interface nativa. No entanto, apresentam uma dificuldade maior durante implementação de animações e elementos gráficos não comuns (não padronizados) ou mais complexos. Essa abordagem se destaca no custo benefício em relação ao orçamento e ao cronograma quando comparado ao número de plataformas que podem ser implementadas. Isso acontece devido à sua principal vantagem, o reuso de código entre diferentes plataformas, mesmo que um código de plataforma específico ainda seja necessário para recursos nativos específicos. Nem todas as tecnologias híbridas fornecem um ambiente de desenvolvimento bem estabelecido (IDEs, SDKs, CLIs). As abordagens híbridas mais usadas atualmente também fornecem documentação extensa, wiki e suporte de suas comunidades. O produto final ou aplicativo binário não é otimizado para uma plataforma individual, levando a uma sobrecarga de hardware maior do que a solução nativa específica. Recomendamos a abordagem híbrida para:

- Projetos destinados a duas ou mais plataformas;
- Projetos que exigem acesso a recursos básicos do dispositivo;
- Projetos que não exigem excelência em gráficos desempenho ou conformidade;
- Projetos que possuem restrições orçamentárias.

Abordagem WebApp:A abordagem do Web App apresenta a menor conformidade com a interface do usuário / UX, pois suas estruturas não estão em conformidade com as diretrizes específicas da plataforma. Essa abordagem é excelente para casos de uso que já possuem um ambiente de produção e recursos simples do usuário final (por exemplo, aplicativos baseados em CRUD). Essa abordagem tem o melhor nível de reutilização de código. Conforme mencionado neste documento, o navegador da web de dispositivos é usado para acessar o aplicativo da web, levando à mesma experiência em cada plataforma. Em contrapartida de sua maior compatibilidade, o acesso aos recursos do dispositivo é muito limitado. Recomendamos esta abordagem para:

- Projetos que possuem requisitos simples de UI / UX;
- Projetos que não precisam ser completamente armazenados o dispositivo;
- Projetos que precisam ser executados em um número indeterminado de dispositivos;
- Projetos que não exigem recursos específicos do dispositivo (por exemplo, câmera, acelerômetro, touch-id, giroscópio, bússola, proximidade, microfone)

A partir da análise dos dados, concluímos que os profissionais, independentemente da sua experiência, utilizam fatores técnicos e não técnicos. Descobrimos que, na prática, o aspecto central a ser considerado na escolha de uma abordagem para desenvolvimento mobile, são aspectos não técnicos como: o suporte oferecido pela tecnologia (documentação, wiki e comunidade). Outros fatores essenciais não técnicos (mas não restritos a ele) para orientar a decisão entre as abordagens são: desempenho do produto final, necessidade do projeto e curva de aprendizado.

Quando se trata de aspectos técnicos, estudamos as vantagens e desafios da abordagem móvel nativa, híbrida e web app. Realizamos um experimento com desenvolvedores intermediários, onde eles implementaram recursos em Java nativo Android e Ionic e, posteriormente, compararam sua experiência. Os alunos tiveram conclusões semelhantes às dos especialistas, onde concluímos que os profissionais estão fortemente inclinados a escolher uma abordagem nativa quando se trata de alcançar alto desempenho (baixo consumo de bateria, baixa latência, alto desempenho gráfico, animações, baixo uso de memória) . No entanto, se o desempenho não for obrigatório e houver poucos acessos de hardware / dispositivo, as soluções híbridas serão consideradas uma abordagem melhor para desenvolver aplicativos móveis para várias plataformas.

Para mitigar ameaças à validade das recomendações feitas, este trabalho propõe a aplicação destas mesmas recomendações em um contexto de testes, buscando confirmar se algumas afirmações, resumidas na Tabela 4 inicialmente se provam como verdadeiras. Para executar tal teste confirmação trabalho também propõe a implementação de um exemplo de uso, que consiste em uma aplicação "teste"; esta aplicação será implementada utilizando uma abordagem nativa (iOS) e uma abordagem híbrida (Ionic). Ao fim desta implementação serão analisados os aspectos técnicos de cada uma das soluções, utilizando também de ferramentas de *benchmarking* (para comparações como o consumo de RAM), e aspectos não técnicos relativos ao desenvolvimento de tais soluções. Após a execução deste exemplo de uso e sua análise, teremos a confirmação de algumas veracidades propostas pelo guia, assim como uma possível iteração do mesmo. No restante deste trabalho apresentaremos os detalhes do teste executado para a confirmação das informações obtidas, tal como seus resultados e como esses resultados se apresentam mediante o guideline.

6 Exemplo de Uso

Nesta seção descreveremos mais a fundo o exemplo de uso que será utilizado para a confirmação de informações declaradas no guideline. Esta confirmação será feita inicialmente através deste experimento. Em suma o mesmo consistirá na implementação de um aplicativo utilizando as plataformas iOS e Ionic, as quais são tecnologias nativa e híbrida respectivamente, estima-se que todas as *features* listadas na Seção 6.1 sejam implementadas em aproximadamente 8 horas para cada plataforma (estimativa baseada em *planning poker* realizado com 5 desenvolvedores). Após a implementação do aplicativo em ambas as plataformas será realizado um *benchmark* comparando a performance de ambos, buscando repostas de consumo de bateria, consumo de RAM, uso de CPU, uso de rede, tamanho do binário, tempo de desenvolvimento. Ao final do desenvolvimento os desenvolvedores farão considerações livres sobre o desenvolvimento e o produto final.

Na Subseção 6.3 serão descritos os resultados do exemplo de uso. Também será feita uma discussão e análise dos resultados obtidos e sua correlação com o guia comparativo (guideline) anteriormente proposto.

6.1 O aplicativo

Um dos grandes argumentos das tecnologias híbridas é que é possível utilizar "facilmente"os recursos nativos através de plugins na plataforma, em contra partida um dos grandes argumentos da abordagem nativa é o fácil acesso a recursos de sistemas. Estas duas premissas, não exclusivamente, serão o foco deste aplicativo. O aplicativo consistirá em uma tela principal listando várias *features* as quais utilizam de recursos nativos do aparelho, representado nos protótipos de baixa fidelidade na Figura 20. Não é o objetivo deste aplicativo a implementação de elementos complexos de UI/UX e animações. As tecnologias escolhidas foram a linguagem Swift utilizando de frameworks e IDEs nativas, e o Framework Ionic para o desenvolvimento híbrido (utilizando das ferramentas necessárias para seu desenvolvimento).

As *features* escolhidas para o aplicativo do experimento são:

- Download de arquivo;
- Upload de arquivo;
- Fotografar utilizando a Câmera do dispositivo;
- Localização do usuário em tempo real (GPS e mapa);

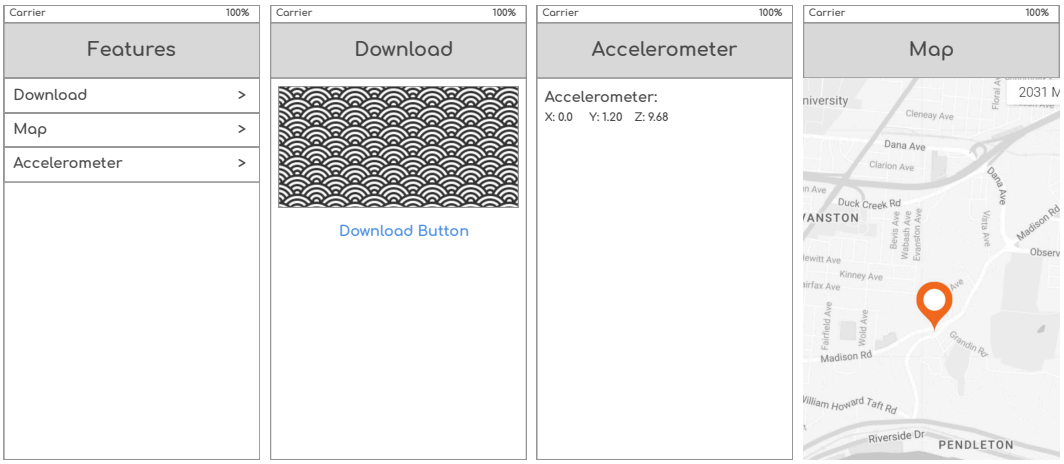

Figura 20 – Prototipo de baixa fidelidade.

- Visualização Acelerômetro e Giroscópio em tempo real;
- Visualização de dados da bússola do dispositivo;
- Autenticação por biometria (TouchID ou FaceID para iOS);

6.2 Os desenvolvedores

Para a realização do teste foram selecionados dois desenvolvedores de perfil semelhante para a implementação de cada uma das versões (nativa e híbrida) do aplicativo. Para a implementação da versão nativa em iOS o desenvolvedor A responsável possui o perfil de "especialista em mobile"e médio conhecimento sobre tecnologias de desenvolvimento web e híbrido (Ionic Framework), aproximadamente a 5 anos desenvolvendo aplicativos e frameworks mobile [1]. De acordo com este perfil é esperado que o desenvolvedor não apresente dificuldades técnicas durante o desenvolvimento do aplicativo nativo e finalize todas as *features* dentro de um prazo estimado de 8 horas. O desenvolvedor responsável pelo aplicativo híbrido, implementado em Ionic, possui possui um perfil de "especialista mobile e especialista web", aproximadamente a 6 anos desenvolvendo aplicativos e sistemas web [2]. Os desenvolvedores foram escolhidos com este perfil para se desconsiderar o critério de "falta de afinidade com as linguagens da plataforma", tal critério deve ser considerado em futuros experimentos como os citados no Capítulo 7.

[1] GitHub:<https://github.com/fel-cesar>
[2] GitHub:<https://github.com/harimasora>

6.3 Resultados e Análise

6.3.1 O Desenvolvimento do aplicativo

A versão nativa do aplicativo foi implementada para iPhone Operating System (iOS) versão 11, o desenvolvedor utilizou da documentação oficial da plataforma para consulta [3], não havendo necessidade de consultas a fóruns de desenvolvedores para resolução de eventuais problemas durante o desenvolvimento. Todas as ferramentas e IDEs utilizadas também foram exclusivamente providenciadas pela plataforma escolhida, não sendo necessárias configurações externas à IDE (Xcode e UIKit). Todas as funcionalidades foram implementadas como descritas na Sessão 6.1, demonstrado na Figura 21. O tempo total de desenvolvimento do aplicativo foi de 4 horas.

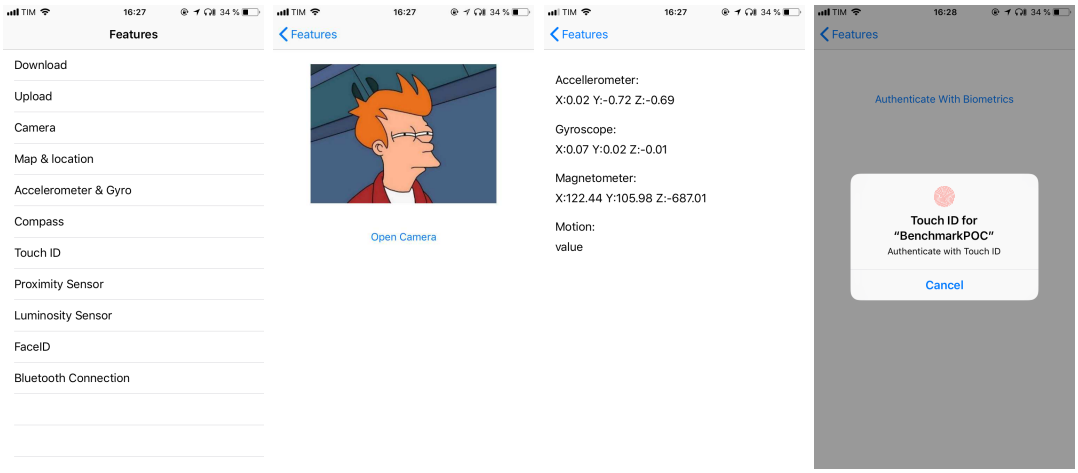

Figura 21 – Capturas de telas da versão iOS do aplicativo.

A versão híbrida do aplicativo foi implementada para todas as plataformas, porém compilada apenas para iOS versão 11, essa decisão foi tomada para se obter parâmetros mais próximos de comparação. Todas as funcionalidades foram implementadas como descrito na Sessão 6.1, demonstradas na Figura 22. O tempo total de desenvolvimento do aplicativo foi de aproximadamente 16 horas, durante seu desenvolvimento foram utilizados para consulta a documentação oficial da plataforma Ionic[4], o fórum de desenvolvedores da plataforma Ionic[5] , o fórum de desenvolvedores StackOverflow[6] e também a documentação oficial do iOS. As ferramentas utilizadas para o desenvolvimento foram a IDE Atom, compilador Ionic, compilador Cordova e Xcode.

Podemos observar que a primeira grande diferença entre os desenvolvimentos é o tempo de produção, a implementação do aplicativo em Ionic necessitou do dobro do tempo estimado para ser implementado, mesmo com determinado conhecimento prévio

[3] Referência:<https://developer.apple.com/documentation/>
[4] Referência:<https://ionicframework.com/docs/>
[5] Referência:<https://forum.ionicframework.com/>
[6] Referência:<https://stackoverflow.com/>

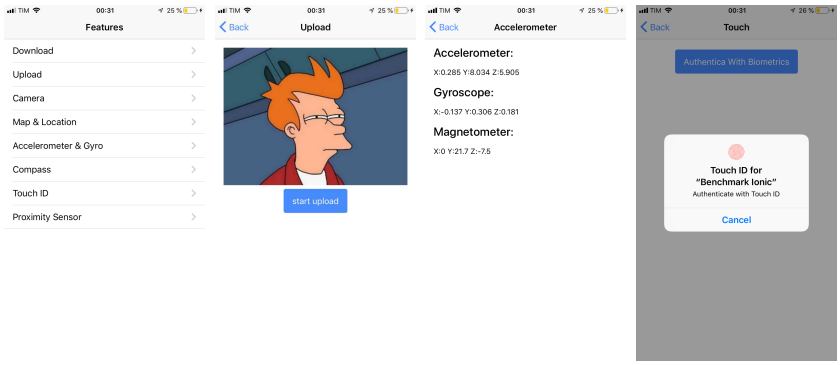

Figura 22 – Capturas de telas da versão iOS do aplicativo.

na plataforma selecionada e também conhecimento em tecnologias web, as quais são as mesmas utilizadas para desenvolvimento com o Ionic Framework (HTML, CSS, Javascript, AngularJS). Este resultado corrobora com com nossa hipótese e com o guideline ao determinarmos que a "curva de aprendizado"de plataformas híbridas no geral tendem a ser mais lentas; quando arguido sobre este fato em uma entrevista informal, o desenvolvedor afirmou que a documentação da plataforma híbrida era superficial, e que ao necessitar de algo mais especifico, o que não necessariamente é algo complexo, teve dificuldades em encontrar soluções adequadas. O desenvolvedor da plataforma híbrida também afirmou que teve dificuldades em configurar o ambiente de desenvolvimento para a plataforma, necessitando desde o inicio recorrer ao fórum de duvidas StackOverflow; tal fato corrobora tanto com os resultados obtidos no primeiro questionário citado na Seção 3 quanto com o que afirmamos em nosso guideline em validação. Podemos também ressaltar que embora todas as funcionalidades foram implementadas (através de plugins) na versão híbrida do aplicativo.

6.3.2 O Benchmark do Aplicativo

Os testes de *benchmark* foram realizados utilizando XCode 9.4 e ferramenta Instruments, distribuída juntamente com o ambiente de desenvolvimento nativo iOS no sistema operacional macOS 10.14 em um MacBook Pro (Retina, 13-inch, Late 2013) com 8GB de memória RAM. O dispositivo o qual executou os aplicativos foi um iPhone 6s modelo A1688.

As medições foram realizadas com sucesso em ambas as versões nativa e híbrida do aplicativo, os resultados podem ser observados nas Tabelas 5 e 6.

De acordo com o guideline, para uma aplicação que se utiliza fortemente de recursos nativos do sistema, escopo o qual o aplicativo é focado, a recomendação primaria é a utilização de tecnologias nativas. Podemos observar de acordo com os resultados obtidos que o aplicativo híbrido consumiu até 390% RAM a mais do que sua versão nativa, corroborando com a recomendação do guideline que afirma que aplicativos híbridos em geral

Tabela 5 – Resultados do *benchmark* para plataforma nativa

Nativo	**Memória RAM (MB)**	**Uso de Rede (MB)**	**CPU (até 200%)**	**Consumo médio de bateria (0 a 20)**
Download	15.9	0.2	17%	12
Upload	17.12	0.2	17%	14
Câmera	18.4	0	17%	15
Map & Localização	89.1	0	100%	16
Acelerometro e Giroscopio	14.3	0	30%	10
Bussola	13.3	0	18%	12
Autenticação Biométrica	15.5	0	15%	13
Sensor de proximidade	13.8	0	18%	10

Tabela 6 – Resultados do *benchmark* para plataforma híbrida Ionic

Híbrido	**Memória RAM (MB)**	**Uso de Rede (MB)**	**CPU (até 200%)**	**Consumo médio de bateria (0 a 20)**
Download	47.4	4.4	40%	15
Upload	48.1	6.2	35%	16
Câmera	51.5	0	38%	17
Mapa & Localização	46.8	0	45%	19
Acelerometro e Giroscopio	47.4	0	60%	13
Bussola	46.8	0	23%	16
Autenticação Biométrica	47.7	0	20%	15
Sensor de proximidade	45.1	0	35%	11

consomem mais recursos do dispositivo, logo para aplicações, as quais fazem uso ostensivo dos recursos nativos que naturalmente aumentam o consumo de recursos do aparelho, recomenda-se tecnologias nativas. É válido observar que para a feature de "Mapa e Localização"a performance geral do aplicativo híbrido aparenta ser superior, considerando que a aplicação nativa utilizou de 100% de CPU e 89MB de RAM contrastando com 45% e 46.8MB de RAM consumidos pela aplicação híbrida; isso ocorre dado a característica da aplicação nativa utilizar do recurso "MKMapView"do iOS, o qual consiste em uma renderização em 3D do mapa com alta taxa de atualização (necessitando assim de alto recurso gráfico do aparelho), enquanto a mesma feature na aplicação híbrida foi implementada utilizando uma "UIWebView"com acesso ao GoogleMaps; quando arguido sobre a diferença de *features*, o desenvolvedor responsável pelo app Ionic afirmou não existir plugins necessários para acessar o recurso "MKMapView"diretamente do código web.

Ao observarmos o consumo de bateria médio durante a execução das *features*, como esperado dado o consumo de RAM e CPU e também enfatizado pelo guideline, podemos constatar que o consumo do aplicativo híbrido é em todas as *features* mais impactante do que sua alternativa nativa.

Ao observarmos o tamanho final dos aplicativos, ao contrário do que havíamos teorizado anteriormente e afirmado pelo guideline, obtivemos um aplicativo 53.1% menor com a tecnologia híbrida. Buscando analisar o motivo desta contrapartida, concluímos que o app Ionic possui um binário menor em consequência do fato de que todo o seu código de interface de usuário é interpretado (e não compilado), o que gera um maior consumo de CPU e RAM a troco de um menor espaço em disco.

Como considerações finais podemos afirmar que as recomendações feitas pelo guideline, as quais foram colocadas neste teste, foram confirmadas com os testes; com excessão da afirmação referente ao "tamanho final do binário do aplicativo", a qual observamos não estar correta para todos os casos em nosso experimento.

7 Conclusão

O objetivo principal deste trabalho de conclusão de curso era estudar os fatores determinantes que influenciam na decisão entre uma abordagem nativa ou multiplataforma. Através desta necessidade traçamos questões de pesquisa para nossa própria orientação, e para responder as mesmas elaboramos 3 questionários focados em públicos diferentes: desenvolvedores iniciantes, intermediários, e especialistas. Vale também destacar que a execução do questionário 2 foi acompanhada de um experimento no qual os estudantes que responderam ao mesmo, tiveram que implementar uma mesma solução em ambas as tecnologias (nativo e híbrido). Com a execução desta pesquisa concluímos que para os desenvolvedores mobile como um todo, os aspectos técnicos e não técnicos devem ser considerados para a escolha da abordagem a utilizar (nativa ou multiplataforma), porém, em geral os aspectos não técnicos como (mas não exclusivamente) "suporte oferecido pelos fornecedores da tecnologia, comunidade, documentação", tem prioridade considerável sobre alguns aspectos técnicos.

Ainda buscando uma comprovação sobre as afirmações declaradas conduzimos um exemplo de uso. Este exemplo de uso consiste na implementação de um aplicativo utilizando de duas tecnologias separadamente, iOS (nativo, Objective-C) e Ionic (híbrido, Javascript, HTML, CSS). Ao final de sua implementação observamos que varias comparações e afirmações declaradas pelo guia comparativo se provaram verdadeiras, afirmações como:"aplicações nativas possuem melhor curva de aprendizado", "aplicações híbridas consomem maior bateria", "aplicações híbridas possuem acesso a diversos (as vezes todos) recursos nativos, porém limitados".

Podemos observar que independente da experiencia prévia, e até mesmo em contextos diferentes, os desenvolvedores envolvidos em toda a pesquisa (sejam dos questionários, do experimento ou do exemplo de uso), tendem a ter veredictos bastante semelhantes (com pequenas mudanças) em relação às abordagens. Ao avaliar os dados fornecidos pelos mesmos, observamos que quando independente de contexto, a abordagem escolhida tende à abordagens nativas.

Como trabalhos futuros, podemos considerar um experimento semelhante ao que foi apresentado no Capítulo 6. Porém diferentemente da aplicação executada neste trabalho (a qual utilizou diversos recursos nativos do aparelho), este trabalho futuro pode realizar validações no guideline a partir da implementação (em ambas as abordagens nativa e híbrida) em diferentes tipos de aplicativos (como os mencionados na Figura 19); podendo assim avaliar aspectos e afirmações diferentes declaradas pelo guia comparativo resultante deste TCC.

Referências

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, CompSysTech 2016, Palermo, Italy, June 23-24, 2016*. [s.n.], 2016. p. 41–48. Disponível em: <http://dl.acm.org/citation.cfm?id=2983484>. Citado 2 vezes nas páginas 27 e 28.

AXELSSON, O.; CARLSTRöM, F. *Evaluation Targeting React Native in Comparison to Native Mobile Development*. 2016. Student Paper. Citado na página 31.

BERNARDES, T. F.; MIYAKE, M. Y. Cross-platform Mobile Development Approaches: A Systematic Review. *IEEE Latin America Transactions*, v. 14, n. 4, p. 1892–1898, abr. 2016. Citado 2 vezes nas páginas 21 e 22.

CHARLAND, A.; LEROUX, B. Mobile Application Development: Web vs. Native. *Commun. ACM*, v. 54, n. 5, p. 49–53, 2011. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1941487.1941504>. Citado na página 29.

CIMAN, M.; GAGGI, O. An empirical analysis of energy consumption of cross-platform frameworks for mobile development. *Pervasive and Mobile Computing*, v. 39, p. 214–230, 2017. Citado na página 27.

CORRAL, L.; JANES, A.; REMENCIUS, T. Potential Advantages and Disadvantages of Multiplatform Development Frameworks - A Vision on Mobile Environments. *Procedia Computer Science*, v. 10, p. 1202–1207, jan. 2012. ISSN 1877-0509. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050912005303>. Citado 2 vezes nas páginas 25 e 27.

DALMASSO, I. et al. Survey, comparison and evaluation of cross platform mobile application development tools. In: *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*. [S.l.: s.n.], 2013. p. 323–328. ISSN 2376-6492. Citado na página 30.

EL-KASSAS, W. S. et al. Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Engineering Journal*, 2015. ISSN 2090-4479. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2090447915001276>. Citado 2 vezes nas páginas 21 e 25.

GAOUAR ABDELKRIM BENAMAR, F. T. B. L. Desirable requirements of cross platform mobile development tools. 2015. Citado 2 vezes nas páginas 23 e 30.

HEITKÖTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. In: _____. *Web Information Systems and Technologies: 8th International Conference, WEBIST 2012, Porto, Portugal, April 18-21, 2012, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 120–138. ISBN 978-3-642-36608-6. Disponível em: <http://dx.doi.org/10.1007/978-3-642-36608-6_8>. Citado 2 vezes nas páginas 25 e 26.

JIANG, S. Comparison of native, cross-platform and hyper mobile development tools approaches for ios and android mobile applications. *University of Gothenburg*, 2016. Citado na página 31.

JOHNSON, H. et al. *Methods and systems for providing platform-independent shared software components for mobile devices*. Google Patents, 2006. US Patent 6,986,148. Disponível em: <https://www.google.com/patents/US6986148>. Citado 2 vezes nas páginas 27 e 29.

LATIF, M. et al. Cross platform approach for mobile application development: A survey. In: *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. [S.l.: s.n.], 2016. p. 1–5. Citado 3 vezes nas páginas 22, 23 e 30.

LATIF, M. et al. Review of mobile cross platform and research orientations. In: *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS*. [S.l.]: IEEE, 2017. p. 1–4. Citado na página 33.

MAJCHRZAK, T.; GRØNLI, T.-M. Comprehensive analysis of innovative cross-platform app development frameworks. In: *Proceedings of the 50th Hawaii International Conference on System Sciences*. [S.l.: s.n.], 2017. Citado na página 30.

MALAVOLTA, I. Beyond native apps: web technologies to the rescue! (keynote). In: *Proceedings of the 1st International Workshop on Mobile Development, Mobile!SPLASH 2016, Amsterdam, Netherlands, October 31, 2016*. [s.n.], 2016. p. 1–2. Disponível em: <http://doi.acm.org/10.1145/3001854.3001863>. Citado 2 vezes nas páginas 26 e 29.

MARTINEZ, M.; LECOMTE, S. Towards the Quality Improvement of Cross-Platform Mobile Applications. In: *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. [S.l.: s.n.], 2017. p. 184–188. Citado 3 vezes nas páginas 22, 25 e 26.

P.K, R.; M.KANNAN. Online mobile application development using ionic framework for educational institutions. In: *International Journal of Advanced Research Methodology in Engineering & Technology*. [S.l.: s.n.], 2017. v. 1, n. 3. ISSN 24566446. Citado 2 vezes nas páginas 27 e 28.

PREZOTTO, E. D.; BONIATI, B. B. Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas. *Universidade Federal de Santa Maria, Trabalho de Conclusão de Curso*, 2014. Citado na página 30.